Ubiquitous Computing

2009 International Symposium, UCS

Beijing, China, August 24-26, 2009

ISBN 978-4-915256-77-6 C3040

Publisher :
Information Processing Society of Japan
Kagaku-kaikan (Chemistry Hall) 4F
1-5 Kanda-Surugadai, Chiyoda-ku, Tokyo 101-0062 JAPAN
Tel:+81-3-3518-8374 Fax:+81-3-3518-8375

Printed in Japan

International Symposium on Ubiquitous Computing Systems (UCS2009), pp. 43-50 (Aug. 2009)

A Method for Context Awareness using Peak Values of Sensors

Kazuya Murao
Osaka University, Japan
murao.kazuya@ist.osaka-u.ac.jp

Kristof van Laerhoven
TU Darmstadt, Germany
kristof@mis.tu-darmstadt.de

Tsutomu Terada
Kobe University, Japan
tsutomu@eedept.kobe-u.ac.jp

Shojiro Nishio
Osaka University, Japan
nishio@ist.osaka-u.ac.jp

***Abstract*—In wearable computing environments, various applications are assumed to get a richer sense of context via a set of wearable sensors. When obtaining the wearer's context, raw sensor values typically have to be pre-processed before recognition can take place. This process of feature-extraction in wearable sensing has thus far favored combinations of mean, variance, and Fourier coefficients over a sliding window as highly-discriminative features and have been used extensively so far in the literature. However, the computational cost of these features can be relatively high and the size of the features tends to become larger than that of the raw data itself, conflicting with the often low-power hardware in wearable computing. In this research we evaluate several features on their distinctive qualities via traditional boosting, cluster precision and support vector machine, as well as their cost of implementing them in wearable computing hardware. In addition, we suggest features that perform in the range of conventional features but that have lower costs to embed in hardware with limited resources.**

***Keywords*-Wearable computing; Wearable sensing; Features**

I. Introduction

The downsizing of computers and sensors in the last decades has made the paradigm of wearable computing increasingly feasible to implement with off-the-shelf components. Wearable computing systems differ from conventional computing in three significant ways [1]: (1) Hands-free operation: information can be obtained without manual operation because the computer is worn. (2) Power always on: the computer is always available because the power is always on. (3) Adaptation to personal: Complete service is provided according to user's detailed information obtained from sensors. Many context-aware systems with various kinds of sensors have been introduced in this field, such as systems with an electromyograph [2], electrocardiogram [3], Galvanic Skin Reflex (GSR) [4], and inertial sensors [5]. Power consumption is a major bottleneck in particular for robust deployment of wearable systems that are designed to capture sensor data continuously, and especially the efficient sensing and processing of sensed data are crucial. Advancing research here would also impact similar ubiquitous and pervasive computing research.

Context-aware systems in wearable computing are applied in a variety of services such as health care [4], recognition of workers' routine activity [6], and support of assembly and maintenance tasks [7]. A health-care system [4] for instance recognizes situations of life habits in real time using a heat sensor, GSR sensor, accelerometer, electric sphygmograph, GPS, geomagnetic sensor, and gyroscope. The wearable system recognizes contexts as they happen, and advises the user on how to make improvements in one's life to keep healthy.

Research in power-efficient detection mechanisms for wearable sensors has produced frameworks such as Groggy Wakeup [8] and individual sensor examples such as the Porcupine [5], which change sensor modalities and go into variable power modes depending on what has been sensed before. Moving the processing algorithms of the raw sensor data close to the sensor, i.e., *in* the sensor hardware, has in these types of research proven to result in significant reductions in power consumption, while maintaining the capacity to classify patterns in the sensor data.

A crucial step in any classification process from sensed signals is the mapping of the raw sensor values to pre-processed features that facilitate a faster and more accurate recognition. This has been done by calculating values such as the mean, variance, and Fourier coefficients over a sliding window, which have proven to be features that perform very well regarding classification accuracy, and have as a consequence been used extensively in systems seeking a degree of context awareness.

However, the computational cost of these features on wearable devices, which tend to be based on microcontroller architectures with limited processing and memory resources, is typically not considered in these evaluations. The impact of features' implementation details is not limited to the processing required to convert raw data to features. It is for example not uncommon to have the memory space of the produced features exceed that of the raw data itself, creating an overhead on storage and communication, and creating conflicts with low-power hardware requirements.

In this paper, we have set out to evaluate features on how suitable they are to distinguish contexts via traditional techniques, notably boosting and cluster precision, and support vector machine, which is a high performance classifier, and factor in the cost of their implementation in typical wearable computing environments. The purpose of this work is to create a framework in which newly suggested features can be compared to conventional ones, given a common dataset and assuming the need of them to be implemented on low-power hardware that needs to remain active and sensing while being

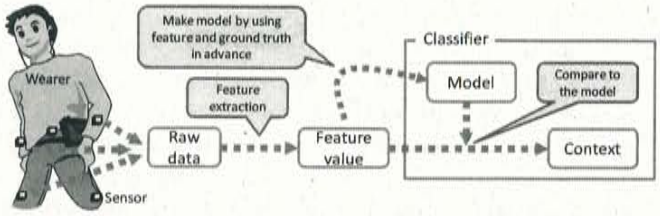

Figure 1. Our sensing platform consists of wearable inertial sensors, and follows a common structure of context awareness systems. Raw data is transformed into features, directly on the sensors, while the trained model and classification are assumed to be done in a central component of the wearable system.

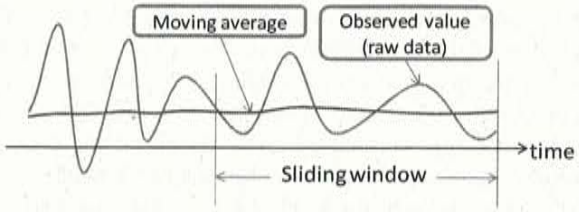

Figure 2. Illustration of the mean of a sensor signal calculated over a sliding window

worn.

II. Platform: wearable activity sensors

Figure 1 shows the separate processing stages in our specific wearable system: Inertial sensors are placed on the hip, wrists and legs of the user, measuring motion and posture-related signals. These raw data are pre-processed into features, after which they are passed on to a classification system to infer what activity the wearer of the system is performing. The target is to implement adequate features on the sensor nodes themselves, and thus enabling a more power-oriented context-aware system.

There are two ways this system could be envisioned: one is an on-line application that obtains the wearer's activity in real-time, with support systems for workers [6], [7] and user-situated information display systems [9] as just a few of many possible examples in this category. The other type of system assumes a purely off-line application, which collects wearer's sensor data while active and then analyzes the collected data, with exercise support systems [10], [11] as an example of this type of system. Implementing features on the sensor hardware produce the same effect for both approaches: in the on-line approach, the classifier needs to be implemented on a central wearable computer in constant communication with the sensor nodes, while in the off-line approach this can be done on a common computing system afterwards, but collected feature data still need to be stored locally.

The wearable sensors used in this paper are based on three-dimensional accelerometers embedded in a small-scale logging platform [5], which was designed to last for as long as possible on a light-weight rechargeable battery. A mini-USB port provides a means to configure and simultaneously recharge the device, and download the logged data to a host PC. The sampling frequency used in the remainder of this paper is 150 Hz, where raw accelerometer values are logged to the device's microSD memory card to serve afterwards as the data set for feature evaluation.

III. Features

Here we will cover a basic set of features and compare their performances using the previously discussed measures. Various other features may prove to perform better, but their inclusion is left as future work since this paper focuses on an appropriate method to evaluate features for usage in wearable sensing environments.

A. Mean

Mean is one of the features used in this field. This feature is calculated by taking the arithmetic mean of all sensor values over a sliding window of data. The arithmetic mean (from here on referred to as mean) in accelerometer data (see Figure 2) tends to smoothen deviations in the signal and extract average posture of the sensor. For static activities, such as sitting, lying down, or standing upright, this feature would extract therefore overall posture while neglecting any motion-related information. The implementation of the mean algorithm in a microcontroller environment usually restricts itself to keeping a sum variable over the window, and updating this variable with the first and last values of the window (i. e., the value going out on the next iteration, and the value coming in). Therefore, the time- and memory-complexity of this algorithm is constant and is extremely well-suited for implementation on limited sensor hardware.

B. Variance

Variance is often combined with the mean, and is computed by taking the variance statistic over a sliding window of sensor values, similar to the mean. In accelerometer data, variance expresses how much activity (motion, vibration, shaking, etc.) the sensor underwent, and would therefore be a distinctive feature to separate data between activities such as standing still, walking, and running. The implementation of the variance can be done efficiently by using the computational formula $Var(X) = E(X^2) - (E(X))^2$ with E the expected value or mean. A numerically more stable algorithm, however, is given by Knuth [12] which computes both mean and variance in one loop, with the aid of a delta variable and two accumulative variables for mean and variance. Although this algorithm has additional variables, multiplications and especially a division present in

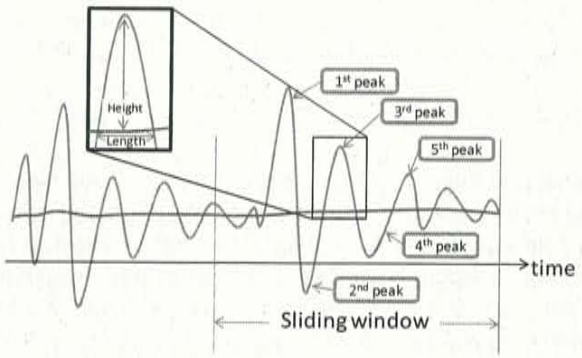

Figure 3. Illustration of peaks detection in a sensor signal, where the peak height and length in time are extracted to characterize the data.

the loop, it operates without problems in current low-power microcontroller environments.

C. Minimum and maximum

Minimum and maximum are features that keep the minimum and maximum values over a sliding window. In accelerometer data, these features would be capable to efficiently representing amplitude of impacts, such as taps, bumps, and knocks against the sensor. The implementation of these features requires comparisons against simple variables, and can be optimized by checking outgoing values in cases of large sliding windows. It is therefore particularly appropriate for resource-limited platforms.

D. Peak detection

Peak detection extracts basic information on the shapes that can be found in the signal, often using the points where the mean crosses the signal and extrema in the signal as descriptive values. In accelerometer data, these features are highly descriptive for activities that exhibit exactly the same patterns at all times, even though the posture or amount of motion might be extremely similar. Figure 3 illustrates the concept in a timeseries plot, where a sequence of 5 peaks is described by the peaks' heights (positive or negative) and lengths in time. The basic implementation of peak extraction using the mean as a baseline, in essence combines the algorithms for finding the mean, minimum and maximum, and requires just a few extra variables. It is therefore in the same range of time-and memory complexity as the previously described features, and slightly lower than the algorithm producing both mean and variance.

IV. Analysis methods

This section lists the methods that were used in this paper's study to evaluate features on their qualities for their distinctiveness towards use in a classification process and their cost of implementing in low-power sensing hardware.

A. Ranking Features with Cluster Precision

Clustering is a method to clarify a structure in a data set by grouping the data points together according to a particular distance metric. Cluster precision (see for instance [13]) is a simple measurement of how discriminative a data set is, knowing the classes the data samples belong to. By using this metric, a feature-processed data sets can be ranked against others according to the quality of the resulting clustering (commonly done by K-means clustering).

Ideally, each cluster would contain samples of only one activity. This would indicate that the feature was clearly separable and thus well-suited as an input for classification. In the worst case, the fraction of samples of an activity in each cluster would be equal to the a priori probability of the activity. This would imply that the feature was not discriminative for the given set of activities and thus unlikely to be suited for recognition.

In order to measure the distribution of samples for different activities in the cluster, we first computed for each cluster i and activity j the fraction

$$p_{i,j} = \frac{|C_{i,j}|}{\sum_j |C_{i,j}|} \tag{1}$$

where $C_{i,j}$ is the set of samples in cluster i labeled with activity j. We then formed a weighted sum of these fractions to obtain a cluster precision p_j for each activity j.

$$p_j = \frac{|\sum_i p_{i,j} C_{i,j}|}{\sum_i |C_{i,j}|} \tag{2}$$

Thus, if an activity has a cluster precision close to one, this means that there are many clusters mainly consisting of samples for this activity. We weight each fraction by the number of samples it represents in order to prevent smaller clusters from dominating the result.

B. Ranking of Features with Adaboost

By summing up weights of multiple weak classifiers, strong classifier can be formed according to the result, and this method is called Boosting. Adaboost [14] is an algorithm which determines the weak classifier and its weight successively. The algorithm of Adaboost is shown in Figure 4. The boosting algorithm takes as input a training set of N examples $(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_N, y_N)$, where $\mathbf{x}_i$ is an explaining variable vector, in this paper this is sensing data, and $y_i \in Y$ is the class label associated with $\mathbf{x}_i$, in this paper the labels are contexts. In addition, the boosting algorithm has access to another unspecified learning algorithm, called the weak learner f_i. In this paper, we adopts decision stump as the weak learner. While the iteration for Q times, weak learner classifies the data given. If a weak learner f_i fails in classification, its weight w_i will be increased. If f_i success in the classification, its weight does not change, but it will be decreased because all weights are normalized after one iteration. After the iteration, all weights are calculated, and

1) **Input**: Sequence of N examples $(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_N, y_N)$ with labels $y_i \in Y = \{-1, +1\}$
2) **Initialization**: $w_{i,t} = \frac{1}{N}$ for all $i = 1 \cdots N$
3) **Do for** $q = 1, 2, \cdots, Q$:
   a) Train a base learner with respect to weighted example distribution w_i and obtain hypothesis f
   b) Calculate the training error e_q of $f_q(\mathbf{x})$

$$e_q = \sum_{i:f_q(x_i) \neq y_i} w_{i,q}$$

   c) Set

$$c_q = log\frac{1 - e_q}{e_q}$$

   d) Update weight w for all $i = 1, 2, \cdots, N$

$$w_{i,q+1} = \frac{w_{i,q} * e^{-y_i * e_q * f_q(x)}}{Z_q}$$

   where Z_q is a normalization constant and we apply $Z_q = \sum_i w_{i,q+1}$ so that Z_q will be 1.
4) **Output**: the final hypothesis is given by

$$sign(\sum_{q=1}^{N} c_q * f_q(\mathbf{x}))$$

Figure 4. the Adaboost algorithm

f_i with larger weight plays an important roll in recognition. Finally, one classification function $sign(\sum_{q=1}^{N} c_q * f_q(x))$ is obtained, where c_q is a confidence coefficient and $f_q(\mathbf{x})$ is a decision(-1 or +1) when data $\mathbf{x}$ is given.

Originally, though Adaboost is one of the classifiers, focusing on the weight in the algorithm, we can measure how discriminative the features are. In other words, weight w increases when its weak classifier misses recognition while the iteration. Small w indicates that its classifier performs better in recognition of the data. Then, as w decreases, c increases, and the larger c is, the more discriminative the feature is, and vice versa.

C. Ranking of Features with Support Vector Machine

SVM is a classification algorithm that often provides competitive or superior accuracy for a large variety of real-world classification tasks [15]. Consider the problem of separating a set of training data $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \cdots, (\mathbf{x}_J, y_J)$ into two classes, where $\mathbf{x}_i \in R^N$ is a feature vector and $y_i \in \{-1, +1\}$ its class label. Supposing that the classes can be separated by the hyperplane $\mathbf{w} * \mathbf{x}_i + b$, and no knowledge about the data distribution is given beforehand, the optimal hyperplane is the one with the maximum distance to the closest points in the training dataset. We can find the optimal values for w and b by solving the following problem:

$$\min \frac{1}{2}||\mathbf{w}||^2, \quad \text{subject to } y_i(\mathbf{w} * \mathbf{x}_i + b) \geq 1, \ \forall i = 1, \cdots, n.$$

The factor of 1/2 is used for mathematical convenience. By using lagrange multipliers $\lambda_i (i = 1, \cdots, n)$, it is rewritten in this way:

$$\max \sum_{i=1}^{N} \lambda_i - \sum_{i,j=1}^{N} \lambda_i \lambda_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j,$$

$$\text{subject to } \sum_{i=1}^{N} y_i \alpha_i = 0, \ \lambda_i \geq 0,$$

and results in a classification function

$$f(\mathbf{x}) = sign\left(\sum_{i=1}^{n} \lambda_i y_i \mathbf{x}_i * \mathbf{x} + b\right). \qquad -(*)$$

Most of the λ_i take zero. Those $\mathbf{x}_i$ with nonzero λ_i are so-called support vectors, all of which is on the each hyperplain. In cases where the classes are not separable, modification of the Lagrange multipliers to $0 \leq \lambda_i \leq C, i = 1, \cdots, n$, where C is the penalty for misjudge. This arrangement is called *soft margin* and a reason SVM performs well.

The original optimal hyperplane algorithm proposed by Vapnik was a linear classifier. To obtain a non-linear classifier, one maps the data from the input space $\mathbb{R}^N$ to a high dimensional feature space by $\mathbf{x} \rightarrow \Phi(\mathbf{x})$. However non-linear classifiers were created by applying the kernel trick to maximum-margin hyperplanes. Assuming there exists a kernel function $K(\mathbf{x}, \mathbf{x}') = \Phi(\mathbf{x}) * \Phi(\mathbf{x}')$, a non-linear SVM can be constructed by replacing the inner product $\mathbf{x} * \mathbf{x}'$ by the kernel function $K(\mathbf{x}, \mathbf{x}')$ in eqn. (*). Kernels commonly used are polynomials $K(\mathbf{x}, \mathbf{x}') = (\gamma \cdot \mathbf{x} * \mathbf{x}' + c)^d$, the Gaussian Radial Basis Function (RBF) $K(\mathbf{x}, \mathbf{x}') = exp(-\gamma||\mathbf{x} - \mathbf{x}'||^2)$, and the sigmoid $K(\mathbf{x}, \mathbf{x}') = tanh(\gamma \cdot \mathbf{x} * \mathbf{x}' + c)$.

We have examined the kernels with changing their parameters; penalty C: 5000, 50000, and 50000; γ in RBF and sigmoid: 0.0001, 0.005, 0.001, 0.01, 0.1, and 1; and constant c in RBF and sigmoid: 0, 0.1, and 1. Any kernels did not show better performance than linear classification, and C of 50000 showed the best performance. The extension of 2-class SVM to N-class can be achieved e.g. by training N SVMs, one class will be separated against the others.

D. Ranking of Features by Output Footprint

Apart from the straightforward implementation effort and the impact that the complexity of an algorithm has on how well it performs on a platform with limited processing and memory capabilities, we will primarily argue for using the amount of data *produced* by a feature algorithm as a characterization. To justify this choice, we illustrate the effect of a few commonly-used feature sets, described before in section III, on a single 3D accelerometer and look at the amount of data that is produced after the features have been calculated and its cost in power to handle (i. e., record or communicate) these data, using the sensor from section II as a characteristic sensor for the measurements.

V. Analysis Experiments

A. Implementation details on peak features

A set of peaks in one periodical action consists of initial motion, main motion, and vestigial motion. Main motion describes the action itself well and has richer information, on the other hand initial and vestigial motion are noisy. In the implementation, we used the highest peak and the followings over a sliding window in order to exclude initial motion. In addition, by limiting the number of the followings, vestigial motion can be excluded. In the peak extraction algorithm, the moving average is coated by two lines which constitute a tube of epsilon width. If the cross-point is inside the epsilon tube, the cross-point is not taken in account and does not become an edge of a peak. By applying the epsilon tube, tiny peaks are neglected and only meaningful peaks are extracted. In addition to peak length and height, in this paper, we evaluate gradient and kurtosis so as to verify whether index of sharpness can be applicable to recognition. Gradient means, how much changed around the top of the peak. In this paper, we take 5 points (a top and 2 points on both sides) into consideration. Gradient is given in

$$Gradient(t) = |x_t - x_{t-2}| + |x_t - x_{t+2}|$$

where x_t is raw data of time t, and t is the time when top of the peak appears. Kurtosis means, strictly speaking how heavy base of the peak is, how keen the peak is. Kurtosis is given in

$$kurtosis(t) = \frac{n(n+1)}{(n-1)(n-2)(n-3)} \sum_{i=t-2}^{t+2} \left(\frac{x_i - \overline{x}}{s}\right)^4 - \frac{3(3n-5)}{(n-2)(n-3)}$$

where $n = 5$, $s = \sqrt{\frac{n}{n-1} m_2}$ and $m_2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{x})^2$. It is known that kurtosis of normal distribution becomes 3.

Along with the above features, Fourier coefficient is often used. Fourier coefficient is produced by Fast Fourier Transform (FFT), and maximum value of its spectral density as a feature. FFT, however, consumes much processing power. We assume that sensor has a memory to store data and these processed are done on micro-controller embedded in the sensor. Processing power of micro-controller is cheap due to limitation in size and power consumption, and it is difficult to calculate FFT on it.

All feature vector $\boldsymbol{X}$ is normalized using the following equation since each feature has different scale, where $\boldsymbol{M}$ and $\boldsymbol{S}$ are the mean and standard deviation of $\boldsymbol{X}$, respectively.

$$\boldsymbol{Z} = \frac{\boldsymbol{X} - \boldsymbol{M}}{\boldsymbol{S}}$$

After this conversion, the mean and variance of $\boldsymbol{Z}$ becomes 0 and 1, respectively. The logged data in the scenario was manually labeled, 10% of which becomes training data and mean and varians for this normalization, and the remaining 90% data is used for testing.

B. Data Set

To evaluate the features, data set was captured by a test subject who wore three sensors: wrist, hip, and ankle. The captured data consists of 16 activities: *ride a bicycle, go down in an elevator, go down stairs, kneel, run, sit in a bus, sit while working at desk, sit while eating, sit still, stand in a bus, stand still, go up in an elevator, go up stairs, walk with pushing a bicycle, walk slowly, walk.* These activities includes static, dynamic, and similar ones.

The amount of captured data is 965380 samples which is equivalent to 107 minutes. Ground truth is given by manual labeling.

C. Evaluation

In the evaluation, we used the following sets of features.

1) F1: mean and variance: Data format is $[mean_x, var_x, mean_y, var_y, mean_z, var_z]$, where $mean_x$ and var_x are mean of x-axis and variance of x-axis.

2) F2: peak length, height and mean: Data format is $[L1_x, H1_x, \cdots, L5_x, H5_x, mean_x, L1_y, H1_y, \cdots, L5_y, H5_y, mean_y, L1_z, H1_z, \cdots, L5_z, H5_z, mean_z]$, where $L1_x$ and $H1_x$ are length of first peak of x-axis and height of first peak of x-axis. However, since dimension of F2 is high, we applied Principal Component Analysis(PCA) to drop it. PCA involves a mathematical procedure that transform a number of possibly correlated variables into a smaller number of uncorrelated variables called principal component. The first principal component accounts for as much of the variability in the data as possible, and each succeeding component accounts for as much of the remaining variability as possible. Original F2 has 10 peak features and one mean for each axis. By using PCA, we extracted top 5 principal component scores from 10 peak features. In short, F2 after this transformation becomes $[s1_x, \cdots, s1_x, mean_x, s1_y, \cdots, s1_y, mean_y, s1_z, \cdots, s1_z, mean_z]$. In addition, since PCA cannot be calculated for all the data when using a real-time application, variance-covariance matrix of 10% of captured data was used for the calculation of remaining 90% data.

3) F3: peak gradient and mean: Data format is $[G1_x, \cdots, G5_x, mean_x, G1_y, \cdots, G5_y, mean_y, G1_y, \cdots, G5_z, mean_z]$, where $G1_x$ is gradient of first peak of x-axis.

4) F4: peak kurtosis and mean: Data format is $[K1_x, \cdots, K5_x, mean_x, K1_y, \cdots, K5_y, mean_y, K1_z, \cdots, K5_z, mean_z]$, where $K1_x$ is kurtosis of first peak of x-axis.

5) F5: mean, variance, min, and max: Data format is $[mean_x, var_x, min_x, max_x, mean_y, var_y, min_y, max_y, mean_z, var_z, min_z, max_z]$, where min_x and max_x are minimum of x-axis and maximum of x-axis. Please note that mean is required because the peak features alone do not have position information. In addition, for F2, F3, and F4, 5 components occupy the format as a peak feature, and mean has only one component. Since in cluster precision and boosting, all components are treated evenly, power of

Table I
RESULTS OF CLUSTER PRECISION

Contexts	F1	F2	F3	F4	F5
Ride a bicycle	0.941	0.961	0.906	0.891	0.943
Go down in an elevator	0.507	0.474	0.449	0.341	0.508
Go down stairs	0.929	0.954	0.742	0.387	0.976
Kneel	0.974	0.990	0.973	0.985	0.972
Run	0.995	0.996	0.979	0.948	0.997
Sit in a bus	1.000	0.808	0.998	0.956	0.990
Sit while working at desk	0.901	0.746	0.916	0.853	0.911
Sit while eating	0.910	0.944	0.938	0.861	0.937
Sit still	0.947	0.945	0.924	0.918	0.955
Stand in a bus	1.000	1.000	0.995	0.716	1.000
Stand still	0.986	0.981	0.892	0.441	0.987
Go up in an elevator	0.562	0.531	0.453	0.327	0.547
Go up staris	0.930	0.954	0.792	0.435	0.973
Walk with pushing a bicycle	0.985	0.991	0.956	0.887	0.985
Walk slowly	0.974	0.972	0.951	0.632	0.969
Walk	1.000	1.000	0.960	0.404	1.000
Average	0.909	0.890	0.864	0.686	0.916

influence of mean will be relatively weak. Therefore, we multiplied coefficient by mean. For preliminary evaluation, we set the coefficient in 5.

D. Results

1) Cluster precision: The first result is the cluster precision of the features described in Section 4.1. The results are shown in Table I. In the evaluation , we adopt the number of class $K = 100$ from preliminary experiment. Each column shows cluster precision for each context acted in the evaluation, and each row shows each feature (e.g. F1). For the result, especially for the column of average, the cluster precision of F1, F2, and F5 are almost same, and F3 and F4 are inferior. Comparing F3 and F4 to F2, length and height has richer information of contexts than gradient and kurtosis. Maximum and minimum also help mean and variance by fractions of percentage. Most important, for F1 and F2, length and height of peaks reach the results of variance.

2) Adaboost: A second evaluation uses confidence coefficient of Adaboost. The results are shown in Table II. Table II has 5 sub-tables, which are held together with features (e.g. F1). Confidence coefficient is inverse of error in Adaboost, and large coefficient indicates that the feature corresponding to it has contributed to classify the activities. As you see, figures of mean are higher than the others and this indicates that mean is the most discriminative for activity recognition. In addition, order of the summed confidence coefficients shown in the right-bottom in each sub-table, are corresponding to that of the results of cluster precision.

Summing up the confidence coefficients among features (e.g. for F2, $s1_x + \cdots s5_x + mean_x + s1_y + \cdots s5_y + mean_y + s1_z + \cdots s5_z + mean_z$), F1=3240.9, F2=3187.5, F3=3241.1, F4=3085.9, and F5=3616.5. These indices indicate that peak feature is as discriminative as variance and variance can be replaced by peak feature.

3) Results of Support vector machine: The third result is accuracies of Support vector machine. The reuslts are shown in Table III. Since support vector machine requires learning beforehand differently from cluster precision, we evaluated by 5-fold cross validation. The results are same as that of both two methods and our proposed peak feature performs competitive with mean and variance.

4) Computational costs: The last result is computational costs in terms of data size of the features. Before starting the argument of data size, we describe how many peaks appear in sensing data. This is because number of peaks affect its data size. Table IV shows the number of peaks per 10000 samples. The number of peaks are different for contexts. In detail, the number of peaks reflects how hard the context is, like a variance. For example, data of *run* has 8294 peaks, *walk* has 6583 peaks, *walk slowly* has 1289, and *stand still* has only 8 peaks, and this order is degree of activity. Figure 5 shows the graph of raw data and peaks of *walk* and *sit*. The upper one shows *walk* and the lower one shows *sit*, and few peaks appears in *sit* data.

Number of data means how many figures the features of raw data needs. The number of raw data becomes 90000 because 1 sample is made of 3 axes by 3 sensors and 10000-sample data makes 90000 data. F1, mean and variance, needs 180000 data because 1 sample is made of 2 features (mean and variance) of 3 axes by 3 sensors. 11512 peak of *bike* has 46048 data because 1 peak has height, width, mean, and time. F5 produces 36000 data because of 4 kinds of feature (mean, variance, min, and max).

Size of data shows the size of the data file. In the evaluation, the data was stored with in the csv format. For 10000 samples of raw data, F1 and F5 generate 347 KB, 1230 KB and 1939 KB, respectively, while peak features (F2) generate 136 KB at most and 39 KB on average. Supposing all the contexts are evenly performed, F2 reduces the data size of raw data and F1 by 96.8% and 88.8%, respectively.

In addition, we evaluated accuracies when sampling frequency is changed. In the evaluation, sampling frequency is changed to 150 Hz, 75 Hz, 50 Hz, 30 Hz, 25 Hz, 15 Hz, 10 Hz, 5 Hz, 3 Hz and 1 Hz, then data rate (KB/min) of F1 and F2 are measured. From the results shown in Table V, data rate of F2 at 30 Hz is 18.9 KB/min and frequency of F1 at the equivalent data rate 22.1 KB/min is 3 Hz. Accuracies of both features are 90.7% and 82.1%, and it is clear that our proposed feature takes advantage over conventional mean and variance feature at same data rate.

Here, please note that the previous results are on the condition when all 16 contexts are evenly conducted and time period for static contexts is much longer that that for dynamic ones in daily life. [16] mentioned that we *sit* for 3000 minutes out of 4100 minutes from 16-day monitoring. Besides, since this result does not include *sleeping*, considering 24-hour data including sleeping, larger part

Table II
RESULTS OF ADABOOST

	place	x-axis		y-axis		z-axis		Sum
		$mean$	var	$mean$	var	$mean$	var	
F1	Wrist	111.2	54.3	63.9	96.1	82.9	50.1	
	Hip	1874.4	107.4	93.2	62.6	67.8	57.2	
	Ankle	89.5	70.4	95.3	63.8	102.9	98.0	
	Total	2075.2	232.0	252.5	222.4	253.6	205.2	3240.9

	place	x-axis						y-axis						z-axis						Sum
		$s1$	$s2$	$s3$	$s4$	$s5$	$mean$	$s1$	$s2$	$s3$	$s4$	$s5$	$mean$	$s1$	$s2$	$s3$	$s4$	$s5$	$mean$	
F2	Wrist	32.0	15.4	11.4	9.6	19.3	115.0	21.4	15.1	10.6	10.4	13.3	67.3	6.0	13.2	10.3	10.4	9.9	68.3	
	Hip	36.8	5.7	8.4	10.4	10.5	1861.9	17.3	11.8	10.3	11.3	11.3	79.3	16.2	14.0	13.4	5.9	22.3	61.8	
	Ankle	36.2	12.2	9.3	14.8	13.9	83.7	20.9	16.1	13.3	12.3	9.6	89.9	25.4	15.1	16.0	13.5	19.0	98.8	
	Total	105.1	33.2	29.2	34.8	43.7	2060.6	59.5	42.9	34.2	33.9	34.1	236.6	47.6	42.2	39.7	29.9	51.2	229.0	3187.5

	place	x-axis						y-axis						z-axis						Sum
		$G1$	$G2$	$G3$	$G4$	$G5$	$mean$	$G2$	$G2$	$G3$	$G4$	$G5$	$mean$	$G1$	$G2$	$G3$	$G4$	$G5$	$mean$	
F3	Wrist	36.9	26.2	7.9	7.0	2.0	117.7	27.7	12.4	7.7	10.2	5.7	64.6	29.6	10.9	4.2	2.2	6.9	68.3	
	Hip	59.2	11.1	6.6	6.4	8.5	1861.1	37.9	12.6	8.9	5.8	6.9	90.8	39.5	14.3	8.0	3.3	6.3	60.4	
	Ankle	60.7	12.4	12.8	14.4	9.3	86.5	30.2	22.6	7.7	9.5	6.4	100.3	41.2	11.9	11.3	6.5	10.3	91.8	
	Total	156.8	49.7	27.2	27.8	19.8	2065.2	95.8	47.5	24.3	25.5	19.0	255.7	110.3	37.0	23.5	12.0	23.5	220.6	3241.1

	place	x-axis						y-axis						z-axis						Sum
		$K1$	$K2$	$K3$	$K4$	$K5$	$mean$	$K2$	$K2$	$K3$	$K4$	$K5$	$mean$	$K1$	$K2$	$K3$	$K4$	$K5$	$mean$	
F4	Wrist	36.3	20.7	9.3	6.8	5.0	110.6	23.0	10.2	5.9	9.3	8.4	79.2	24.4	4.2	8.1	4.0	6.8	68.2	
	Hip	37.4	13.9	6.6	5.9	5.9	1865.1	18.1	8.7	6.9	5.3	10.0	79.3	17.7	7.1	6.4	4.5	7.6	65.8	
	Ankle	33.8	10.1	7.2	10.3	9.1	82.0	26.4	14.0	10.9	4.2	10.8	98.1	23.1	15.4	6.8	9.3	9.6	92.3	
	Total	107.5	44.7	23.1	23.0	19.9	2057.7	67.6	32.9	23.7	18.9	29.2	256.6	65.2	26.7	21.2	17.7	24.0	226.3	3085.9

	Place	x-axis				y-axis				z-axis				Sum
		min	max	mean	var	min	max	mean	var	min	max	mean	var	
F4	Wrist	37.4	42.8	91.5	24.5	83.9	41.4	64.8	45.2	27.0	39.5	55.1	26.6	
	Hip	71.3	48.3	1857.3	60.7	43.3	58.3	58.0	34.9	25.2	54.2	48.7	33.3	
	Ankle	72.4	24.5	75.7	40.6	31.6	50.0	66.3	55.2	43.2	59.0	75.2	49.6	
	Total	181.1	115.6	2024.5	125.7	158.9	149.7	189.1	435.3	95.5	152.7	178.9	109.4	3616.5

Table III
RESULTS OF SVM

Contexts	F1	F2	F3	F4	F5
Ride a bicycle	0.887	0.852	0.602	0.894	0.775
Go down in an elevator	0.471	1.000	1.000	0.968	1.000
Go down staris	0.998	0.976	0.887	0.934	0.992
Kneel	0.954	0.778	0.867	0.701	0.953
Run	0.995	0.940	0.812	0.937	0.998
Sit in a bus	1.000	1.000	1.000	0.958	1.000
Sit while working at desk	0.889	0.896	0.916	0.853	0.882
Sit while eating	0.910	0.817	0.813	0.743	0.876
Sit still	0.818	0.651	0.675	0.810	0.715
Stand in a bus	0.993	0.995	1.000	0.859	1.000
Stand still	0.929	0.988	0.988	0.988	0.975
Go up in an elevator	0.596	0.740	0.774	0.860	0.977
Go up stairs	0.991	0.859	0.859	0.737	0.978
Walk with pushing bicycle	0.931	0.768	0.698	0.797	0.964
Walk slowly	0.967	0.912	0.858	0.738	0.944
Walk	1.000	0.989	0.969	0.943	1.000
Average	0.896	0.885	0.857	0.857	0.939

Table IV
NUMBER OF PEAKS PER 10000 SAMPLES (CA 67 SEC.)

	Contexts	Number of Peaks				Number of Data	Data size [KB]
		Wrist	Hip	Ankle	Total		
Raw	-	-				90000	347
F1	-	-				180000	1230
F2	Ride a bike	5949	2137	3426	11512	46048	136
	Go down in an elevator	13	0	8	21	84	1
	Go down stairs	1265	2318	3001	6584	26336	81
	Kneel	79	46	13	138	552	2
	Run	1351	3272	3671	8294	33176	100
	Sit in a bus	8	0	0	8	32	1
	Sit while working at desk	73	27	25	125	500	2
	Sit while eating	690	167	151	1008	4032	13
	Sit still	53	37	35	125	500	2
	Stand in a bus	93	49	321	463	1852	6
	Stand still	1	3	4	8	32	1
	Go up in an elevator	16	0	3	19	76	1
	Go up stairs	981	1584	2431	4996	49984	62
	Walk with pushing a bicycle	5145	2021	2734	9900	39600	119
	Walk slowly	275	275	739	1289	5156	17
	Walk	1145	2342	3096	6583	26332	80
	Average	1071	892	1229	3192	12768	39
F5	-	-				360000	1939

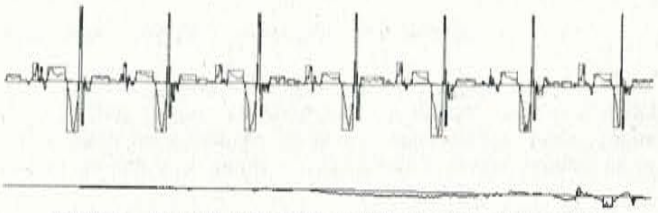

Figure 5. Raw data and peaks of walk (upper) and sit (lower)

Table V
PEAK FEATURES VS. MEAN+VAR FEATURES IN SAME DATA RATE.

Sampling frequency	mean+var (F1)		Peak width+height (F2)	
	Data rate [KB/min]	Cluster precision	Data rate [KB/min]	Cluster precision
150 Hz	1106	90.6	35.1	89.0
75 Hz	553	91.7	30.0	90.0
50 Hz	369	90.5	25.7	89.7
30 Hz	221	90.2	18.9	90.7
25 Hz	184	90.6	16.7	89.4
15 Hz	111	89.7	12.5	87.3
10 Hz	73.7	88.4	9.68	87.5
5 Hz	36.9	89.0	5.63	85.3
3 Hz	22.1	82.1	3.38	80.4
1 Hz	7.37	71.6	1.63	67.5

of our activity can be static. In order to prove that our proposal suits for actual sensed data, we captured 24-hour data of acceleration on hip at 20 Hz excluding bath, and measured its data size. A waveform of the captured data and data size of the features are shown in Figure 6 and Table VI, respectively. For Figure 6, peaks rarely appear while sleeping. Many peaks seem to appear while active though, static activities can be seen by zooming it. In addition, for Table VI, comparing to raw data and F1, the peak feature (F2) reduces its data size by 96.6% and 99.2%, and it has been celar that degree of data size reduction is larger than on the previous condition where all the contexts are evenly

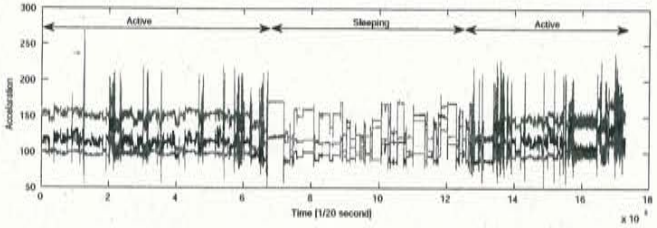

Figure 6. A waveform of acceleration for 24 hours.

Table VI
DATA SIZE OF 24-HOUR DATA FOR EACH FEATURE.

Features	Number of data	Data size [KB]
Raw data	5184000	19776
mean+var (F1)	10368000	80082
Peak width+height (F2)	209144	680

performed.

VI. CONCLUSION

We have evaluated distinctiveness and computational costs of several features often used in wearable sensing. In this paper, we used some combination of mean, variance, minimum, maximum, and peak of wave, and confirmed length and height of peaks are comparable to variance in distinctiveness for a context-aware system in wearable computing environment. In addition, we measured size of file produced by the feature-extraction process, and confirmed that peak features can produce much smaller file than mean and variance which has been used in many researches.

As future work, we plan to evaluate with a long-term dataset collected in real life. In our current evaluation, reduction in data size was obtained just in realistic environment, but one where all contexts evenly occur. From long-term observation, we measure how much the feature can be reduced and how long the data can be logged.

ACKNOWLEDGMENT

This research was supported in part by a Grant-in-Aid for Scientific Research (A) (20240009) and Priority Areas (21013034) of the Japanese Ministry of Education, Culture, Sports, Science and Technology, and a Grant-in-Aid for JSPS Fellows (21·249) of Japan Society for the Promotion of Science.

REFERENCES

[1] M. Miyamae, T. Terada, M. Tsukamoto, and S. Nishio: "Design and Implementation of an Extensible Rule Processing System for Wearable Computing," in *Proc. of the 1st IEEE Int'l Conference on Mobile and Ubiquitous Systems: Networking and Services (MobiQuitous 2004)*, pp. 392–400 (Aug. 2004).

[2] M. Toda, J. Akita, S. Sakurazawa, K. Yanagihara, M. Kunita, and K. Iwata: "Wearable Biomedical Monitoring System Using TextileNet," in *Proc. of the 10th IEEE Int'l Symposium on Wearable Computers (ISWC 2006)*, pp.119–120 (Oct. 2006).

[3] C. L. Shen, T. Kao, C. T. Huang, and J. H. Lee: "Wearable Band Using a Fabric-Based Sensor for Exercise ECG Monitoring," in *Proc. of the 10th IEEE Int'l Symposium on Wearable Computers (ISWC 2006)*, pp.143–144 (Oct. 2006).

[4] K. Ouchi, T. Suzuki, and M. Doi: "LifeMinder: A wearable Healthcare Support System Using User's Context," in *Proc. of the 2nd Int'l Workshop on Smart Appliances and Wearable Computing (IWSAWC 2002)*, pp. 791–792 (July 2002).

[5] K. V. Laerhoven and H. W. Gellersen: "Spine versus Porcupine: a Study in Distributed Wearable Activity Recognition," in *Proc. of the 8th IEEE Int'l Symposium on Wearable Computers (ISWC 2004)*, pp. 142–149 (Oct. 2004).

[6] F. Naya, R. Ohmura, F. Takayanagi, H. Noma, and K. Kogure: "Workers' Routine Activity Recognition using Body Movement and Location Information," in *Proc. of the 10th IEEE Int'l Symposium on Wearable Computers (ISWC 2006)*, pp. 105–108 (Oct. 2006).

[7] T. Stiefmeier, G. Ogris, H. Junker, P. Lukowics, and G. Tröster: "Combining Motion Sensors and Ultrasonic Hands Tracking for Continuous Activity Recognition in a Maintenance Scenario," in *Proc. of the 10th IEEE Int'l Symposium on Wearable Computers (ISWC 2006)*, pp. 97–104 (Oct. 2006).

[8] A. Y. Benbasat and Joseph A. Paradiso: "A Framework for the Automated Generation of Power-Efcient Classiers for Embedded Sensor Nodes" in *Sensys 2007*, pp. 219-232 (2007).

[9] J. Ho and S. S. Intille: "Using Context-Aware Computing to Reduce the Perceived Burden of Interruptions from Mobile Devices," in *Proc. Conference on Human Factors in Computing System (CHI 2005)*, pp. 909–918 (Apr. 2005).

[10] Nike+iPod: http://www.apple.com/ipod/nike/.

[11] Y. Kawahara, C. Sugimoto, S. Arimitsu, A. Morandini, T. Itao, H. Morikawa, and T. Aoyama: Context inference techniques for a wearable exercise support system, in *Proceedings of ACM International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH 2005)*, p. 95 (July/Aug. 2005).

[12] D. E. Knuth. The Art of Computer Programming, 3rd edn. Boston: Addison-Wesley, pp. 232. (1998).

[13] T. Huynh and B. Schiele: "Analyzing Features for Activity Recognition," in *Proc. of the 2005 joint conference on Smart objects and ambient intelligence: innovative context-aware services: usages and technologies*, pp. 159–163 (Oct. 2005).

[14] Y. Freund and R. E. Schapire: "Experiments with a new boosting algorithm," in *Proc. of the Thirteenth International Conference on Machine Learning*, pp. 148–156 (1996).

[15] V. Vapnik: "The Nature of Statistical Learning Theory," Springer, (1995).

[16] T. Huynh, M. Fritz and B. Schiele: "Discovery of Activity Patterns using Topic Model," in *Proc. of 10th International Conference on Ubiquitous Computing (UbiComp 2008)* (Sep. 2008)