Learning Location from Vibration

Ilari Shafer
ishafer@cs.cmu.edu

Abstract

Identifying the "semantic" location of a phone — for example, determining if it is resting on a desk, in a pocket, or on a chair — is a growing area of interest as smartphones become more prevalent. Taking advantage of the accelerometers and vibrators common in these devices, we use a machine learning approach to classify semantic location of a phone, even when it is at rest on a solid surface. An analysis of features extracted from time-series acceleration data and an ensuing classification process reveals the effectiveness of the approach. This analysis culminates in an implementation of a decision tree that produces 78% accuracy for a two-class demonstration.

1 Introduction

1.1 Problem

Location-based services for mobile devices are a recent and growing area of interest. Whether for simple device tracking, location-specific behavior, or social and advertising applications, determining the location of a smartphone is becoming increasingly relevant [6].

Broadly, two sorts of location information are notable: coarse geographical position (e.g. GPS latitude) and contextual location data (e.g. is a phone in my pocket or on a desk?). This paper considers the latter: how we can classify the surface or object a phone is resting on? One particularly appealing way to obtain this information is by monitoring the variety of sensors available on modern smartphones.

1.2 Approach

Many learning systems that deliver location data take a sample of a device's ambient environment through sensors like an accelerometer and feed it to a classification algorithm. Here, we use an augmented tactic: we actuate the vibrator in the phone and simultaneously use its accelerometer to capture the response of the phone+surface system. This data is input to a classifier for the semantic location of a device.

Using this form of active, device-specific sensing and a classifier that can be stored and executed on a device distinguishes our approach in a few primary ways:

- It can distinguish location while at rest. Passive sensing approaches have difficulty discriminating surfaces when a device is at rest or context without activity. This aspect can be valuable, for example, in the use case of locating a misplaced smartphone.
- It begins to address one of the primary limitations of context-awareness systems: the need for extensive user training of a classifier. Our approach shifts some of this burden to be device-specific rather than user-specific by focusing on characteristics of the phone and surface (its accelerometer and physical response).
- It does not require a network connection to send data to a central classifier. Our prototype implementation (see Sec. 5) runs on a phone. Energy optimality is a primary concern for mobile devices, and communication is a particularly expensive operation [9].

2 Related Work

Gathering contextual information from a mobile device's surroundings to identify its semantic location is hardly an untouched area. For instance, interest and activity is exemplified by The Mobile Sensing Group at Dartmouth [1] and context-awareness research at Intel [5].

One of the seminal papers in mobile phone location detection from sensor context was SurroundSense [2]. The authors use sensor data from a microphone, a light sensor, the wireless radio, and passive accelerometer data to produce a "fingerprint" of location. The accelerometer data was first averaged, then labeled using a support vector machine into two classes: moving and not moving.

Since then, other simple feature extraction techniques have also been fruitful; for example, one effort uses the mean, standard deviation, and number of peaks of accelerometer data as inputs to a decision tree [12]. KNN has also been used on x, y, z acceleration values [13]. For more output labels, another project transformed time-series data to the frequency domain and used signal mean, standard deviation, energy, and correlation as features for five classifiers [16].

A general survey of the area [8] identifies the need for individual user training as an obstacle. A number of approaches have aimed to reduce the extent of such training. Kern et al use solely a running window of mean and variance from multiple accelerometers as input to a Bayesian classifier for human activity [7]. The "Darwin Phones" project also pools more sensors for higher accuracy, but instead obtains them from multiple devices and users and combines the resulting models [11].

3 Method

3.1 Measurement

Our work clearly depends upon obtaining sensor data. We actuate the device and gather accelerometer measurements for a location (hereby referred to as "fingerprints"). To do so, we record accelerometer data (x-, y-, and z- axis acceleration) at the fastest rate we can obtain ($\approx$ 50Hz). First, after pressing a button on the device to begin the gathering process, we allow the device to rest so that the effects of pressing the button and previous movement are removed. We then obtain the response for both the passive sensing portions and vibrator-actuated system response, as well as any "impulse response" transition between the two states. A timeline for this process is shown in Figure 1.

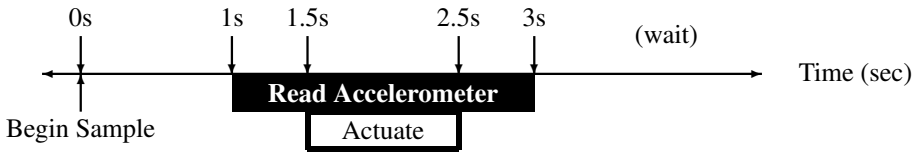


Figure 1: **Fingerprint Acquisition**: We gather both passive data (1-1.5s, 2.5-3s) and active data (1.5-2.5s) from the system. Buffer periods surround data capture to remove the effects of perturbation from a previous sample or starting measurement.

Since our goal is to shift the training burden to more extensive device-specific training, we gather a relatively large number of training fingerprints for each location we choose. We believe these choices exemplify a few locations a user might leave a device; though certainly not exhaustive, they serve as a small representative sample. The locations for which we obtain fingerprints are described in Table 1.

3.2 Signal Conditioning and Challenges

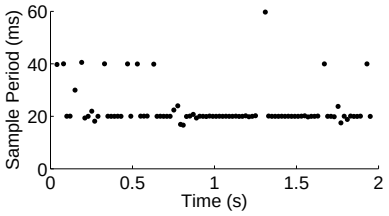
A significant obstacle to use of accelerometer data from the smartphones we assess is the uneven sample rate of the device, accompanied by significant sensor inaccuracy, and drift.

First, though we can request a nominal sample rate from the device sensor, an examination of the timestamps on the resulting output reveals that there are often significant gaps in the otherwise constant sample period. At left in Fig. 2 is the observed sensor sample period for one instance. To ensure that all instances are a time-series vector of the same length, we interpolate samples for missing data that are equal to the next value in the time series, as demonstrated at right in Fig. 2 with filled values in

Location	# Fingerprints	Description
desk_home	221	Resting on a glass surface
pocket_walk	236	In a pocket while user is walking
seat_bus	234	Multiple locations on a seat on a moving bus
seat_deskhome	237	Multiple locations on a leather seat
table_dining	230	A wooden table
table_kitchen	223	A kitchen countertop
table_placemat	235	Resting on a placemat on a rigid surface

Table 1: **Labels and Locations**: We capture multiple fingerprints for the same device from a variety of locations, representative of a few different characteristics we would expect of surfaces.

italic. We use this technique for simplicity of processing, as most gaps are one or two samples wide. Adding samples where cumulative skew due to delay exceeds one sample period is performed in the same manner.



	Expected Sample Index				
	1	2	3	4	5
$\mathbf{a}_x$	*-0.65*	-0.65	*-0.61*	-0.61	-0.61
$\mathbf{a}_y$	*-0.61*	-0.61	*-0.61*	-0.61	-0.61
$\mathbf{a}_z$	*-10.03*	10.03	*9.99*	9.99	10.03

Figure 2: **Adding sample fill**: We observe that samples are often missing or delayed from the constant sample rate we desire. Filled values are shown in italic. The data shown is from the desk_home location.

After collecting the data described in Sec 3.1, we observed a more fundamental problem with the sensor. The mean of accelerometer measurements changes over time as it becomes uncalibrated (drifts). This drift happens over even relatively short time intervals, as shown in Fig. 3 for data without vibration (that is, the phone was merely resting on a solid surface).

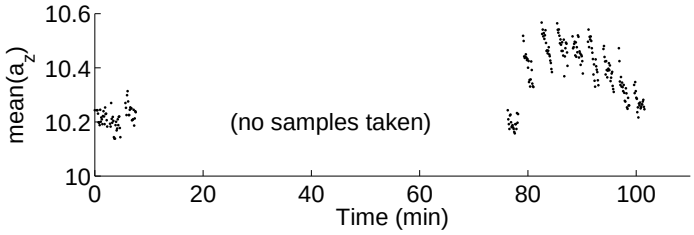


Figure 3: **Sensor drift**: Illustrated are the means of all z-axis acceleration measurements for a set of instances gathered at different times. A clear change in the sensed value of gravity is perceptible from one set of measurements to the next. A calibrated sensor should generate a near-constant value.

This drift becomes a challenge since for practicality we collect data in batches — for example, Fig. 3 shows two such batches separated by just over an hour. Then, if we use the offset of data as a feature, we will include information about which batch the sample belonged to, which is strongly correlated with its label. We attempt to account for this issue by subtracting the offset (which we define as the first sample) from all samples, although other sensor behavior differences over time still remain.

3.3 Reduced Feature Extraction

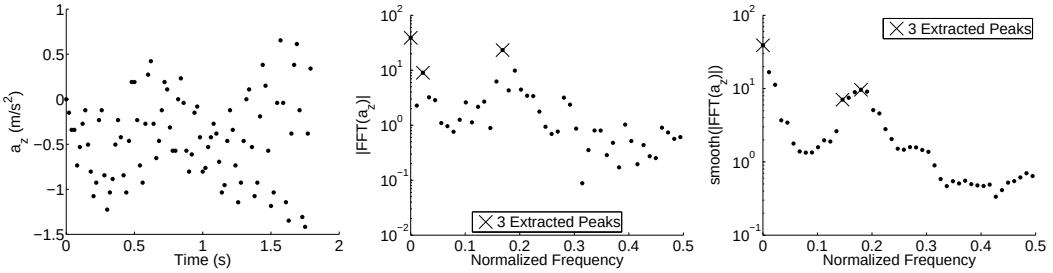
One of the core requirements of a learning algorithm based on these time-series fingerprints is the identification of a summary set of features that allows a learning algorithm to distinguish instances.

We begin with a consideration of some of the common features used in the literature: standard deviation and peak-to-peak amplitude [12, 13, 16] on each of the three axes. We are forced to ignore the

mean due to the drift described in Sec. 3.2. Additionally, we are less interested in the mean, since it corresponds to the tilt of the device (which is often very different for identical surfaces). The signal standard deviation provides a close predictor of its RMS power.

Furthermore, we use the frequency-domain representation of the signal as part of our feature space to provide information about the response of the system. Frequency-domain analysis of a vibration response has been used extensively for system identification [14]; however, these analyses typically require knowledge of actuator/sensor placement and the nature of the input signal. Here, we consider a simplified analysis based on identification of the strongest frequencies in the spectrum.

Our definition of "strongest" frequency is based upon peaks in a Fast Fourier Transform (FFT) of the time-series acceleration. We choose the z-axis and limit the time range of the FFT to the time when the device is vibrating. For a device in a resting position on a surface, we expect this configuration to provide the most system-unique information. We normalize frequencies to $\max(f) = 1$, since all fingerprints are taken with the same sample rate and duration. To accommodate for the relatively small number of accelerometer samples (50) in this range, we use a mixed-radix FFT [10].



(a) z-axis acceleration (b) FFT with identified peaks (c) FFT with smoothed peaks

Figure 4: **Frequency Domain Extraction**: We select n of the strongest frequencies in the norm of the signal FFT. This example illustrates the selection of three "peaks" from a sample in the seat_bus location.

To obtain features from the resulting data, we select the n strongest "peaks" of the frequency spectrum, which we identify as the frequencies with the greatest norm of a smoothed FFT. Due to the short length of time series data, the raw frequency spectra are noisy. We use a sliding-window average , which compromises absolute comparison but maintains relative peak location and amplitude. Figure 4(c) illustrates the selection of $n = 3$ peaks from a signal; we search for absolute maximum values (not local maxima), and forbid multiple peaks within the width of the sliding window average.

Since we smooth the FFT, we are also interested in how much variance was captured by this smoothing process; to do so, we use a metric noise$(\mathbf{x}, \mathbf{s})$ between the original spectrum $\mathbf{x}$ and the smoothed spectrum $\mathbf{s}$:

$$\text{noise}(\mathbf{x}, \mathbf{s}) = \sum_{i=1}^{n-w} (s_{i-w} - x_i)^2$$

This definition is reminiscent of "slowness" used in the neural signal processing community [17]. This noise metric is another feature considered in the reduced feature set.

Finally, since our goal for these reduced features is to obtain a small feature vector, we perform feature selection using a chi-square independence statistic on the training data. We find that the peak *locations* have the smallest contribution, so we retain only the peak amplitudes from the smoothed FFT. Our resulting reduced feature space (with summary statistics and frequency information) lies in $\mathcal{R}^{11}$.

3.4 Raw Feature Extraction

Separately, we consider another feature space, composed of the raw timeseries data. As mentioned in Sec. 3.2, we ensure that each instance is interpolated to a vector of deterministic length n:

$$\mathbf{a}_x = [x_1, \ldots, x_n] \qquad \mathbf{a}_y = [y_1, \ldots, y_n] \qquad \mathbf{a}_z = [z_1, \ldots, z_n]$$

Since the number of resultant features is simply $3n$, the feature space grows linearly with the sampling time. Therefore, we consider a downsampled signal $\mathbf{a}_{d_{ds}}$ on each axis d formed by taking the mean of a bin of w samples (and also analyze the feature space for $w = 1$, the original signal):

$$\mathbf{a}_{d_{ds}} = \left[\frac{\sum_{i=0}^{w-1} d_i}{w}, \frac{\sum_{i=w}^{2w-1} d_i}{w}, \ldots, \frac{\sum_{i=n_{\text{ds}}w}^{n} d_i}{\min(w, n_{ds}w - n)} \right] \tag{1}$$

where the number of downsampled features along each axis n_{ds} is simply

$$n_{\text{ds}} = \left\lceil \frac{t_{\text{end}} - t_{\text{start}}}{\mathit{SamplePeriod} \cdot w} \right\rceil = \left\lceil \frac{n}{w} \right\rceil$$

This trivial downsampling may introduce aliasing, but given the inaccuracy of the original time-series data and the use of $\mathbf{a}_{d_{ds}}$ solely as input to a classifier, the concern is minimal.

3.5 Classifiers

We use a variety of classifiers for the features we select in Sec. 3.3 and 3.4. Our choice of classifiers is restricted to parametric classifiers, since we would like to be able to store the classifier in a small amount of space and evaluate it rapidly on the mobile device itself. Within this space, we choose a range of different classifiers in the spirit of evaluating different techniques. The classifiers are:

bayes.NaiveBayes A Naïve Bayes classifier, which is characterized by the conditional independence assumption.

trees.J48 An implementation of the C4.5 algorithm [15], which in turn expands upon the ID3 algorithm. Like ID3, it builds a decision tree based upon information gain, recursively splitting to create a tree.

functions.LibSVM A support vector machine classifier implemented using the popular LibSVM library [3]. We use a radial basis function kernel.

functions.Logistic A Logistic Regression classifier.

meta.AdaBoostM1 A classifier built on a weighted combination of decision stumps. This combination is built with the AdaBoost algorithm.

Our implementations for these standard classifiers are from the Weka machine learning library [4], rather than ones we construct. This library provides a consistent interface to these classifiers, which allows the feature extraction algorithm we write to be evaluated cleanly and uniformly.

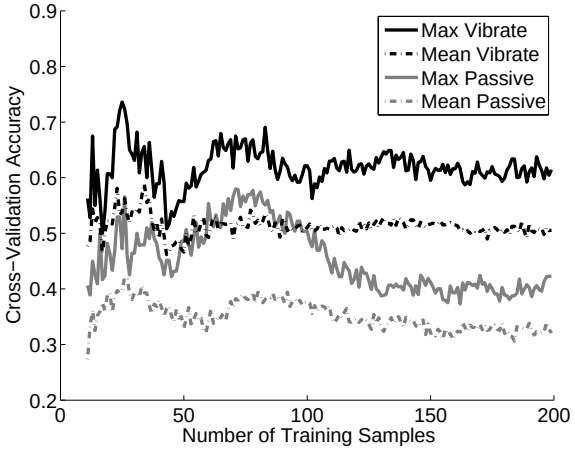
4 Results

4.1 Vibrate vs Passive Sensing

As mentioned in Sec. 1.2, one of the proposed advantages of performing active fingerprinting is that it enables us to distinguish location while at rest. To assess this hypothesis, we collect additional data when not activating the vibrator from four of the locations we use as classes. We then perform a comparison of identical models and features to determine if what (if any) benefit to classification accuracy is provided by the addition of vibration.

We find that, indeed, using the vibrator and accelerometer simultaneously to perform active sensing provides a considerable gain in performance as compared to simply using passive accelerometer information. Fig. 5 illustrates the performance of our chosen classifiers on vibrate and passive data. We study 10-fold cross-validation accuracy for a training set of identical size for both locations. Since this metric has high variance for few instances, we analyze it as a function of training set size and see that it stabilizes. The best classifier (shown in solid lines) is a J48 decision tree for both cases, for which using vibrate data yields an absolute $\approx 18\%$ gain in accuracy (a 56% improvement relative to the passive approach).

Also, we study classifier accuracy when used with the raw feature set, and find that the trend is even more pronounced. Using vibration information again outperforms passive sensing, but by a larger margin. For raw features with a bin width $w = 10$, SVM provides the best performance, and with vibrate data has an $\approx 30\%$ absolute gain in accuracy (a 115% improvement relative to passive sensing).



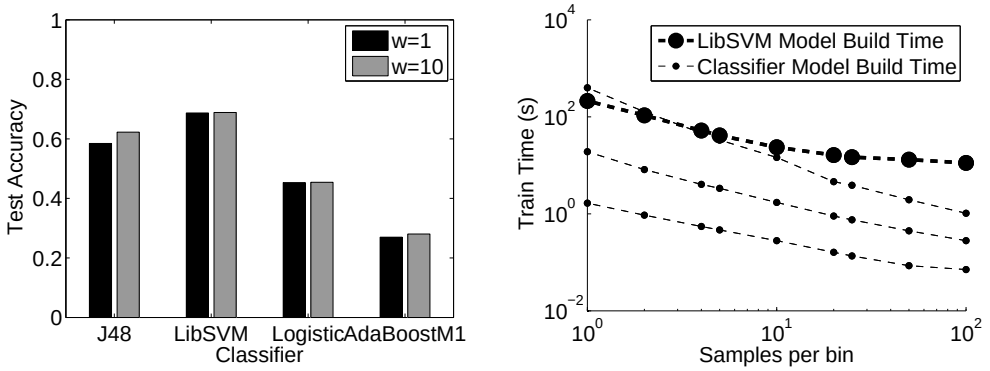
Location	# V	# P
desk_home	221	242
table_dining	230	240
table_kitchen	223	240
table_placemat	235	241

Number of vibrate (# V) and passive (# P) instances from each location. This is a subset of the same locations given in Table 1.

Figure 5: **Vibrate vs Passive Accuracy**: 10-fold cross-validation accuracy on the training set as a function of the number of training samples used is shown for vibrate (black) and passive (gray) data. The same set of classifiers and the reduced feature set is used for both cases. Note that the accuracy axis does not start at 0.

4.2 Raw Feature Performance

Another key question of interest is the performance of the raw feature set described in Sec. 3.4. To evaluate the set of classifiers for raw features, we consider test accuracy on the 30% of the collected data that was held out. This technique is not appropriate for model selection, but we do not use this evaluation to choose a final classifier — merely for a fair comparison to the same test accuracy used for the final results in Sec. 4.3 below. Running the same experiment with 10-fold cross-validation on the training set yields very similar results.



(a) Classifier accuracy for $w = 1$ and $w = 10$

(b) Training time as a function of bin width w

Figure 6: **Raw Feature Performance**: At left, test accuracy for bin width $w = 1$ and $w = 10$ (see Eq. 1). SVM has the best performance up to the $w = 25$ we consider, where accuracy drops rapidly. However, training time increases nearly linearly with the dimension of the feature space (not directly visible here). All locations in Table 1 are used.

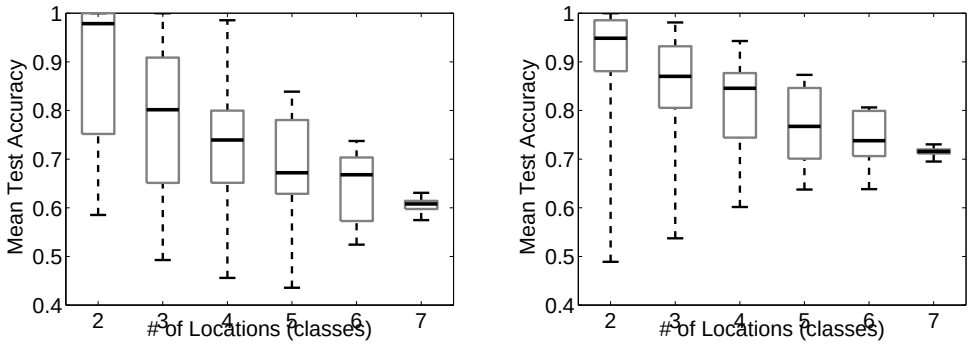
Fig. 6(a) illustrates the test accuracy for two bin widths. We choose SVM as the best classifier based on cross-validation accuracy (not shown); its test accuracy for $w = 10$ and all 7 locations as shown in the figure is 68.9%. This result is sensible: SVM is robust to the likely irrelevant features in the potentially large feature space produced by the raw feature set. For clarity, we omit Naïve Bayes due to low accuracy.

Since the raw feature set explicitly considers the relative means of the sensor data, however, further data collection is necessary to truly characterize its performance relative to the reduced feature set described in the next section (which does not consider the absolute values on x, y, and z axes).

4.3 Classifier Performance

Our primary analysis concerns a more detailed look at the performance of our system on the locations listed in Table 1. We are interested in how accurately the best classifier can be trained on some subset of these locations. For motivation, we imagine a user or device that is interested in discriminating $n \leq 7$ locations; the practical question then arises of how the system performs for some n.

On the basis of selection from 10-fold cross-validation, we choose a J48 decision tree as the best classifier for the reduced feature set and LibSVM as the best classifier for the raw feature set. Fig. 7 illustrates its performance using the reduced feature set as a function of the number of desired locations (classes). To gain intuition about the range of pairings, for each number of locations n we randomly select n locations 100 times and plot the range and median of the resulting accuracy.



(a) Reduced Features, J48

(b) Raw Features, SVM

Figure 7: **Test Accuracy vs Number of Classes**: We select groups of n locations, $2 \leq n \leq 7$, and evaluate accuracy for each group. Boxes lie between the 25^{th} and 75^{th} percentiles, their marks are at the median, and whiskers extend to the range of the data. Accuracies are shown for the classifier selected on the basis of cross-validation, then trained on 70% of the dataset and evaluated on the same 30% as Fig. 6(a). Note that the accuracy axis does not begin at 0.

Clearly, it is increasingly difficult to discriminate more classes. Of note is the wide range of accuracies: some pairs of locations are well-separated (for example, pocket_walk and desk_home), whereas "hard" surfaces are less distinguishable. As expected, as the number of locations grows larger, the variance of accuracy among the chosen classes diminishes. The small variance in the $n = 7$ location case is due to small nondeterminism in the classifier construction process itself.

5 Implementation

To demonstrate the effectiveness and practical application of the system described here, we have completed a prototype implementation of the J48 classifier on a mobile phone. Due to space limitations, a full description is not included here, but a brief description of its operation follows:

- **Training:** We collect and label training data with the procedure in Sec. 3.1 for two demonstration locations (in this case, a soft surface and a glass surface).
- **Model Construction:** We build a model using J48, one which is quite concise (19 tree nodes total for our 134 training instances). This model is placed on the mobile device.
- **Classification:** We have modified the reduced feature set transformation (statistics and FFT) and classifier (J48 tree) to run on the device. For demonstration purposes, the user can press a button to initiate a measurement and observe the resulting classification.

The two-class system described above has 10-fold cross-validation accuracy of 78%, an accuracy similar to that observed in demonstration. Videos of the result are available upon request.

6 Conclusions and Future Study

We have described a system for classifying the semantic location of a mobile device based solely upon accelerometer data. The smartphone uses feedback from induced vibration to identify the response of the system it rests upon. We extract features from the resulting time-series data and use it as input to a set of representative classifiers.

A consideration of the two types of features we study — a "reduced" feature set of size 11 distilled from summary statistics and a "raw" feature set of downsampled time series data — reveals that the raw feature set has superior performance. However, there exists a tradeoff between the number of raw features and the time required to train a classifier: the SVM built with $w = 10$ that is described in Sec. 4.2 requires 23 seconds to train, whereas the J48 decision tree described in Sec. 4.3 requires 0.5 sec. Additionally, for a true portrait of classification accuracy, it will be necessary to gather more data over a longer period of time to account for sensor drift (see Sec. 3.2 for detail).

Despite the obstacles inherent in using acceleration data from vibration, we can still use our system to distinguish "hard" locations. The real-world functionality of the implementation described in Sec. 5 and the result in Sec. 4.1 that vibration truly improves classification performance demonstrate the utility and effectiveness of the approach described herein.

Further data from more locations and types of smartphones gathered over a longer period of time would be valuable in validating and generalizing the present work. With the robustness afforded by broader data and its analysis, a refined implementation, and distribution to users, using our system could bring the promise of identifying semantic location closer to reality.

References

[1] Mobile sensing group, 2010.

[2] Martin Azizyan, Ionut Constandache, and Romit R. Choudhury. SurroundSense: mobile phone localization via ambience fingerprinting. In *MobiCom '09: Proceedings of the 15th annual international conference on Mobile computing and networking*, pages 261–272, New York, NY, USA, 2009. ACM.

[3] Chih-Chung Chang and Chih-Jen Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.

[4] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The weka data mining software: an update. *SIGKDD Explor. Newsl.*, 11:10–18, November 2009.

[5] Intel. Context awareness to radically change how we interact with technology, September 2010.

[6] Iris A. Junglas and Richard T. Watson. Location-based services. *Commun. ACM*, 51(3):65–69, 2008.

[7] Nicky Kern, Bernt Schiele, and Albrecht Schmidt. Multi-sensor activity context detection for wearable computing. In *Ambient Intelligence*, volume 2875 of *Lecture Notes in Computer Science*, pages 220–232. Springer Berlin / Heidelberg, 2003. 10.1007/978-3-540-39863-9_17.

[8] Nicholas Lane, Emiliano Miluzzo, Hong Lu, Daniel Peebles, Tanzeem Choudhury, and Andrew Campbell. A survey of mobile phone sensing. *IEEE Communications Magazine*, 48(9):140–150, September 2010.

[9] Marius Marcu, Dacian Tudor, and Sebastian Fuicu. Towards a network-device unified framework for power-aware wireless applications. In *IWCMC '09: Proceedings of the 2009 International Conference on Wireless Communications and Mobile Computing*, pages 963–967, New York, NY, USA, 2009. ACM.

[10] Bruce R. Miller. Open source physics FFT.

[11] Emiliano Miluzzo, Cory T. Cornelius, Ashwin Ramaswamy, Tanzeem Choudhury, Zhigang Liu, and Andrew T. Campbell. Darwin phones: the evolution of sensing and inference on mobile phones. In *MobiSys '10: Proceedings of the 8th international conference on Mobile systems, applications, and services*, pages 5–20, New York, NY, USA, June 2010. ACM.

[12] Emiliano Miluzzo, Nicholas D. Lane, Kristóf Fodor, Ronald Peterson, Hong Lu, Mirco Musolesi, Shane B. Eisenman, Xiao Zheng, and Andrew T. Campbell. Sensing meets mobile social networks: the design, implementation and evaluation of the cenceme application. In *SenSys '08: Proceedings of the 6th ACM conference on Embedded network sensor systems*, pages 337–350, New York, NY, USA, 2008. ACM.

[13] A. Ofstad, E. Nicholas, R. Szcodronski, and R. R. Choudhury. Aampl: Accelerometer augmented mobile phone localization. *MELT'08 San Francisco, California, USA.*, September 2008.

[14] Rik Pinelton and Johan Schoukens. *System identification: a frequency domain approach*. IEEE Press, 2001.

[15] Ross J. Quinlan. Learning with continuous classes. In *5th Australian Joint Conference on Artificial Intelligence*, pages 343–348, Singapore, 1992. World Scientific.

[16] Nishkam Ravi, Nikhil Dandekar, Prreetham Mysore, and Michael L. Littman. Activity recognition from accelerometer data. *American Association for Artificial Intelligence*, 2005.

[17] Henning Sprekeler, Christian Michaelis, and Laurenz Wiskott. Slowness: An objective for spike-timingdependent plasticity? *PLoS Comput Biol*, 3(6):e112, 06 2007.