Combining Acceleration and Gyroscope Data for Motion Gesture Recognition using Classifiers with Dimensionality Constraints

Sven Kratz
FX Palo Alto Laboratory
3174 Porter Drive, Palo Alto, CA, 94304, USA
kratz@fxpal.com

Michael Rohs
University of Hannover
Appelstraße 9A, 30167 Hannover, Germany
michael.rohs@hci.uni-hannover.de

Georg Essl
University of Michigan
Electrical Engineering and Computer Science
gessl@eecs.umich.edu

ABSTRACT
Motivated by the addition of gyroscopes to a large number of new smart phones, we study the effects of combining accelerometer and gyroscope data on the recognition rate of motion gesture recognizers with dimensionality constraints. Using a large data set of motion gestures we analyze results for the following algorithms: Protractor3D, Dynamic Time Warping (DTW) and Regularized Logistic Regression (LR). We chose to study these algorithms because they are relatively easy to implement, thus well suited for rapid prototyping or early deployment during prototyping stages. For use in our analysis, we contribute a method to extend Protractor3D to work with the 6D data obtained by combining accelerometer and gyroscope data. Our results show that combining accelerometer and gyroscope data is beneficial also for algorithms with dimensionality constraints and improves the gesture recognition rate on our data set by up to 4%.

Author Keywords
Motion Gestures; Mobile; Gesture Recognition; Accelerometer; Gyroscope; Sensor Fusion

ACM Classification Keywords
H.5.2

General Terms
Algorithms, Experimentation, Performance, Measurement, Human Factors

INTRODUCTION
A growing number of smart phones are being equipped with 3-axis gyroscopes in addition to 3-axis acceleration sensors. Combining the data from these two sensor types provides significantly more motion information compared to only using an accelerometer. On devices without a gyroscope, rotation can be approximated using accelerometers alone by using the direction of gravity as a reference for tilt. However, this approximation is not reliable in certain cases, i.e. when the device is rotated in the plane perpendicular to gravity, no tilt and thus rotation information can be obtained. The increase in motion information provided by gyroscopes can increase the recognition accuracy of motion gestures, which has been shown by previous work [4]. The higher recognition accuracy allows more complex gesture types to be used in mobile user interfaces, since the users can also add rotational components to their gesture inputs. Motion gestures can be used for a variety of applications on smart phones. These include gaming interfaces, where, for instance, the user interacts via a spell casting metaphor [1], entering special UI modes [11], UI navigation tasks [12] or user authentication [3].

It is desirable to have gesture recognizers for motion gestures on mobile devices that can be used early in the development of an application. This requires such algorithms be easy to implement and tune. Furthermore, for development on mobile devices in particular, it is beneficial for gesture recognition algorithms to be independent of specialized libraries or toolkits, in order to be able to deploy those algorithms on as many devices as possible.

Template-based techniques such as nearest-neighbor search are generally easy to implement and address most of the concerns mentioned previously. However, naïve template-based techniques will generally not compensate for variations in gesture execution time. A popular algorithm that does compensate for differences in time series is Dynamic Time Warping (DTW) [13]. Protractor3D [5] is a template-based technique developed for 3D acceleration data that compensates for rotational derivations between input sequences and templates by finding the optimal registration between input points and templates, in a way similar to Protractor [7], which only works with 2D data. A problem with Protractor3D is that this algorithm does not work with data dimensionalities greater than than 3.

In this paper we present a consensus-based approach using two instances of a template-based gesture recognizer (such as Protractor3D or DTW) to perform motion gesture recognition on the 6-D data obtained from an accelerometer-gyroscope pair. By analyzing a large corpus of motion gesture entries by users, we show that the combination of accelerometer and gyroscope data increases the gesture recognition rate by up to 4% on our data set. Our results indicate that combining acceleration data with rotation data from a 3-axis gyroscope will improve the gesture recognition rates for motion gestures entered on a mobile device. Reflecting on the results we obtained, we provide recommendations for the choice of a motion gesture recognizers for mobile applications.

RELATED WORK

DoubleFlip [11] addresses the problem of false positive gestures by employing a simple delimiter gesture (the "double flip" gesture) for entering gesture input mode. The criteria for choosing the "double flip" gesture were (1) the gesture being easy to perform and (2) the gesture being sufficiently distinct from everyday movement. To confirm the latter the authors investigated the false positive rate for a large corpus of everyday movement data.

Protractor [7] is a gesture recognition algorithm for touch screens. It is a template-based recognizer that resamples the gesture trace to get a vector with a fixed dimension, which is then translated to the origin and normalized. For each comparison with a template a rotation is performed that optimally aligns the input and template gestures. This is done in an efficient way using a closed-form analytic approach. *Protractor3D* [5] extends this idea to three dimensions. The approach is similar in principle to Protractor. However, finding a closed-form analytic solution in 3D is much more complicated. Compared to DTW, neither Protractor, nor Protracor3D do any kind of warping in the time domain.

Hoffman et al. [4] combined accelerometer and gyroscope data for gestural input. The addition of a gyrosope significantly increased their gesture recognition results. They used linear and AdaBoost [14] classifiers. Both of these methods can cope with high-dimensional feature vectors. In this paper we propose a method to leverage the additional motion data provided by a gyroscope with data-driven classifiers limited to three feature dimensions, such as Protractor3D.

EXTENDING PROTRACTOR3D WITH GYROSCOPE DATA

By design, Protractor3D cannot use data of a higher dimensionality than 3. This is inherent in the mathematics it uses to calculate the optimal registration between input points and templates. A simple extension to add support for gyroscope data in addition to accelerometer data is to run a second instance of Protractor3D in parallel on the gyroscope data. The remaining challenge is then to reconcile the two recognition results in order to determine which gesture has been recognized.

We propose a weighted approach that depends on the order of similarity of comparisons between input gestures and stored templates, i.e., templates that are less similar to the current input have a lower influence on the final recognition result. Our results indicate that the match with the lowest distance (or highest similarity) is most probably the gesture recognized. In other words, comparisons with a lower similarity are less likely to contribute to the decision on the recognition result.

Figure 1 shows pseudocode for the data combination algorithm (DCA) we devised. The algorithm returns the ID of the best matching gesture. The *bias* variable determines the influence of comparison results according to their rank. Results with a low rank (i.e., lower distance to the input gesture) have a higher influence on the final matching result. The $argmax()$ function we used returns the index corresponding to the first occurrence if there are multiple occurrences of the same maximum value.

The DCA can be generalized easily to combine the results for more than two sensor data types. The only parameter of the algorithm that needs to be chosen specifically by the developer is *bias*. The optimal value for *bias* needs to be determined by analyzing existing user inputs. Figure 2 shows the influence of *bias* on *Precision* and *Recall*[1] results for our data set, where $0.8 \leq bias < 1.0$ provides the best result.

```
Inputs:
accResults: acceleration comparison results (list of gesture IDs sorted in
descending order by similarity to the input)
gyrResults: acceleration comparison results (list of gesture IDs sorted in
descending order by similarity to the input)
N: number of items in each of the comparison lists
bias: bias value determining influence of weaker comparisons
nGestureIDs: number of gesture classes (each class has a distinct ID)

counter = float[nGestureIDs]
for i=0 to N-1 do
    accRecogID = accRresults[i]
    gyrRecogID = gyrResults[i]
    counter[accRecogID] += 1.0/(bias*i +1.0)
    counter[gyrRecogID] += 1.0/(bias*i +1.0)
end for
bestGestureID = argmax(counter)
return  bestGestureID
```

Figure 1. Data Combination Algorithm (DCA): pseudocode for a weighted approach to combine recognition results for accelerometer and gyroscope readings.

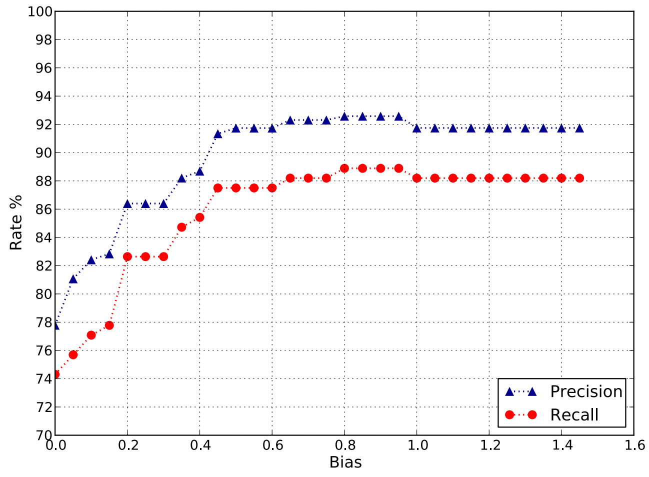

Figure 2. The influence of the bias variable in the weighted reconciliation algorithm. In this case $N = 5$. Choosing a $bias > 0.8$ does not further improve the gesture recognition rate, while giving each result an equal "vote" ($bias = 0$) generally results in a lower gesture recognition rate.

[1]for the definition of *Precision* and *Recall*, see Page 3, "Classifier Performance Metrics"

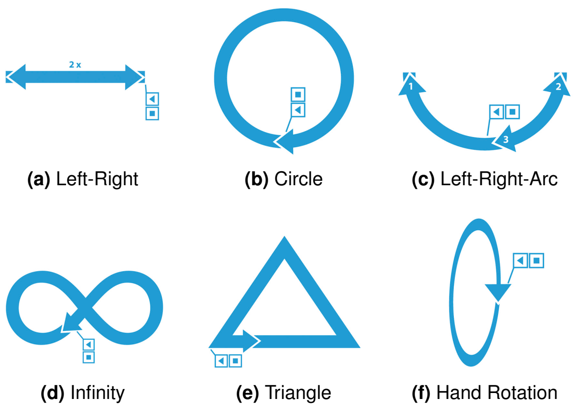

Figure 3. Visualization of the gesture classes we designed for input by the users.

COMBINING ACCELEROMETER AND GYROSCOPE DATA

To analyze the effects of combining 3-axis acceleration data with 3-axis rotation data for recognition by motion gesture recognizers, we present gesture recognition results for the following recognition algorithms: *Protractor3D*, *DTW* and *Regularized Logistic Regression (LR)* [6]. We chose DTW due to its popularity. We chose LR because it is, in contrast to DTW and Protractor3D, a feature-based machine learning algorithm that performs well with higher-dimensional data and it is still fairly easy to implement. LR is very efficient, since classification basically consists of multiplying a weight vector θ of a fixed length N (where N corresponds to the amount of features) with the features of the gesture input. Thus, compared with Protractor3D and DTW, the execution time of LR is independent of the number of training samples used.

Gesture Data Set

The gesture data for our analysis was gathered from 6 female and 9 male participants. All participants were right-handed, students and aged from 20 to 32 ($\mu = 24.3$, $\sigma = 2.9$). We compensated all participants with a small sum of money.

We designed six different gestures for the users to enter, as shown in Figure 3. Each participant provided a total of 15 samples for each gesture. Gestures were delimited using a *push-to-gesture* button. Recording of sensor data started when the button was pressed and ceased when the button was released. For all further analysis, we use the first 5 entries of each gesture by each user as (per-user) training templates and the corresponding last 10 entries for validation (also per-user). Our motivation for using the first 5 entries as training templates is because this reflects a realistic usage scenario for an mobile applications using personalized motion gestures. It is plausible to assume that new users would be required to train the system upon first use of the application.

In total, we recorded $15 \times 15 \times 6 = 1350$ gesture entries. 20 gesture entries had to be discarded, because they had too few samples due to erroneous entry. Our final data set size comprises 1330 gestures. Each gesture entry contains an average of 127 time-based data samples ($\sigma = 38.4$).

An iPhone 4 running a custom-built application, which provided logging and the push-to-gesture button interface, was used for recording gesture entries. We recorded the following data from the iPhone 4's accelerometer[2] and gyroscope[3] at a frequency of 80Hz: *acceleration*, *rotation rate* and *attitude*[4]. All acceleration values were measured in g, all rotation was measured in radians/second and attitude was given in Euler angles.

Recognition Results

We evaluated Protractor3D, DTW and LR for motion gesture recognition on the data set. As input to the algorithms, we used the following data and combinations thereof: *acc* (acceleration), *rot* (rotation), *att* (attitude), *acc-rot* (acceleration + rotation), *acc-att* (acceleration+attitude).

We configured Protractor3D to subsample and normalize the gesture entries to contain 64 samples. In order to keep the length of the feature representation for each gesture constant, we used the subsampled data generated by Protractor3D as input for LR. Each gesture was thus represented as a feature vector with a length of $64 \times 3 = 192$ for *acc, rot* and *att* and $64 \times 6 = 384$ for *acc-rot* and *acc-att*. For LR, we generated models for each gesture type for each user, and utilized per-user multi-class classification with a one-vs.-all strategy to classify the input gestures.

DTW used the unmodified sensor samples as input, as this algorithm is specifically designed to cope with length differences between inputs and templates. Our DTW implementation used the Euclidean (L^2) Norm as the distance function. We did not apply any step constraints for calculating the distance matrix.

Classifier Performance Metrics

As quantitative metrics for classifier performance, we use *Precision* (P), *Recall* (R) and F_1 score (F_1) [8, 9]. Given the amount of true positives tp, the amount of false positives fp and the amount of false negatives fn, Precision is defined as $P = tp/(tp + fp)$, i.e. the ratio of correct gesture recognitions out of all generated predictions. Recall is defined as $R = tp/(tp + fn)$, i.e. the ratio of correct gesture recognition out of all gestures in the data set. F_1 is defined as:

$$F_1 = 2\frac{PR}{P + R}$$

which represents a combined metric for the accuracy of a classifier and is based on the harmonic mean. The F_1 Score ranges from 0 to 1, where 1 is the best score obtainable.

Figure 4 shows the results for Precision (P), Recall (R) and F1-Score ($F1$) for the classifiers Protractor3D, DTW, and LR for each of the combinations of data types *acc* (acceleration), *rot* (rotation), *att* (attitude), *acc-rot* (acceleration + rotation), *acc-att* (acceleration+attitude). The combined data was evaluated using the DCA. In addition, we evaluated DTW and

[2]STMicroelectronics STM331DLH 3-axis MEMS accelerometer

[3]STMicroelectronics L3G4200D (equiv.) 3-axis MEMS gyroscope

[4]*attitude* represents the absolute change of rotation with respect to an initial reference frame, i.e. the device attitude at the beginning of a gesture recording.

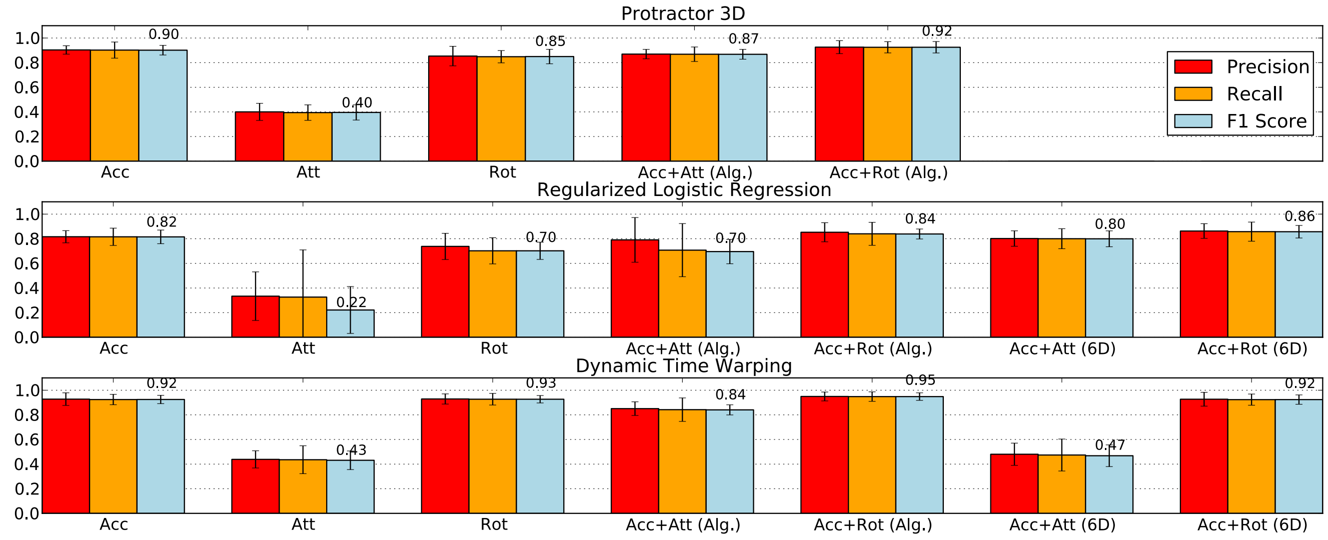

Figure 4. Mean recognition results for Protractor3D, Regularized Logistic Regression and DTW, for the data types accleration (*acc*), attitude (*att*), rotation (*rot*), acceleration+attitude *acc+att*), acceleration+rotation (*acc+rot*). "Alg." marks results the using the DCA. "6D" marks results using a 6D merged representation of accelerometer combined with gyroscope data. The error bars show the standard deviation. DTW with *acc+rot* using the combination algorithm achieved the best result with an F_1 Score of 0.95. The error bars show the standard deviation of the means.

LR without the DCA by using 6D feature vectors obtained by merging each 3D acceleration sample with its rotation counterpart. DTW with *acc-rot* and the DCA achieved the best result with an F_1 Score of 0.95, followed by Protractor3D using the same settings with an F_1 Score of 0.92. The best F_1 Score for LR was 0.86 using combined accelerometer and gyroscope data without the DCA. The lower score for LR suggests that a larger amount of training samples may be needed in order to obtain better results for this algorithm.

Statistical Analysis

We studied the effects of the *algorithm* (*P3D*, *DTW* and *LR*) and the *data type* (*acc*, *att*, *rot*, *acc+att* and *acc+rot*) on the mean F_1 score per gesture type. Not observing a normal distribution of the means, we ran Kruskal-Wallis tests for *algorithm* and *data type*, respectively. The results for *algorithm* ($H(2) = 9.46, p < 0.01$) and *data type* ($H(4) = 52.57, p < 0.001$) show significant effects. Non-parametric Dunn-Bonferroni [2] pairwise comparisons show a significant difference between *DTW* and *LR* ($Q = 20.45$, $p = 0.007$) for *algorithm*. Pairwise comparisons using the same method for *data type* unfortunately did not reveal significant effects between *acc* and *acc+rot* ($Q = -14.17$, $p = 0.1$), although we measured a 2%–4% increase in mean F_1 score depending on the algorithm used. The relatively low sample size $N = 18$ (mean values from 6 gestures $\times$ 3 algorithms) for each *data type* and the relatively small increase in F_1 score may not be sufficient to achieve $p < 0.05$ statistical significance in our case.

Algorithm Execution Times

For motion gesture recognizers, gesture recognition performance is not the only important criterion. The execution time is, arguably, equally important, especially on mobile platforms. The runtime of template-based approaches Protractor3D and DTW directly depends on the number of training templates, whereas this does not affect LR. To exemplify this for our data set, Protractor3D and DTW have to compare each of the 450 template gestures with 900 gesture entries, resulting in a total of 405,000 necessary comparisons for a single pass over the data set. For each comparison, DTW needs to construct a distance matrix with a size of $n \times m$, where n is the number of elements in the template and m of the sample, including evaluation of the distance function. A number of optimizations exist to improve the speed of DTW [10]. The results in this paper, however, reflect the performance of unoptimized DTW. Protractor3D does not have as much overhead as DTW, but still needs to complete the same number of comparison operations. By contrast, LR requires just 900 matrix multiplications to perform all predictions. Nevertheless, in comparison to DTW and Protractor3D, feature-based classifiers such as LR need a large number of training samples to work optimally, so developers face a trade-off between the time requested from users to enter training samples and the computational efficiency of the gesture recognizer.

Algorithm	**Approx. Number of required comparisons**	**Profiled Cumulative Execution Time (s)**	**Average Processing Time per Gesture (s)**
LR	900	0.4	0.004
Protractor3D	405,000	881	0.98
DTW	405,000	3966	4.4

Table 1. The approximate number of operations and the execution time (including data loading and pre-processing) required by the analyzed algorithms for a run on our data set. The average processing time per gesture was calculated by dividing the total execution time by the number of test samples in the training set. (Unoptimized) DTW is by far the slowest algorithm. LR has the lowest measured execution time as it performs only a single operation for each gesture in the validation set.

Table 1 shows the measured execution times for each algorithm when running on our data set. We implemented all gesture recognition algorithms in Python 2.7.1 using the *NumPy*, *SciPy* and *rpy2* libraries. All computation was performed on a Mac Pro with 2.66GHz Intel Core i7 processors. In order to measure the classifiers' execution time, we used Python's *cProfile* library. All classifiers were run exclusively as single-threaded applications and did not make use of parallel processing.

The comparison in Table 1 shows that the average per-gesture execution time of LR is two orders of magnitude lower than Protractor3D and three orders of magnitude lower than DTW. The runtime advantage of LR is thus very clear but the problem remains that this algorithm requires an optimization library in order to train models. Thus, the most beneficial use of LR would be in deployment-stage applications, with appropriate models for gesture recognition already trained, since the classification step does not require an optimization library. Protractor3D compares favorably with DTW, requiring about 4s less computation time. The measured processing time per gesture could be further improved by decreasing the subsampling size and switching from Python to a lower-level language such as C.

Summary and Discussion

Using a substantial data set, we analyzed the effect of adding the data of a gyroscope to accelerometer-based motion gesture recognition. In particular, we were interested in showing how adding additional gyroscope data affects dimensionally-constrained recognition algorithms, i.e. those that cannot process data of a dimensionality higher than 3. We devised a method to combine two or more dimensionally-constrained classifiers to perform motion gesture recognition for data dimensions greater than 3.

Confirming results obtained in previous work [4], our results indicate that combining accelerometer with gyroscope data leads to improvement in gesture recognition rates. Our key insight here is that also appears to work for simpler, template-based methods as well as dimensionally-constrained recognition algorithms, such as Protractor3D, using the data combination algorithm (DCA) we propose. Concretely, we observed 2%, 3%, and 4 % increases in F_1 score for Protractor3D, DTW and LR, respectively. Conversely, we observed that the proposed DCA does not improve the recognition rates of algorithms that can cope with higher-dimensional data, such as LR or DTW.

We recommend that developers choose their gesture recognition algorithm by (1) the number of training samples available (template-based approaches perform well, even with few training samples), (2) the performance constraints of their target devices (we obtained the highest F_1 score when using DTW, but it was also the algorithm with the highest execution time by far) and (3) the time available for coding during each iteration (template-based methods are easier to implement and deploy on different types of devices, since they do not require specialized libraries).

REFERENCES

1. Ballagas, R., Kratz, S.and Borchers, J., Yu, E., Walz, S., Fuhr, C., Hovestadt, L., and Tann, M. REXplorer: a mobile, pervasive spell-casting game for tourists. In *CHI'07 extended abstracts*, ACM (2007), 1929–1934.
2. Dunn, O. Multiple comparisons using rank sums. *Technometrics 6*, 3 (1964), 241–252.
3. Guse, D. Gesture-Based User Authentication on Mobile Devices using Accelerometer and Gyroscope (Master's Thesis). *Quality and Usability Group, Deutsche Telekom Laboratories, TU Berlin* (2011).
4. Hoffman, M., Varcholik, P., and LaViola, J. Breaking the status quo: Improving 3d gesture recognition with spatially convenient input devices. In *Virtual Reality Conference (VR), 2010 IEEE*, IEEE (2010), 59–66.
5. Kratz, S., and Rohs, M. Protractor3D : A Closed-Form Solution to Rotation-Invariant 3D Gestures. In *Proc. IUI 2011*, ACM (Palo Alto, CA, USA, 2011).
6. Lee, S., Lee, H., Abbeel, P., and Ng, A. Efficient L1 Regularized Logistic Regression. In *Proc. National Conference on Artificial Intelligence*, vol. 21, Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999 (2006), 401.
7. Li, Y. Protractor: a fast and accurate gesture recognizer. In *Proc. CHI 2010*, ACM (Atlanta, Georgia, USA, 2010), 2169–2172.
8. Makhoul, J., Kubala, F., Schwartz, R., Weischedel, R., et al. Performance measures for information extraction. In *Proceedings of DARPA Broadcast News Workshop* (1999), 249–252.
9. Powers, D. M. W. Evaluation: From precision, recall and f-measure to roc., informedness, markedness & correlation. *Journal of Machine Learning Technologies 2*, 1 (2011), 37–63.
10. Rakthanmanon, T., Campana, B., Mueen, A., Batista, G., Westover, B., Zhu, Q., Zakaria, J., and Keogh, E. Searching and mining trillions of time series subsequences under dynamic time warping. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM (2012), 262–270.
11. Ruiz, J., and Li, Y. DoubleFlip: a motion gesture delimiter for mobile interaction. In *Proc. CHI 2011*, ACM (2011), 2717–2720.
12. Ruiz, J., Li, Y., and Lank, E. User-defined motion gestures for mobile interaction. In *Proc. CHI 2011*, ACM (2011), 197–206.
13. Sakoe, H., and Chiba, S. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech and Signal Processing 26*, 1 (1978), 43–49.
14. Schapire, R. E. A brief introduction to boosting. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, IJCAI '99, Morgan Kaufmann Publishers Inc. (San Francisco, CA, USA, 1999), 1401–1406.