Online pose classification and walking speed estimation using handheld devices

Citation: Jun-geun Park, Ami Patel, Dorothy Curtis, Seth Teller, and Jonathan Ledlie. 2012. Online pose classification and walking speed estimation using handheld devices. In Proceedings of the 2012 ACM Conference on Ubiquitous Computing (UbiComp '12). ACM, New York, NY, USA, 113-122.

As Published: http://dx.doi.org/10.1145/2370216.2370235

Publisher: Association for Computing Machinery (ACM)

Persistent URL: http://hdl.handle.net/1721.1/90925

Version: Author's final manuscript: final author's manuscript post peer review, without publisher's formatting or copy editing

Online Pose Classification and Walking Speed Estimation using Handheld Devices

Jun-geun Park, Ami Patel, Dorothy Curtis, Seth Teller
MIT CS & AI Laboratory (CSAIL)
Cambridge, MA, USA
{jgpark,ampatel,dcurtis,teller}@csail.mit.edu

Jonathan Ledlie
Nokia Research Center
Cambridge, MA, USA
jonathan.ledlie@nokia.com

ABSTRACT
We describe and evaluate two methods for device pose classification and walking speed estimation that generalize well to new users, compared to previous work. These machine learning based methods are designed for the general case of a person holding a mobile device in an unknown location and require only a single low-cost, low-power sensor: a triaxial accelerometer. We evaluate our methods in straight-path indoor walking experiments as well as in natural indoor walking settings. Experiments with 14 human participants to test user generalization show that our pose classifier correctly selects among four device poses with 94% accuracy compared to 82% for previous work, and our walking speed estimates are within 12-15% (straight/indoor walk) of ground truth compared to 17-22% for previous work. Implementation on a mobile phone demonstrates that both methods can run efficiently online.

Author Keywords
Handheld device, Activity classification, Machine learning.

ACM Classification Keywords
I.5.4 Pattern Recognition: Applications: Signal Processing.

General Terms
Algorithms, Experimentation, Measurement, Performance.

INTRODUCTION
For mobile handheld devices to intelligently react to their environment, they must infer user activities from raw sensor data. We anticipate that descriptions of low-level activities, rather than raw sensor data, will be exposed to application developers in future mobile operating systems, much like touch screen gestures are today. Developers will then stitch these activities together with context, such as the user's calendar entries and friends' locations, to yield intelligent high-level behavior. Inference of activity type from raw sensor data is a prerequisite for effective context-aware behavior.

We study two "activities" in particular: the device's position, or *pose*, relative to the body, and the user's walking speed. Inferring the pose of the device has many uses, as Kunze *et al.* [11] and Miluzzo *et al.* [13] have discussed. For example, in global sensing applications (*e.g.,* pollution monitoring) knowing whether the data-collecting phone is inside a bag or pocket is crucial for labeling the data and determining if it is worthwhile to turn on a more resource-intensive pollution sensor. Device pose is also an important input into other classifiers (*e.g.,* mode of transport) as the output from many sensors can be affected by pose. Walking speed estimates have numerous applications ranging from dead reckoning to indoor and outdoor navigation to health monitoring. In particular, we plan to use speed estimation to approximate people's locations in a building as part of our on-going work on crowd-sourced positioning systems [14]. We selected these two activities because they exemplify what will be exposed to developers, and because they can be categorized using similar methods from the same sensors.

We form estimates of these two activities with closely related methods using a single triaxial accelerometer. Previous work on inferring these activities has tended to use more sensor modalities and, in the case of speed estimation, to place several sensors at fixed positions on the user's body. While fixed sensors are realistic in specific medical contexts, such as elder care support [12], we are focused on the more general case of a person carrying a personal handheld device, such as a mobile phone or a tablet. With this case in mind, our experimental participants walked in a normal indoor environment, including up and down stairs, with a mobile device held in a variety of regular poses.

This paper presents single-sensor methods for device pose classification and walking speed estimation that generalize well to standard mobile phones and tablets. In particular, the paper makes the following contributions:

- We give device pose classification and walking speed estimation methods that are predictive for new users, and do not assume known sensor orientation or placement;
- Our approach, based on regularized kernel methods, can

UbiComp '12, Sep 5-Sep 8, 2012, Pittsburgh, USA.

be efficiently implemented online; and

- Through a leave-one-participant-out generalization experiment with 14 participants, we show that online pose estimation accuracy was 94%, and online speed estimator had an error of less than 12-15% compared to truth.

BACKGROUND

Pose Classification

Because the pose classification problem by definition requires a personal mobile device to classify — a recent phenomenon — it is a significantly newer problem than walking speed estimation. Before this phenomenon, however, the area of *activity recognition* addressed several closely related topics. For example, Bao and Intille [2] classified physical activities from acceleration data obtained from biaxial accelerometers attached to five different locations of the body. Ravi *et al.* [16] compared various base-level classifiers and meta-classifiers such as bagging or boosting methods for the activity recognition task. These studies provide useful insight on what acceleration signal features and learning algorithms are useful for pose classification.

Pose classification has been studied as a prior stage of activity recognition. Some previous work used the knowledge of the device pose to guide activity recognition. For example, Khan *et al.* [10] use a cascaded classifier for their accelerometer-based position-independent activity recognition system. They first determine whether the accelerometer is affixed to the upper or lower half of the body, then classify the acceleration signal separately. While binary pose classification has proved effective for activity recognition, we aim both for a broader range of applications and for finer-grained pose classification. Kawahara *et al.* [9] use heuristics to infer sensor position, then separately determine whether a subject is walking or running. However, their approach may not be applicable to a wider variety of device poses.

Others have attempted device pose classification using sensors other than accelerometers. For example, Miluzzo *et al.* [13] studied how to determine whether the phone is inside or outside of a person's pocket, using acoustic signals measured by the phone's microphone.

Our work is most closely related to that of Kunze *et al.* [11], who also approach the pose classification problem using classification algorithms. However, our method uses features that capture the orientation of the device and an algorithm capable of learning the complex nonlinear relationships among the data without overfitting. In our experiments, our method shows superior pose classification accuracy when applied to new users.

Walking Speed Estimation

Accelerometer-based walking speed estimation methods can be divided into two categories: (a) machine learning-based methods and (b) model-based methods. Our work employs a machine learning approach, which has the potential to exploit associative information within data beyond an explicit model chosen by the system designer. However, most previous work that applies machine learning methods to walking speed estimation assumes that there are one or more accelerometers at *fixed* positions on the body [8,20]. For example, Vathsangam *et al.* [20] use Gaussian process regression to estimate walking speed from one accelerometer fixed on a subject's hip. In contrast, our work estimator does not assume that the accelerometer, embedded in the user's mobile device, is in any particular position.

In machine learning-based activity recognition, there have been several different approaches to handling varying sensor positions when trying to identify a user's behavior (*e.g.* walking, running, climbing stairs). For example, Yang [22] addresses different phone positions by projecting data from the accelerometer's x, y, and z axes onto the horizontal and vertical axes to obtain orientation-independent features.

An example of a model-based system, on the other hand, is an accelerometer-based pedometer, which counts walking steps. These approaches often strongly depend on domain-specific knowledge, such as stride length, which may require manual input. The AutoGait system [4] avoids manual input by combining the pedometer with GPS to infer the linear relationship between step frequency and stride length. Other pedestrian navigation systems have also studied speed estimation, often using inertial and magnetic sensors along with heuristic- or rule-based speed estimation [6, 15]. Another method posits a linear relationship between a metric representing step bounce and stride length [21]. It has been employed for speed estimation when the device pose is unknown [1].

APPROACH

We approach both pose classification and speed estimation as supervised learning problems that learn, from labeled training data, an association from the input (the acceleration signal) to the output (device pose, or walking speed). Specifically, for estimation of user walking speed and mobile device pose, we use *regularized kernel methods* (or *regularization networks*) [5, 18], which combine kernel methods and regularization. *Kernel methods* are a family of statistical learning methods that exploit training data through implicit definition of a similarity measure between data points, from which a learning algorithm predicts the output of a new data point. On the other hand, *regularization* improves the predictive capability of a fitted model on new data points, by preventing overfitting of the model on the training data.

In this work, we tackle device pose classification and user walking speed estimation independently, using two different classes of regularized kernel methods. Intuitively, one could imagine that a speed estimation model could be affected by phone pose — a speed estimation model with the device in hand may be different from that with the device in pocket. However, we found that the performance improvement in speed estimation from knowing the phone pose was very small when using our method. Because of this minor improvement in accuracy, and because keeping the processes separate makes an online implementation more viable, we

decided to keep them independent.

For device pose classification, we use *support vector machines* (SVM), a well-known kernel method, to categorize accelerometer input as originating from one of four distinct device poses. For walking speed estimation, we use *regularized least squares* (RLS), a regression algorithm that benefits from regularization and kernels, as opposed to traditional ordinary linear regression. We employ these regularized kernel methods as they do not make any explicit modeling assumptions, and they are known to perform well for many task domains.

Algorithm Implementation

In the implementation, the time-series acceleration signals collected are partitioned with a sliding window into n segments, from which features are extracted to form a feature vector $\mathbf{x}_i,\ 1 \leq i \leq n$. The trained algorithm can then predict a label y_i for segment i based on rules learned from data in a previously collected training set.

For a detailed explanation of regularized kernel methods, readers may refer to Schölkopf and Smola [18], Bishop [3], or Evgeniou *et al.* [5]. We note only that both algorithms, support vector machines and regularized kernel methods, can be formulated in the Tikhonov regularization framework [5], and have the same functional form for their prediction function for a new input x' as

$$f(\mathbf{x}') = \sum_{i=1}^{n} c_i^* K(\mathbf{x}', \mathbf{x}_i) \tag{1}$$

which is a finite expansion of symmetric, real-valued *kernel functions* $k(\mathbf{x}', \mathbf{x}_i)$, with the optimal coefficient c_i^* for the i-th training example x_i. Therefore, once each algorithm computes the optimal coefficients offline at the training phase, a host computer where predictions occur (*i.e.,* a mobile device) need only store training examples $\mathbf{x}_i$ along with its optimal coefficients c_i^* to compute a prediction $f(\mathbf{x}')$. The computation is linear in the number of training examples, and can be done in tens of milliseconds on modern handheld devices, as we show in the online evaluation.

We also note that one can derive a task-specific kernel function by combining two or more kernels. Given two valid kernels $K_1(\mathbf{x}, \mathbf{x}')$ and $K_2(\mathbf{x}, \mathbf{x}')$, the following are also valid:

$$K_3(\mathbf{x}, \mathbf{x}') = cK_1(\mathbf{x}, \mathbf{x}') \tag{2}$$

$$K_4(\mathbf{x}, \mathbf{x}') = K_1(\mathbf{x}, \mathbf{x}') + K_2(\mathbf{x}, \mathbf{x}') \tag{3}$$

where $c > 0$ is a constant. We use these properties to combine different features into one kernel matrix.

Feature Extraction

We extract features from the time-series acceleration signals by partitioning them with a sliding-window of m samples ($m = 256$ or 512), depending on the evaluation scenario, with 50% overlap between subsequent windows. At a sampling rate of 100 Hz, each window corresponds to a time period of 2.56 or 5.12 seconds. We then represent each segment as a vector of features based on the accelerometer signal.

The primary features we use are the magnitudes of the low-frequency portion of the acceleration signal spectrum, as in previous work [2, 16]. Because human walking is cyclic, the frequency components of the discrete Fourier transform (DFT) of the acceleration signal contain information about walking motion. For example, the largest peak in the spectrum (usually located around 2 Hz) corresponds to the gait cycle, the spectral energy represents walking intensity, and the shape of the spectrum represents how each individual walks and/or where the phone is placed on the body.

Let $\mathbf{a}_j = (x_j, y_j, z_j),\ 1 \leq j \leq m$, represent one sample within the window. Since we want our method to be robust with respect to device orientation, we first compute three different components of the triaxial accelerometer data: the acceleration magnitude, $m_j = \|\mathbf{a}_j\|$, and the magnitudes of the vertical and horizontal components of the acceleration, v_j and h_j.

The horizontal and vertical components of the acceleration are computed as in [22]; we first estimate the unit gravity vector by taking the mean of the x, y, and z components over the entire window, and normalizing the result:

$$\hat{\mathbf{g}} = \frac{(\overline{x}, \overline{y}, \overline{z})}{\|(\overline{x}, \overline{y}, \overline{z})\|}. \tag{4}$$

The magnitude of the vertical component of the acceleration vector $\mathbf{a}_j$ is then computed by taking the dot product of the gravity vector and the original acceleration vector:

$$v_j = \mathbf{a}_j \cdot \hat{\mathbf{g}} \tag{5}$$

We can also compute the projection of the acceleration onto the vertical component:

$$\mathbf{v}_j^{proj} = v_j \hat{\mathbf{g}} \tag{6}$$

The magnitude of the horizontal acceleration is found by subtracting the vertical projection from the original acceleration vector, and computing the magnitude of the result:

$$\mathbf{h}_j^{proj} = \mathbf{a}_j - \mathbf{v}_j^{proj} \tag{7}$$

$$h_j = \|\mathbf{h}_j^{proj}\|. \tag{8}$$

The DFT components of the acceleration magnitude are used for speed estimation, while the DFT components of the horizontal and vertical acceleration are used for pose classification. We compute the DFT components using a 512-point FFT, after subtracting the mean value (*i.e.,* DC component) from each sample. Because the low-frequency bands typically represent human motion, while higher-frequency bands may contain noise, we use only the first 60 components (up to 11.7 Hz).

While prior work on activity recognition employs various acceleration data features including mean, spectral entropy, and cepstral coefficients, we found that a DFT vector with one set of additional features for each task was sufficient

to achieve good performance. We next describe what additional features are used for each task, and how to combine features to compute a kernel function between data points.

Features and Kernels for Pose Classification

For device pose classification, we use a vector composed of the DFTs of the horizontal and vertical components of the acceleration signal as well as tilt features derived from the gravity vector. The decomposition into horizontal and vertical components is used due to the different oscillation patterns of the mobile device along the horizontal and vertical plane when it is placed in different positions of the body. For example, when the mobile is in a trouser pocket, the horizontal movement is stronger than when it is in the user's hand. Overall acceleration magnitude did not capture this difference very well.

However, we found that using only horizontal and vertical DFT features was insufficient for achieving high-accuracy device pose classification. The DFT vectors alone did not give a clear separation between different device poses (Figure 1a). Therefore, we added a set of features derived from the gravity vector, obtained by averaging the acceleration signal over the entire segmentation window. However, we did not use the raw gravity vector (Figure 1b). Our choice of features is inspired by the fact that some device poses have a canonical tilt angle when the phone is held naturally. For instance, when a user makes a phone call and places the phone at the ear, it is mostly upright with a slight tilt outward from the face. Similarly, when the phone is in a trouser pocket, it is not usually positioned with its front or back pointing down.

Thus, we compute the following gravity tilt feature vector:

$$\mathbf{x}_G = \left(|g_x|, |g_y|, |g_z|, \sqrt{g_x^2 + g_y^2}, \sqrt{g_x^2 + g_z^2}, \sqrt{g_y^2 + g_z^2}\right) \tag{9}$$

where g_x, g_y, and g_z are gravity components along x-, y-, and z-axis of the accelerometer respectively. The first three elements of the gravity feature represent one-dimensional projections of the gravity vector, while the last three represent two-dimensional projections on the x-y, x-z, or y-z plane. The pair $|g_z|$ and $\sqrt{g_x^2 + g_y^2}$ represent the tilt angle along the x-y plane of the device and are invariant to rotation along the z-axis. This representation gave a good separation between different poses (Figure 1c). Let $\mathbf{x}_{HV}$ denote a vector composed by concatenating horizontal and vertical DFT vectors. Our final feature vector for the pose classification task is then $\mathbf{x} = (\mathbf{x}_{HV}, \mathbf{x}_G)$.

We sum two radial basis function (RBF) kernels for the DFT vectors and the gravity tilt feature to obtain the pose classification kernel function. We take the weighted sum of these kernels as:

$$K_P(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\|\mathbf{x}_{HV} - \mathbf{x}'_{HV}\|^2}{2\sigma_{HV}^2}\right) + \omega_G \exp\left(-\frac{\|\mathbf{x}_G - \mathbf{x}'_G\|^2}{2\sigma_G^2}\right) \tag{10}$$

with $\omega_G = 0.1$, an empirically determined parameter. The kernel widths σ_{HV} and σ_G could be chosen empirically to maximize cross-validation performance. However, doing so leaves the problem of simultaneously searching for optimal values of three parameters: σ_{HV}, σ_G, and the regularization parameter λ. Instead we compute the median pairwise distance among all $\mathbf{x}_{HV}$ and among all $\mathbf{x}_G$ in the training dataset, and fix σ_{HV} and σ_G to be half the respective median distances.

Features and Kernels for Speed Estimation

In addition to overall acceleration magnitude, we compute the energy, or the sum of the squared magnitudes of the components of the signal, and use it as an additional feature for speed estimation. We incorporate this quantity because the energy represents walking intensity, which shows a good correlation with walking speed ($R^2 = 0.615$ in our data). By Parseval's theorem, the signal energy can be estimated in the frequency domain by summing squared frequency magnitudes. That is, given a DFT vector of the acceleration magnitude signal m_j where the magnitude of each coefficient is taken $(|M(f_1)|, ..., |M(f_N)|)$, the energy is

$$x_E = \sum_{j=1}^{N} |M(f_j)|^2. \tag{11}$$

In order to make the DFT vector independent of the overall energy (so that the two different feature sets capture different aspects of walking), we normalize the DFT magnitude vector by the square root of the energy . Let this normalized vector $\mathbf{x}_M = (|M(f_1)|/\sqrt{x_E}, ..., |M(f_N)|/\sqrt{x_E})$. Then, our final feature vector is $\mathbf{x} = (\mathbf{x}_M, x_E)$.

We combine the two feature sets in a way analogous to pose classification. The kernel function for speed estimation is the sum of RBF kernels defined by the FFT and by the energy:

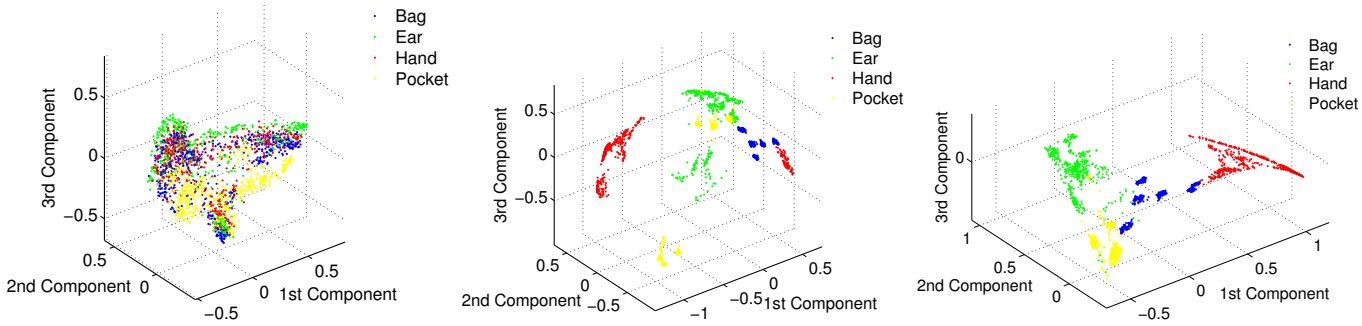
$$K_S(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\|\mathbf{x}_M - \mathbf{x}'_M\|^2}{2\sigma_M^2}\right) + \exp\left(-\frac{(x_E - x'_E)^2}{2\sigma_E^2}\right) \tag{12}$$

where both kernels are given the same weight. The kernel widths σ_M and σ_E are determined similarly to the corresponding parameters used in pose classification: each is half the respective median pairwise distance.

HARDWARE AND ONLINE IMPLEMENTATION

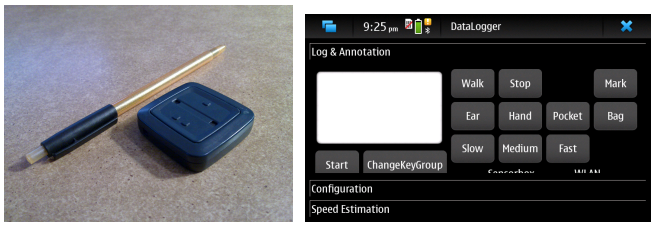
To demonstrate that our method can be directly applied to a commodity device, we used a Nokia N900 mobile phone and a Nokia Sensorbox to implement an online version of the pose and speed estimation algorithms.

The sensing device we used for experimentation and the online implementation is the Nokia Sensorbox (Figure 2a), developed for research and containing five sensing modalities in a compact package: a consumer-grade accelerometer, gyroscope, magnetometer, thermometer, and barometer. We use only the accelerometer signal, both for simplicity and



(a) Horizontal/vertical DFT vector (b) Raw gravity vector (c) Gravity tilt feature vector (Equation (9))

Figure 1. Low-dimensional visualization of feature vectors used for device pose classification. Each feature set is visualized in a three-dimensional space by PCA. The DFT vector of horizontal/vertical component alone does not give a clear separation between different poses (1a). While raw gravity vector gives separation between poses, several clusters exist for each pose, making the classification algorithm unable to find a good decision boundary between poses (1b). Our representation of the gravity vector (Equation (9)) places data points from the same pose adjacent to each other (1c).



(a) (b)

Figure 2. Nokia Sensorbox and data logging program

because this was generally sufficient to achieve high performance on our tasks.

The Nokia Sensorbox was connected to a host computer, a Nokia N900 mobile phone, via Bluetooth. We wrote a data logging program (Figure 2b) that records sensor data transferred via Bluetooth. The ground-truth annotations were made either through manual input on the N900 or by capturing simultaneous video and later annotating it.

For each run, the device pose and the user's perceived speed were recorded in the corresponding log file. Even though the N900 has its own accelerometer, we used the separate sensing device so that the annotation actions would not perturb natural walking motion.

We used data collected from previous experiments to train the system offline. The minimizer vector $\mathbf{c}^*$ in Equation 1 as well as the feature vectors from the training data were output to a file, which was then transferred onto the N900. Our program on the phone computes feature vectors based on the accelerometer signal obtained from the Sensorbox, and uses the training information in the file to estimate walking speed and device pose, according to Equation 1. For this implementation, we compute feature vectors based on 512-sample windows. At a 100 Hz sampling rate and with a 50% overlap between adjacent windows, our online C++ implementation produces speed and pose estimates approximately every 2.6 seconds.

EVALUATION

Methodology

We evaluated our algorithms under two scenarios: a controlled experiment evaluating different aspects of the algorithm, and an end-to-end evaluation with more natural walking indoors. We collected data from 14 participants total (of whom three participated in both experiments).

In the first experiment, we collected data from nine participants walking along a long corridor in a campus building. The participants consisted of five men and four women of varying ages and heights. Each participant was asked to carry a sensing device in one of four representative device poses — in hand (*hand*), at ear (*ear*), in trouser pocket (*pocket*), or in a backpack (*bag*) — and walk along the corridor with three different speeds — *slow*, *medium*, or *fast*, as perceived by the walker. Therefore, there were twelve combinations of device poses and (subjective) walking speeds. We did not ask the participants to carry the device in a certain orientation or with a certain hand, and did not attempt to guide their walking speed during the data collection process. That is, we attempted to capture data from each participant's natural walking motion. We collected 2853 data samples (121.7 person·minutes) in total.

To measure ground-truth speeds for the first experiment, we relied on colored tiles which occurred every 18 feet in our test corridor. Whenever the participant stepped on a colored tile, an experimenter following the participant recorded a timestamp by pressing a button on the N900 he or she was holding. Because the sensor data was received wirelessly, the data and ground-truth annotations were recorded without interfering with the participant.

The second set of experiments was designed to evaluate our walking speed estimation algorithm in indoor environments under a more natural setting. The target scenario for this experiment was the use of a mobile handheld device with an inertial measurement unit (IMU) for indoor positioning and navigation, where good walking speed estimation is expected to improve positioning accuracy.

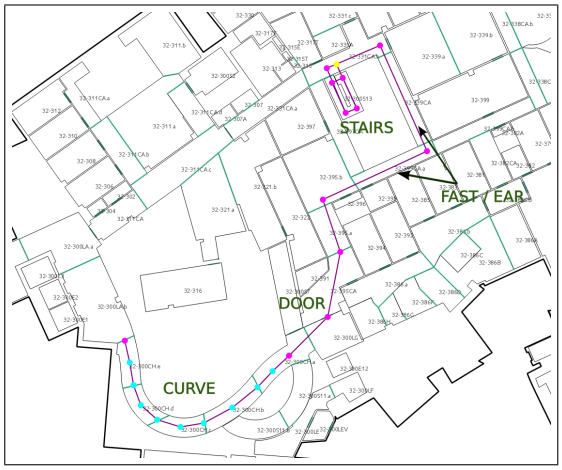


Figure 3. Path for the indoor walking experiment

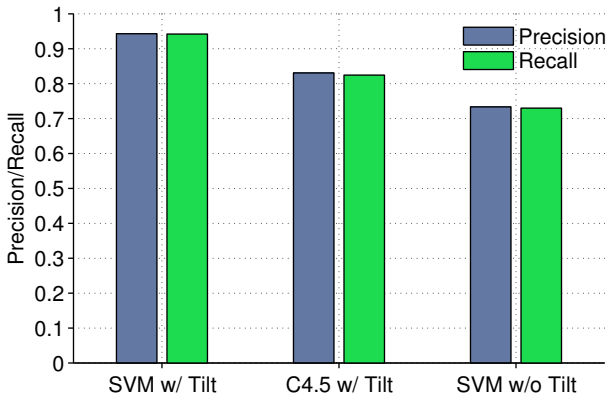


Figure 4. Generalization performance of pose classifiers to new users. The regularized kernel method with proper features (`SVM w/ Tilt`) outperforms other combinations for new users.

In the experiment, we had eight participants walk naturally along a choreographed path consisting of gentle curves, doors, turns, stairs, and "special" intervals (Figure 3). Each participant walked two round trips on the path, was asked to walk faster on the special interval on the first trip, and to put the Sensorbox to the ear as if he/she was making a phone call during the second trip. The experimenter following the participant captured video of the participant's walking, which was later used for ground-truth annotation. The experiment took about 7-10 minutes per participant.

Device Pose Classification

Here we describe the performance of our pose classification method. For the training of the SVM classifier, we used Shogun [19], a machine learning toolbox that provides a MATLAB interface.

Cross-Validation Test

We tested the performance of the algorithm by 10-fold cross validation including all participants from the first experiment. By using DFT and gravity tilt features together, we were able to obtain near-perfect overall accuracy of 99.6% in cross-validation. When only the DFT features were used, the classification accuracy was about 82% (confusion matrix not shown), among which a large portion of confusion was between *hand* and *ear*, whose oscillatory patterns are relatively similar compared to the other poses. However, as they have clearly different tilt angles (Figure 1c), the inclusion of the gravity tilt feature greatly improves accuracy.

Generalization to New Participants

We evaluated the algorithm using a leave-one-participant-out approach. For each user, we held out data from that user and trained the algorithm on data from all other users. Then, the algorithm was evaluated on the data from the held-out user.

Figure 4 shows the average precision/recall using the leave-one-participant-out evaluation method. The addition of the gravity tilt feature (Equation 9) substantially increases the ability of the algorithm to generalize to a new user. Table 1 shows the confusion matrix of the classification results.

		Predicted				
		Bag	Ear	Hand	Pocket	Total
Actual	Bag	701(92.6%)	0	0	56(7.4%)	757
	Ear	13(1.8%)	672(94.5%)	0	26(3.7%)	711
	Hand	15(2.0%)	0	730(97.9%)	1(0.1%)	746
	Pocket	13(1.7%)	50(6.4%)	0	714(91.9%)	777
	Total	742	722	730	797	2991

Table 1. Confusion matrix of pose classification - leave-one-participant-out validation

We also compare the performance of SVMs to that of the C4.5 decision tree algorithm, another classification algorithm that was shown to have good device classification results by Kunze *et al.* [11]. When the same set of features are used, our regularized kernel method shows better generalization capability, as it captures the similarity between the training and test data more effectively. As in [11], we use WEKA [7] for the decision tree implementation.

Online System Evaluation

Figure 5 shows the behavior of online pose classification in a test run. The participant followed a choreographed device pose pattern while walking. Overall, the pose classifier predicted true device pose well. As the classifier was running while the participant changed the device pose, the algorithm sometimes misclassified instances near pose transitions. However, the misclassification did not propagate because the classifier uses only the most recent 512 samples.

We also measured the running time of the online pose classification on an N900. The average running time for one run was 10.46 ms (std. err. = 1.14 ms). Considering that estimation occurs at every 256 samples (2.56 sec.), the computation time is negligible. While both algorithms require a large amount of training data, which can be done offline, the

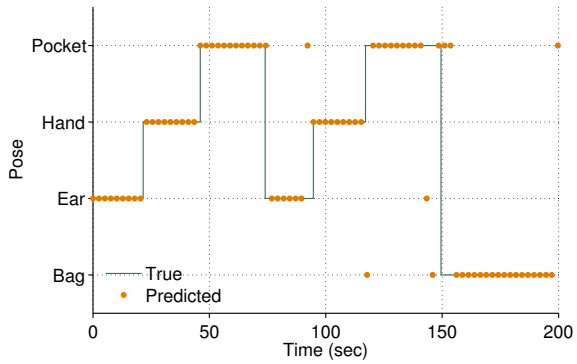


Figure 5. True and predicted device pose online. Pose classifier predicts true pose accurately, with a few misclassifications near pose transitions.

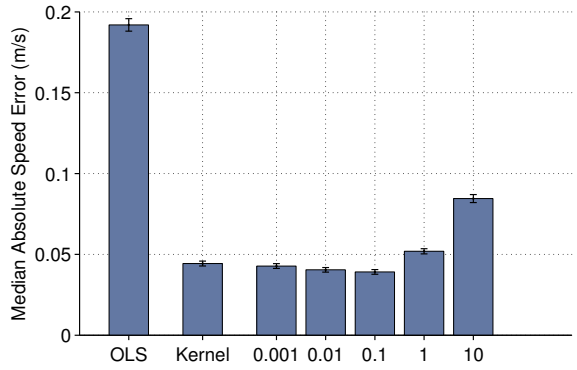


Figure 6. Comparison of ordinary least squares (OLS) with regularized least squares. OLS is compared to the kernelized least squares with varying regularization parameters $\lambda = 0.001$ to 10. Regularized least squares gives the best performance when the regularization parameter is suitably chosen (0.1).

prediction rule in Equation 1 for the online phase is relatively simple, making the algorithm viable for online use.

Speed Estimation

In this section, we show the performance of our speed estimation algorithm under various settings. We first study the role of regularization and kernels in speed estimation, comparing it against a simpler linear regression model. Then, we show that the performance improvement by knowing the device pose is minimal with our algorithm. We evaluated the algorithm's performance for new users by comparing it to previous work. Finally, we present results from an indoor walking scenario, assessing the comparative advantages and limitations of our walking speed estimation method.

Regularization and Kernels

Here we highlight the benefit of kernels and regularization. We compared the RLS algorithm to the ordinary linear least squares method, which does not take advantage of kernels and regularization. We used MATLAB's `regress` function to perform ordinary linear regression.

Figure 6 shows the median absolute speed error from a 10-fold cross-validation test. The inclusion of the kernel (Equation (12)), greatly improves speed prediction, giving a median error under 0.05 m/s. The inclusion of regularization with a suitable choice of the regularization parameter further improves speed estimation performance. The median absolute speed error achieved was 0.039 m/s, which amounts to 3% of the average normal walking speed (1.26 m/s) of the participants.

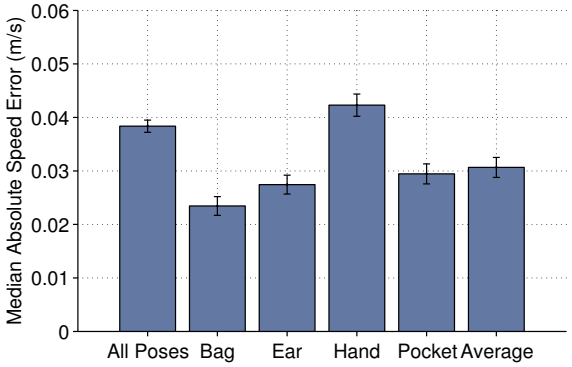


Figure 7. Comparison of pose-independent vs. pose-specific training for speed estimation. The rightmost bar is the average speed error of the four pose-specific training results.

Pose-Independent Training vs Pose-Specific Training

Our intuition was that knowledge of device pose would improve speed estimation. We also wondered how much improvement pose-specific training would yield.

We studied two cases to resolve these questions. In the first case, we trained the algorithm using all available data in the training set. In the second case, we chose training data from each specific device pose only and evaluated each pose separately. As shown in Figure 7, prior knowledge of the pose indeed improves speed estimation performance, but the improvement is very small: less than 0.01 m/s on average, amounting to less than 0.6 meters in distance for each minute of walking. This improvement is achievable only when the device pose can be determined accurately in advance.

Accumulated Distance Error

One of the primary applications of speed estimation using mobile devices is indoor navigation. We tested the applicability of our speed estimation algorithm to indoor navigation by examining a single continuous user trajectory and observing end-to-end behavior. Figure 8 shows the result obtained by accumulating speed estimates from two combinations of user, device pose, and perceived speed. The predicted distance lies within 7-9% of ground truth over a duration of 100 seconds.

Generalization to New Participants

We tested the algorithm's ability to predict the walking speed of new users whose data were not used for training. We compared our walking speed estimator to four previous methods for walking speed estimation. Every method compared assumes a different linear model between user's stride length and acceleration signal characteristics:

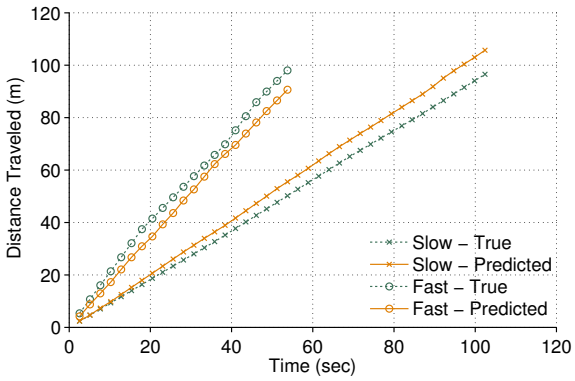


Figure 8. True and predicted distances of one participant's trajectories. The participant carried the Sensorbox in a bag while walking slowly or fast. For each data window, the walking speed was predicted and the corresponding distance was accumulated. The distance errors at the end of the trajectory were 7.4 m (fast) and 9.2 m (slow).

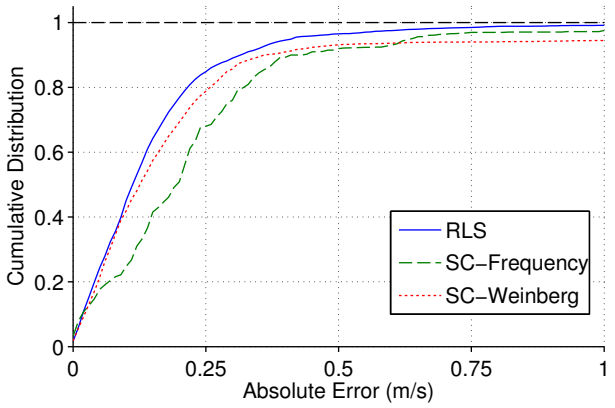


Figure 9. Leave-one-participant-out speed estimation error. The cumulative distribution of our algorithm (blue, solid) and other step-counting methods (dashed) are shown.

1. k_1 (constant stride length)
2. $k_2 \cdot h$ (proportional to user height)
3. $k_3 \cdot f_s$ (proportional to step frequency [4])
4. $k_4 \cdot \sqrt[4]{a_{max} - a_{min}}$ (proportional to 4th root of acceleration range [21])

The linear coefficients for all methods were computed from the same training data as our method. Due to limited space, we report comparisons only with AutoGait [4] and Weinberg's method [21], the two best-performing among the four methods compared.

To evaluate each algorithm's generalization performance on new users, we compared both algorithms to our RLS-based method using a leave-one-participant-out method, as in our evaluation of the device pose classification. The resulting cumulative error distribution is shown in Figure 9. The resulting median speed error for the RLS method was about 0.154 m/s (12.2% of the average normal walking speed), which outperforms both step-counting methods. In addition, our algorithm has fewer extreme errors (0.87% samples with $>$ 1 m/s error) than the others (5.55% for Weinberg and 2.67% for AutoGait).

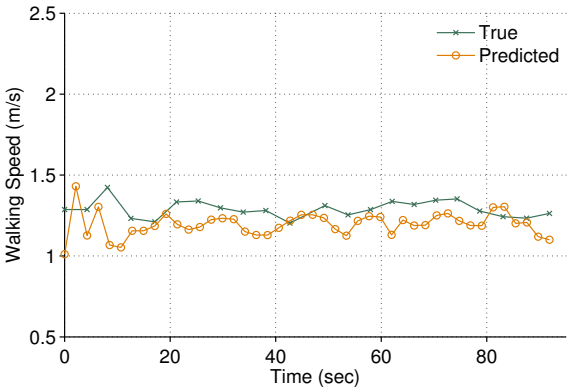


Figure 10. True and predicted walking speed online. Our online estimator predicts walking speed consistently with average error just under 0.1 m/s.

Online System Evaluation

Similarly to the online pose classification, we tested our online speed estimator on an N900 while a participant was walking along a straight corridor. The online walking speed algorithm estimates the true speed consistently, while slightly underestimating it by 0.098 m/s (Figure 10). From the consistent underestimation of the speed, we may infer that the participant walked with lower walking intensity compared to the average of all participants. The online walking speed estimator was running concurrently with the pose classification algorithm, and the average running time was 9.37 ms (std. err. = 2.4 ms), which was similar to that of online pose classification.

Indoor Walking Experiment

As described earlier, we performed an end-to-end test of the speed estimation algorithm for an indoor walking scenario. In this scenario, each user experienced various types of movements other than walking straight at normal speed, including walking on a gently curved path, turning a corner, opening a door, walking faster, changing phone pose to/from the ear, and walking up/down stairs (Figure 3). We used 256-sample windows for each data point to accommodate rapid motion changes in the designed experiment path.

Figure 11 compares the performance of our method against Weinberg's method, the best-performing one among the algorithms compared in the previous section The result was computed by the leave-one-participant-out method, similarly to the previous evaluations.

In general, our method predicts walking speed better than Weinberg's method, having comparable prediction error on curves or fast-walk interval, while performing better (with statistical significance) when the participant is holding device on ear, or walking on stairs. In particular, Weinberg's method tended to overestimate walking speed on stairs, because walking up/down stairs has higher acceleration range

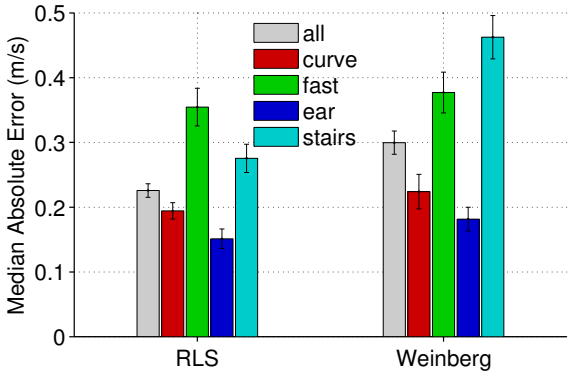


Figure 11. Walking speed estimation performance on a variety of motion segments in the indoor walking experiment.

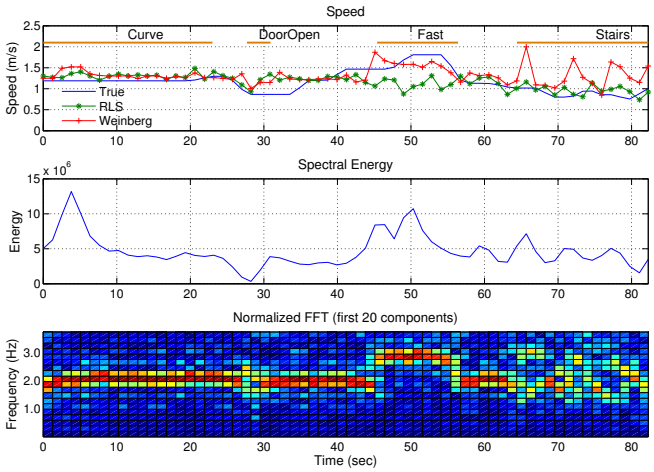


Figure 12. An example of an indoor walking experiment from a particular participant for which our walking speed estimator performed poorly. The first subplot shows true and estimated speeds. The second and third plot show spectral energy feature and FFT spectrum (the first 20 components). Our algorithm performed worse than the compared method for the fast walk interval.

compared to walking on a flat surface, while the actual strides were confined to the step length of the stairs. Therefore, Weinberg's method, which assumes that the stride length is proportional to the acceleration range, consistently overestimated the walking speed. Our method makes better use of additional information from the FFT spectrum, whose shape is different when walking on stairs (See right corner of FFT spectrum in Figure 12).

However, this experiment reveals some limitations of our current method. Figure 12 demonstrates one instance where our method performed poorly compared to Weinberg's, along the "fast" interval (from 45 to 56 sec.). In this case, the participant was "running" at around 3 Hz, which is considerably higher than normal walking frequency, and in fact higher than most of walking frequencies in our training data set. (We instructed participants to walk faster, rather than to run.) That is, that specific motion segment from this particular participant was an *unseen pattern* that machine-learning based approach may have difficulties with.

We observed one other condition in which our method (as well as Weinberg's) performed poorly: when the participant was swinging the device while walking. Swinging involves higher energy motion (Equation 11) than the pose conditions we originally considered for the experiment. This makes the algorithm predict a walking speed that is considerably higher than the true speed.

DISCUSSION AND LIMITATIONS

Evaluation of our methods with different criteria show that regularized kernel methods combined with suitable features and kernels can accurately classify device pose and estimate walking speed. We also demonstrated the viability of these algorithms in an online prediction system, which can serve as a foundation for higher-level context-aware applications. In particular, the results showed good performance on new users whose data were not used for training.

We tackled the problems of device pose classification and walking speed estimation independently after investigating the potential benefit of knowing device pose beforehand in walking speed prediction. Indeed, the resulting improvement was marginal even when perfect knowledge of device pose was assumed. This observation confirms the findings of Reddy *et al.* [17], where it is shown that the accuracy of pose-independent transport mode classifier using mobile phone sensors decreases only by 1.1 %. Furthermore, there is always a risk of pose misclassification, which may lead to degraded performance in subsequent speed estimation.

One explanation of why pose-specific did not outperform pose-independent classifiers could be the number of training samples, because, for k poses, each pose-specific classifier was able to use only about $1/k$ of the training data that the all-pose classifier had. That is, the number of training examples compensates for pose-specific difference in walking speed estimation. Therefore, we may attempt a *transfer learning* approach to train four pose-specific classifiers while simultaneously sharing knowledge among them. This is a direction for future work.

Our pose classifier cannot effectively handle the case when the user is not moving, because it utilizes frequency spectrum vectors as an important source of information. Even though the gravity tilt vector can provide some information about the device pose when the user is not moving, there are limitations on how much information we can obtain from the accelerometer found in a typical mobile phone. We expect that the pose classifier can be augmented with other sensing modalities, including light, acoustic, proximity, and touch sensors, to improve accuracy when the user is at rest.

Another limitation that our methods have as a machine learning application is that they may not perform well on an unseen pattern. If the user's motion behavior deviates too much from those in training, for example, the walking speed is far outside of the trained speed range, or the user performs unexpected motion with the device, the algorithm's reliance only

on training data can result in inaccurate output. On the other hand, model-based approaches such as step-counting may result in better predictions as long as the model assumptions hold. Therefore, the generalization ability that our methods exhibit should be understood as an improved ability to detect observed patterns in the presence of significant noise, rather than an ability to interpret previously unseen patterns.

This is again related to our limited sample size, which is a limitation of our experimental methodology. Even though we tried to capture data from participants with diverse demographics, the sample size used for the evaluation, 17 sequences from 14 different participants, was limited due to practical constraints. In particular, one of the biggest difficulties we faced was to obtain precisely annotated data without hampering participants' walking motion in real walking experiments, not on treadmills. Despite these limitations of our presented results, we expect that the algorithm would perform better as it learns from a more diverse population. We anticipate future work with extensive real-world experiments in more natural settings. Our datasets will be made public at `http://rvsn.csail.mit.edu/location`.

CONCLUSION

This paper describes methods for classifying device pose and estimating walking speed using standard mobile devices, such as phones and tablets. We do not assume absolute knowledge of the location of the sensors on the body, but instead analyze a set of a few typical device poses. We use time-series acceleration signals from a single triaxial accelerometer and extract a few sets of features, including spectral magnitudes, gravity tilt features, and spectral energy, which are suitable for each task. We apply support vector machines for pose classification, and regularized least squares for speed estimation. By incorporating appropriate features and kernels, our methods achieve high performance in a series of evaluations, and show good generalization to new users. We also discussed the limitations and possible extensions of the presented approach.

In future work, we plan to integrate our speed and device pose classification system within an indoor navigation system. In addition, even though our online system requires minimal computation in the prediction phase, we plan to investigate if further energy saving is achievable. For example, by integrating our algorithms with a simpler predictor that determines whether the user is walking without computing full FFTs, we can run our system adaptively, computing results only when the user is believed to be walking.

REFERENCES

1. S. Ayub, X. Zhou, S. Honary, A. Bahraminasab, and B. Honary. Sensor Placement Modes for Smartphone Based Pedestrian Dead Reckoning. In *Proceedings of the CICA 2011*, pages 123–132, 2011.
2. L. Bao and S. S. Intille. Activity Recognition from User-Annotated Acceleration Data. In *Proc. PERVASIVE*, 2004.
3. C. Bishop. *Pattern Recognition and Machine Learning*, volume 4. Springer, 2006.
4. D.-K. Cho, M. Mun, U. Lee, W. J. Kaiser, and M. Gerla. AutoGait: A Mobile Platform that Accurately Estimates the Distance Walked. In *PerCom*, pages 116–124, 2010.
5. T. Evgeniou, M. Pontil, and T. Poggio. Regularization Networks and Support Vector Machines. *Advances in Computational Mathematics*, 13(1):1–50, 2000.
6. E. Foxlin. Pedestrian tracking with shoe-mounted inertial sensors. *IEEE Computer Graphics and Applications*, 25(6):38–46, 2005.
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, 2009.
8. R. Herren, A. Sparti, K. Aminian, and Y. Schutz. The Prediction of Speed and Incline in Outdoor Running in Humans using Accelerometry. *Medicine & Science in Sports & Exercise*, 31(7):1053, 1999.
9. Y. Kawahara, H. Kurasawa, and H. Morikawa. Recognizing User Context Using Mobile Handsets with Acceleration Sensors. In *Proc. IEEE PORTABLE*, 2007.
10. A. M. Khan, Y.-K. Lee, S. Lee, and T.-S. Kim. Accelerometer's Position Independent Physical Activity Recognition System for Long-term Activity Monitoring in the Elderly. *Medical and Biological Engineering and Computing*, pages 1–9, 2010.
11. K. S. Kunze, P. Lukowicz, H. Junker, and G. Tröster. Where am I: Recognizing On-body Positions of Wearable Sensors. In *Proc. LoCA*, pages 264–275, 2005.
12. J. Lester, T. Choudhury, and G. Borriello. A Practical Approach to Recognizing Physical Activities. *Pervasive Computing*, pages 1–16, 2006.
13. E. Miluzzo, M. Papandrea, N. D. Lane, H. Lu, and A. Campbell. Pocket, Bag, Hand, etc. – Automatically Detecting Phone Context through Discovery. In *Proc. PhoneSense*, 2010.
14. J. Park, B. Charrow, D. Curtis, J. Battat, E. Minkov, J. Hicks, S. Teller, and J. Ledlie. Growing an Organic Indoor Location System. In *International Conference on Mobile Systems, Applications, and Services (MobiSys)*, San Francisco, CA, June 2010.
15. C. Randell, C. Djiallis, and H. Muller. Personal position measurement using dead reckoning. In *Proc. ISWC*, volume 1530, pages 166–175, 2003.
16. N. Ravi, N. Dandekar, P. Mysore, and M. L. Littman. Activity Recognition from Accelerometer Data. In *Proc. IAAI*, volume 20, page 1541, 2005.
17. S. Reddy, M. Mun, J. Burke, D. Estrin, M. Hansen, and M. Srivastava. Using mobile phones to determine transportation modes. *ACM Transactions on Sensor Networks (TOSN)*, 6(2):13, 2010.
18. B. Schölkopf and A. Smola. *Learning with Kernels*. MIT Press, 2002.
19. S. Sonnenburg, G. Raetsch, S. Henschel, C. Widmer, J. Behr, A. Zien, F. de Bona, A. Binder, C. Gehl, , and V. Franc. The SHOGUN Machine Learning Toolbox. *Journal of Machine Learning Research*, 11:1799–1802, June 2010.
20. H. Vathsangam, A. Emken, D. Spruijt-Metz, and G. S. Sukhatme. Toward Free-Living Walking Speed Estimation Using Gaussian Process-based Regression with On-Body Accelerometers and Gyroscopes. In *Proc. PervasiveHealth*, 2010.
21. H. Weinberg. Using the ADXL202 in Pedometer and Personal Navigation Applications. Technical Report AN-602, Analog Devices, 2002.
22. J. Yang. Toward Physical Activity Diary: Motion Recognition Using Simple Acceleration Features with Mobile Phones. In *Int'l Workshop on Interactive Multimedia for Consumer Electronics*, 2009.