Pervasive and Mobile Computing 8 (2012) 115–130

Contents lists available at SciVerse ScienceDirect

Pervasive and Mobile Computing

journal homepage: www.elsevier.com/locate/pmc

A hierarchical approach to real-time activity recognition in body sensor networks

Liang Wang [a,b], Tao Gu [b,*], Xianping Tao [a], Jian Lu [a]

[a] State Key Laboratory for Novel Software Technology, Nanjing University, China

[b] Department of Mathematics and Computer Science, University of Southern Denmark, Denmark

ARTICLE INFO

Article history:
Received 18 November 2010
Received in revised form 5 December 2010
Accepted 6 December 2010
Available online 15 December 2010

Keywords:
Real-time activity recognition
Pattern mining
Wireless body sensor network

ABSTRACT

Real-time activity recognition in body sensor networks is an important and challenging task. In this paper, we propose a real-time, hierarchical model to recognize both simple gestures and complex activities using a wireless body sensor network. In this model, we first use a fast and lightweight algorithm to detect gestures at the sensor node level, and then propose a pattern based real-time algorithm to recognize complex, high-level activities at the portable device level. We evaluate our algorithms over a real-world dataset. The results show that the proposed system not only achieves good performance (an average utility of 0.81, an average accuracy of 82.87%, and an average real-time delay of 5.7 seconds), but also significantly reduces the network's communication cost by 60.2%.

1. Introduction

Sensor-based human activity recognition has recently attracted much attention in pervasive computing. In this paradigm, various sensors are typically attached to a human body or embedded in the environment. Observations are collected in the form of a continuous sensor data stream, and the data stream is then interpreted by a recognition system. The computation usually involves two phases: (1) observations are used to train an activity model; (2) the trained model will then be used to predict activities for new observations.

Sensor-based activity recognition has many potential applications, including health care [1], assisted living [2], and sports coaching [3]. In the past few years, many efforts have been devoted to this task in various domains by researchers and industrial participants. However, we have not seen any real application being deployed in real life. A number of important and challenging issues still remain unsolved. First, a practical system should be able to recognize activities in real time. The real-time requirement demands a one-pass algorithm over sensor data with short real-time delay. Multiple passes are usually not possible due to the large volume of data arriving continuously at a processing server. Second, most of the wireless body sensor networks typically use star topology in which data generated from each sensor node are transmitted to a centralized server for processing. The network communication can be very costly due to the high sampling rate of the accelerometer. Third, processing sensor data at a fix server may not be practical since humans often move from one place to another in their daily lives. In this case, mobile and portable devices are more suitable for the task, and hence a lightweight and portable solution is highly desired.

To address the above challenges, in this paper we propose a hierarchical model to recognize human activities in real time, which we first identify gestures at the sensor node level, and then recognize complex, high-level activities at the portable device level. Our motivation is that a high-level activity typically includes a sequence of physical gestures and ambulation in the execution. For example, the *household cleaning* activity can be better derived from a sequence of hand

* Corresponding author.
E-mail addresses: wangliang@ics.nju.edu.cn (L. Wang), gu@imada.sdu.dk (T. Gu), txp@nju.edu.cn (X. Tao), lj@nju.edu.cn (J. Lu).

1574-1192/$ – see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.pmcj.2010.12.001

gestures (i.e., wiping and mopping patterns), body gestures (i.e., up and down patterns), and ambulation. In addition, the hierarchical model enables us to distribute the computation from a centralized server to individual sensor nodes so that the network's communication cost can be significantly reduced. In this work, we first design a wireless body sensor network, consisting of a number of sensor nodes attached to a subject for collecting observations. We then design our real-time recognition algorithms which operate in the following two stages. First, acceleration data are processed immediately at each sensor node by a fast, lightweight, gesture recognition algorithm to detect the gestures of a subject. This is done by discovering a template for each gesture using an unsupervised method, and then matching acceleration data with an appropriate template based on the minimum distance which is computed using Dynamic Time Warping (DTW) [4]. This algorithm outputs the gestures of both hands (i.e., moving hand up and down) and body (i.e., walking and running). Second, the recognized gestures and other sensor readings (i.e., tagged object and location) will be transmitted over wireless links to a centralized device for further processing. We propose a real-time, discriminative pattern based approach to recognize high-level, complex activities. We adapt an off-line, Emerging Pattern based algorithm [5] which is capable of recognizing both simple and complex activities (i.e., interleaved [6] and concurrent activities [7])—to meet the real-time requirement in this work. We use a real-world dataset and develop a real-time simulator to evaluate our proposed algorithms. Our experimental studies show that the proposed system is promising in recognizing both gestures and activities in real time.

In summary, the paper makes the following contributions:

- To the best of our knowledge, this paper presents the first formal study of a real-time, hierarchical model to recognize both physical, simple gestures and high-level, complex activities using a body sensor network.
- The proposed algorithms are designed with not only a real-time constraint, but also a lightweight constraint for practical deployment.
- We conduct comprehensive experiments, and the results show that our algorithms achieve good performance in recognition accuracy and real-time delay, and better communication efficiency.

The rest of the paper is organized as follows. Section 2 discusses the related work. In Section 3, we present our body sensor network and provide a system overview. We present our algorithm for gesture recognition in Section 4, followed by the algorithm for activity recognition in Section 5. Section 6 reports our empirical study, and finally Section 7 concludes the paper.

2. Related work

In pervasive computing, researchers are recently interested in recognizing activities using wearable sensor networks. In such a network, various sensors are used to directly measure a user's movements (e.g., accelerometer), the living environment (e.g., temperature, humidity and light sensors), object use (e.g., wrist-worn RFID sensor) and user location (e.g., indoor location sensor).

Most of the existing work [8,9,6,7,10,11] has been done in an off-line manner. There are some recent works focusing on real-time activity recognition. Tapia et al. [12] proposed a real-time algorithm based on a decision tree for physical activities (i.e., gestures) recognition. The sensor readings from 3-axis accelerometers are transmitted wirelessly to a laptop computer for processing. A C4.5 classifier is first trained, and then used to recognize gymnasium activities in real time. Krishnan et al. [13] proposed an AdaBoost algorithm based on decision stumps for real-time classification of gestures (i.e., walking, sitting and running) using 3-axis accelerometer sensors. He et al. [14] presented a Hidden Markov Model for real-time activity classification using acceleration data collected from wearable sensors. The model is used to classify a number of gestures such as standing, sitting, and falling.

Some recent work has been done to recognize gestures or activities in real time on resource-constraint devices. Györbíró [15] presented a real-time mobile activity recognition system consisting of wireless body sensors, a smartphone, and a desktop workstation. A sensor node contains an accelerometer, a magnetometer, and a gyroscope. They proposed a recognition model based on feed-forward back-propagation neural networks which are first trained at a desktop workstation, and then run at the smartphone to recognize six different gestures. Liu et al. [16] proposed an efficient gesture recognition method based on a single accelerometer using DTW. They first define a vocabulary of known gestures based on training, then use these pre-defined templates to recognize hand gestures.

Different from the above works which use a single layer model (i.e., a single point for data processing) for activity recognition, we propose a distributed approach in which the computation is divided to gesture recognition at sensor node and high-level activity recognition at a mobile device. Similar to the DTW-based hand gesture recognition algorithm [16], we use a similar approach to compute the distance of a test instance and a gesture template. However, their pre-defined templates are obtained by a training process in a supervised manner whereas we obtain various hand and body gesture templates using an unsupervised method. The evaluation of existing activity recognition systems mainly focused on accuracy and real-time performance. In addition to these measurements, we evaluate the communication efficiency and the portability of our system which are important for real-life deployment.

3. Body sensor network design and system overview

We design a wireless body sensor network as shown in Fig. 1. It consists of five sensor nodes—three IMOTE2 motes and two RFID reader motes. An IMOTE2 mote is located on each wrist and the body of a subject to capture hand and body

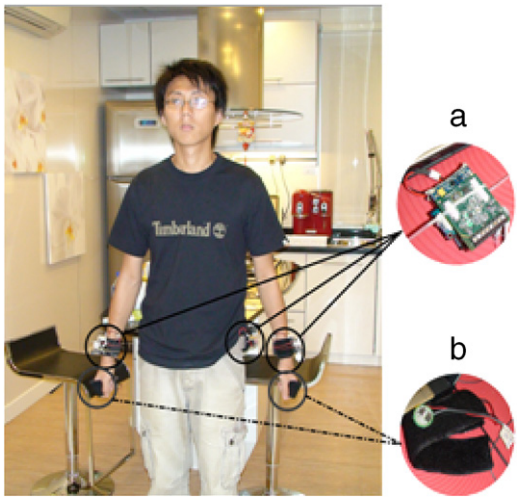

Fig. 1. Our body sensor network, (a) an IMOTE2 mote, (b) an RFID reader mote.

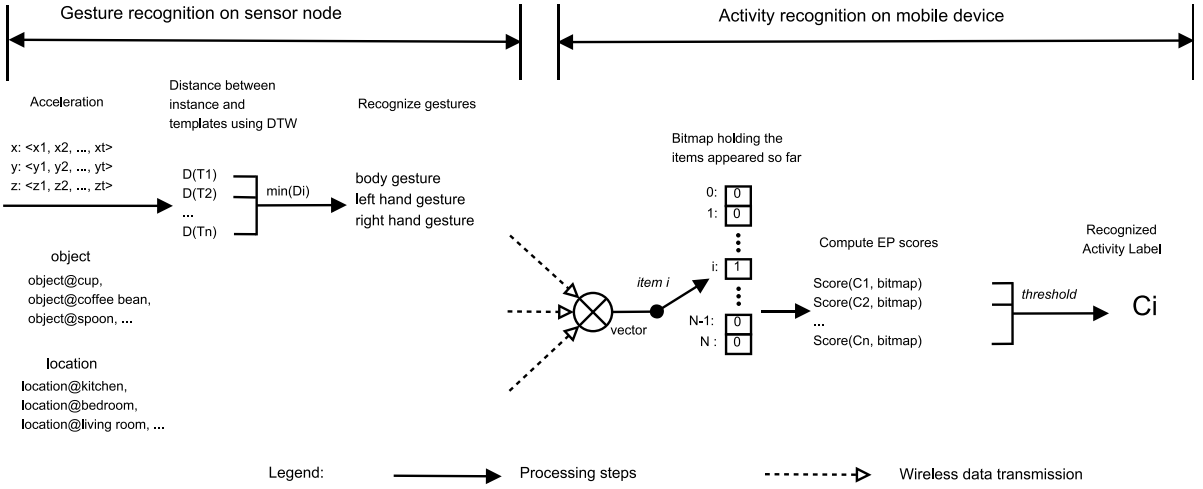

Fig. 2. Overview of our real-time, hierarchical recognition model.

movement; it consists of an IPR2400 processor/radio board and an ITS400 sensor board with a 3-axis accelerometer, as shown in Fig. 1(a). An RFID reader mote is located on each hand to capture object use; it consists of a MICA2Dot mote and a coin-size, short-range RFID reader, as shown in Fig. 1(b). An RFID reader mote is able to detect the presence of a tagged object within a few centimeters. In addition, detecting the user's location at room-level granularity is done in a simple way that an UHF RFID reader is located in each room to sense the proximity of a subject wearing a UHF tag. The sensor data captured by these motes can be transferred to sink nodes and logged in servers.

Fig. 2 gives an overview of our real-time recognition system. The system operates in two stages—gesture recognition at the sensor node and activity recognition at a mobile device. First, each IMOTE2 mote processes its acceleration data to recognize gestures by a fast, lightweight *template matching* algorithm. This is done by first obtaining a specific template for each gesture pattern using an unsupervised method, then matching a test instance obtained by applying a sliding window over the data stream with each possible template. A match is found when the distance between the test instance and a template is a minimum, and then the test instance will be assigned with the corresponding template label. We compute the distance using Dynamic Time Warping which is an efficient, lightweight algorithm to match two time series samples. Next, recognized gestures, tagged objects and user locations from each node will be transmitted over a wireless link to a centralized device. The data will be synchronized and processed to generate a discrete vector stream. We then apply a discriminative pattern based approach to discover complex, high-level activities in real time. A bitmap is used to temporarily hold the data before they can be recognized. When examining a new vector, we map items in the vector into the bitmap. Given a window size of l sensing periods, the algorithm periodically computes the score between the input data kept in the bitmap and discriminative patterns mined for each class C_i. If the score of one class exceeds a predefined *threshold*, it outputs that class as the recognized activity and clear the bitmap; otherwise, it waits for new input data until one activity class is recognized.

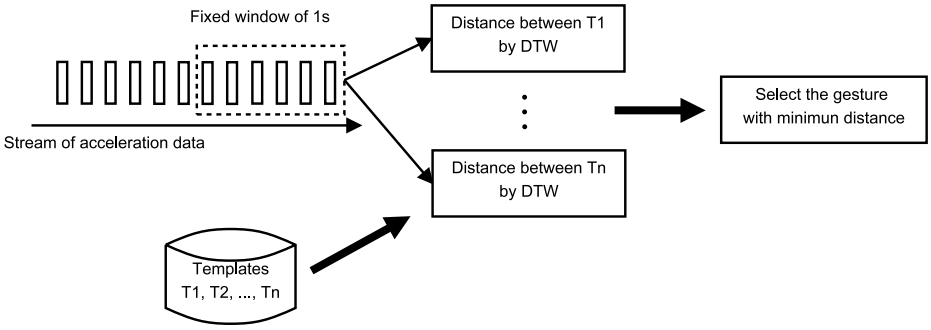

Fig. 3. Gesture recognition using DTW.

4. Gesture recognition

This section describes how we process acceleration data at the sensor node level to recognize the gestures of a subject.

4.1. Sensor data collection

In our body sensor network, we use three IMOTE2 motes with 3-axis accelerometers—one on each wrist to capture the hand motion patterns; one on the body to capture the body motion patterns. An acceleration data stream is generated at each node with a constant sampling rate, the data format is shown as follows:

$$[\text{time\_stamp}]\langle \text{sensor\_}id\rangle\langle x\rangle\langle y\rangle\langle z\rangle$$

where time_stamp denotes time stamp, sensor_*id* denotes sensor node ID, x, y and z are acceleration readings on the three different directions and they can be decoded in a 12-bit resolution ranging from $-2g$ to $+2g$. The data can be transformed to a three-dimensional stream of integers containing the readings of the three axes using a simple parser. A sample taken from the data stream we collected is shown as follows:

$$[12/08/2008 \quad 13:22:24:765] \quad 163 \quad -664 \quad -306 \quad 612. \tag{1}$$

In this example, 163 represents the sensor ID on the subject's right hand.

4.2. Gesture templates

To recognize gestures over acceleration data, we first need to define a set of gesture templates for left hand, right hand and body, respectively. A common way to obtain these templates is based on supervised learning, e.g., the work done in [16]. Using this method, the training data for different hand and body gestures is collected and assigned with proper labels, and will then be used to define a template for each gesture. However, in real deployment, labeling such training data can be very time consuming since an annotator has to analyze the video record for each gesture, and sometime it is not possible if hand motions are blocked from the video camera. In addition, the accuracy of labeling remains uncertain because there is no common vocabulary for all the gestures performed in real life.

In this work, we propose an unsupervised method to discover gesture patterns. We use a K-Medoids clustering method to discover these template gestures. This method finds the k representative instances which best represent the clusters. The number of clusters is set to five for body gestures and ten for each hand in our study. The intuition behind this choice is that there are typically ten patterns for hand movements including moving forward, backward, left, right, left and up, left and down, right and up, right and down. Similarly for body gestures, there are typically five patterns, i.e., moving up, sitting down (contain both a moving down and moving backwards), moving left, right and forward.

4.3. Identifying gestures

Based on the templates we obtained above, we apply a *template matching* algorithm to identify gestures. To get test instances, we use a sliding window with a fixed length of 1 s to segment the data stream. For each instance obtained, we match the instance with the pre-defined templates using DTW which is a classic dynamic programming based algorithm to match two time series with temporal dynamics. We use the Euclidean distance to compute the distance between two time samples. A match is found when the distance between the test instance and a gesture template is a minimum, and then the test instance will be assigned with the corresponding template label. The *template matching* algorithm is illustrated in Fig. 3.

4.4. Complexity analysis

We now analyze the time and space complexities for the *template matching* algorithm. Let $S[1 \ldots M]$ and $T[1 \ldots N]$ denote two time series, n denotes the number of templates stored. The time and space complexities of matching S and T are both $O(M \cdot N)$. The total time complexity of recognition is thus $O(n \cdot M \cdot N)$, the space complexity is $O(M \cdot N)$.

5. Real-time activity recognition

The recognized gestures, tagged objects and user locations from each node will be transmitted to a centralized device for recognizing complex, high-level activities. We adapt an offline, Emerging Pattern based algorithm [5] to real-time requirements in this work. In this section, we first give the background of the Emerging Pattern, and then describe how to use the Emerging Pattern to recognize complex, high-level activities in real time.

5.1. Background of emerging pattern

Emerging Pattern (EP) describes significant differences between different classes of data [17]. An EP is a set of items, and it occurs frequently in one class and rarely in all the other classes. The class in which an EP occurs the most frequently is called the class of the EP. An EP can be viewed as a representative pattern of its class. If an instance contains an EP, then it is very likely that the instance belongs to the class of the EP.

Formally, an EP is defined as follows. Let $D = \{t_1, t_2, \ldots, t_n\}$ be a dataset containing a set of instances, and each instance is a set of items. In our case, an item can be a user gesture, an object touched by users or the location of the user. Each instance has a class label which indicates the activity of the user. Let $\mathcal{C} = \{C_1, C_2, \ldots, C_k\}$ be the set of classes labels. A pattern X is an itemset, and its support in D is defined as the proportion of instances in D that contain it, denoted as $\text{supp}_D(X) = |\{t|X \subseteq t, t \in D\}|/|D|$. The discriminative power of an EP X is measured by the ratio of the support of the EP in its class to the support of the EP in all the other classes, denoted as $\text{GrowthRate}_D(X) =$

$$\begin{cases} 0, & \text{if } \text{supp}_{D_c}(X) = 0 \\ \infty, & \text{if } \text{supp}_{D_c}(X) > 0 \text{ and } \text{supp}_D(X) = \text{supp}_{D_c}(X) \\ \dfrac{\text{supp}_{D_c}(X)}{\text{supp}_D(X) - \text{supp}_{D_c}(X)}, & \text{otherwise} \end{cases}$$

where c is the class of X, and D_c is the set of instances belonging to class c.

Definition 1 (*Emerging Pattern*)**.** Given a labeled dataset D, if $\text{supp}_D(X) \geqslant \text{min\_sup}$ and $\text{GrowthRate}_D(X) \geqslant \rho$, then X is called a $\rho -$ Emerging Pattern, where min_sup is a predefined minimum support threshold and ρ is a predefined minimum growth rate threshold.

5.2. Mining emerging patterns

To use EPs for activity recognition, we first obtain a set of EPs for each activity class. This is done by mining EPs from a training dataset containing labeled activity instances. For each activity C_i, we mine a set of EPs that occur frequently in C_i, but rarely in other classes. We denote this set of EPs as EP_{C_i}. We discover the EPs by an efficient algorithm described in [18]. An example of an EP for the *brushing hair* activity is shown as follows.

{object@comb, gesture@body_forward, gesture@right_forward_upward,
gesture@left_forward, location@bathroom, object@detangling_spray}. (2)

There are typically many EPs with different growth rates being discovered for each activity. To reduce the computation cost, we only select the EPs with the growth rate of $+\infty$ (i.e., the maximum discriminative power) for our recognition algorithm described in the next section.

5.3. EP-based, real-time activity recognition

The offline recognition algorithm [5] use EPs to recognize activities. Although it is effective, it works off-line and there are at least two scans over the data stream. As discussed, multiple scans are not possible in real-time activity recognition. Thus, this algorithm cannot be directly applied in this case. In this work, we design a fast, EP-based algorithm for real-time activity recognition.

First, gesture, object and location data will be synchronized and processed to generate a discrete vector stream.

A vector has the following form:

⟨body_gesture, left_gesture, right_gesture, left_object, right_object, location⟩. (3)

We then map each item in a vector to an integer. A bitmap is used to hold the items that have appeared so far. The ith bit in the bitmap is 1 if item i has appeared, otherwise, it is 0. Initially, all the bits in the bitmap are set to 0. When a new vector is generated, all the bits corresponding to the items in the vector are set to 1.

Next, given that the bitmap contains an EP, named X, which belongs to activity class C_i, we define a score function to measure the contribution of X as follows.

$$\text{Score}(C_i, \text{bitmap}) = \sum_{X \subseteq \text{bitmap}, \text{class}(X)=C_i} \frac{\text{GrowthRate}(X)}{\text{GrowthRate}(X)+1} \tag{4}$$

where class(X) is the class of X. Note that this score is basically the conditional probability that the activity class is C_i given it contains X [19]. If there exists an activity C such that Score(C, bitmap) is higher than a predefined threshold, then C will be the output as the recognized activity and the bitmap is reset by clearing all the bits to 0. If the scores for all the possible activities are below the threshold, then it outputs nothing and waits for a new vector. The computation is done recursively until the end of the data stream. The entire process is described in Algorithm 1.

Algorithm 1 EP-based Real-time Algorithm

Input: a feature vector sequence $V = \{v_1, v_2, ..., v_T\}$ with a length of T;
activities $\{C_1, C_2, \cdots, C_m\}$;
sliding-window size l;
Output: recognized activity sequence.

```
1: Bitmap bitmap;
2: while t ⩽ T
3:    count = 0;
4:    for each item in v_t ∼ v_{t+l}
5:       bitmap[key(item)] = 1;
6:       count = count + 1;
7:    end for
8:       flag = false;
9:    do
10:      for i = 1 to m
11:         if Score(C_i, bitmap) > threshold then
12:            Recognize the current activity as C_i;
13:            Set all elements in the bitmap to 0;
14:            t = t + count;
15:            count = 0;
16:            flag = true;
17:         end if
18:      end for
19:      if flag == false
20:         for each item in v_{t+count}
21:            bitmap[key(item)] = 1;
22:         end for
23:         count = count + 1;
24:      end if
25:   while flag == false
26: end for
```

5.4. Complexity analysis

Let $V[1 \ldots n]$ be the input vector sequence generated in the previous step, k be the number of activities in our system. When examining an input vector, we compute the score for each of the k activities. Let m be the number of EPs and l be the average number of items contained in an EP. The time complexity of matching EPs with items stored in the bitmap is $O(m \cdot l)$. After all the EPs have been checked, we check which class has a score not less than the threshold. The cost of this step, is $O(k)$. Since we only make one pass through the input vector sequence, the time complexity of recognizing the whole sequence is then $O((m \cdot l + k) \cdot n)$.

Let N be the total number of items. The space cost by holding the bitmap is $\Theta(N)$. The space cost for holding the EPs is $\Theta(m \cdot l)$.

6. Empirical studies

We evaluate our system in this section. We first provide a description of our evaluation methodology and the metrics we use. We then evaluate the system in several aspects.

6.1. Metrics

To evaluate the performance of a real-time activity recognition system, two metrics, *accuracy* and *delay*, are commonly used. We propose a new metric—*utility*, which provides a better trade-off between *accuracy* and *delay*.

Table 1
Activities performed.

Index	Activity	Duration (min)	Index	Activity	Duration (min)
0	Making coffee	36.7	13	Ironing	32.2
1	Making tea	45.0	14	Eating meal	89.5
2	Making oatmeal	25.8	15	Drinking	17.1
3	Frying eggs	22.8	16	Taking medication	8.7
4	Making a drink	14.2	17	Cleaning a dining table	32.95
5	Applying makeup	38.7	18	Vacuuming	26.2
6	Brushing hair	15.4	19	Taking out trash	6.8
7	Shaving	13.5	20	Using phone	113.3
8	Toileting	22.9	21	Watching TV	62.7
9	Brushing teeth	45.3	22	Watching DVD/movies	24.3
10	Washing hands	22.9	23	Using computer	40.2
11	Washing face	13.4	24	Reading book/magazine	71.1
12	Washing clothes	34.3	25	Listening music/radio	96.2

Delay: The recognition delay δ is defined as the period when the recognition algorithm starts taking new readings until an activity is recognized. It consists of the time δ_w spent on waiting for data and the time δ_r spent on executing the recognition algorithm.

Accuracy: The recognition accuracy a is defined as the portion of activity labels which are correctly predicted against those which are performed during the recognition delay. For example, for the recognized activity *drinking* and the activities performed during the recognition delay *making tea* and *drinking*, the recognition accuracy is then 0.5.

Utility: To trade off the accuracy and delay defined above, we propose a *utility* function which combines both metrics. We only use the time spent on waiting for data δ_w as the recognition delay as our preliminary results show that the waiting time often dominates (over 99% of) the total delay, and it is independent of the platform on which the recognition algorithm executes. We also use the number of sensing periods spent on waiting for data instead of time to exclude the potential uncertainties caused by different transmission protocols and network conditions.

For each recognized activity, given the recognition accuracy a and the waiting time δ_w, the utility u is defined as

$$u = a \cdot v(\delta_w)$$

where $v(\delta_w)$ is a function that defines the value of a recognition task with respect to its delay. Given two user-defined deadlines d_1 and d_2, where d_1 is for the deadline of the expected recognition delay and d_2 is for the deadline of the longest acceptable recognition delay, $v(\delta_w)$ is given by

$$v(\delta_w) = \begin{cases} 1, & \delta_w \leqslant d_1 \\ \dfrac{d_2 - \delta_w}{d_2 - d_1}, & d_1 < \delta_w < d_2 \\ 0, & \delta_w \geqslant d_2. \end{cases} \tag{5}$$

6.2. Experiment design

We evaluate the performance of our proposed system using the activity dataset collected in our previous work [5]. We built a real-time simulator to evaluate the performance of our system and compare it with a single-layer approach and the Hidden Markov Models (HMMs).

For the single-layer approach, we adapt our previous off-line algorithm [5] for on-line execution. For the HMM-based approach, we use the independent HMM model proposed in [20]. We build one HMM for each activity and compute the probability of each model over the input window, the one with the highest probability is output as a recognized activity.

6.3. Real-world dataset

The data collection was done by four volunteers in a smart home over two weeks. We select 26 activities as summarized in Table 1. Each subject was requested to perform these activities at his choice in an order which is close to his daily practice. Each subject followed his own step to perform each activity, in this way the data for each activity was collected naturalistically. We collected a total number of 532 activity instances, and only the sequential activity instances will be used for training.

6.4. Real-time simulator

We build a real-time simulator to simulate the operation of each sensor node, e.g., generation of continuous sensor data stream. There are a total number of six sensor nodes—three accelerometers, two RFID wristband readers and one location sensor. Each sensor node is simulated by a Java thread on a laptop computer. Each simulated sensor node is

Table 2
Templates of left hand gestures.

x : ⟨−216, −42, −36⟩ y : ⟨249, −48, 0⟩ z : ⟨815, 985, 988⟩
x : ⟨66, −95⟩ y : ⟨1008, 1001⟩ z : ⟨145, −82⟩
x : ⟨605, 455, 442, 389⟩ y : ⟨710, 555, 442, 442⟩ z : ⟨566, 782, 796, 719⟩
x : ⟨241, 92, 658⟩ y : ⟨−269, −395, −70⟩ z : ⟨862, 749, 717⟩
x : ⟨−141, −169⟩ y : ⟨736, 828⟩ z : ⟨−668, −562⟩
x : ⟨−922, −972⟩ y : ⟨283, 38⟩ z : ⟨284, 220⟩
x : ⟨80, −40, −87⟩ y : ⟨491, 665, 905⟩ z : ⟨852, 812, 580⟩
x : ⟨901, 879, 935⟩ y : ⟨146, −5, 44⟩ z : ⟨458, 316, 372⟩
x : ⟨871, 905, 880, 853, 802⟩ y : ⟨469, 489, 495, 565, 572⟩ z : ⟨−260, −260, −183, −255, −202⟩
x : ⟨−4, −21, −18, 82, −175, 7⟩ y : ⟨−896, −1103, −1054, −1168, −963, −1114⟩ z : ⟨96, 138, 133, 93, 52, 82⟩

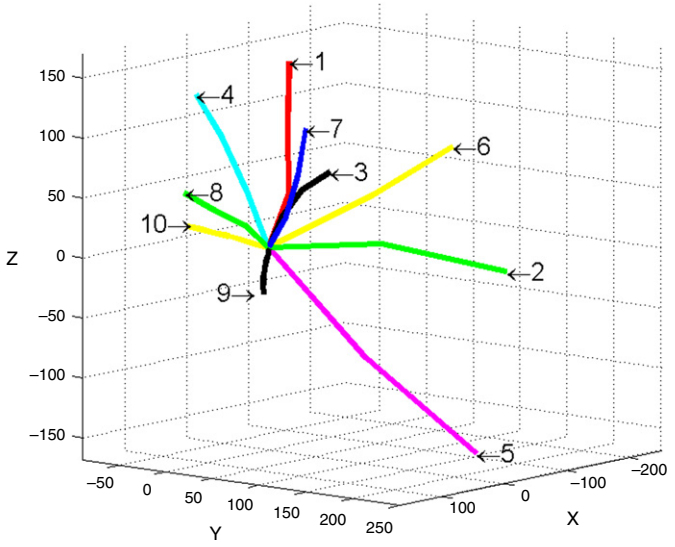

Fig. 4. Traces of templates for left hand gestures.

able to generate sensor readings at an adjustable sampling rate. We implemented the gesture recognition algorithm at each accelerometer sensor node using the simulator. The recognized gestures together with objects and locations will be transferred continuously through a wireless link to a real mobile device in which the EP-based, real-time recognition algorithm runs. The mobile device we use is HTC G7, with 1 GHz CPU and 512 MB RAM, running Android 2.2.

6.5. Gesture recognition performance

To evaluate the gesture recognition algorithm, Table 2 shows the gesture templates we discovered from the acceleration data stream of a subject's left hand using this clustering method. To visualize the gesture templates obtained in Table 2, we show the traces of the left-hand movements in a 3-D space in Fig. 4, assuming the initial position of the hand is at the origin of the coordinate system. Through 3-D visualization, it will be easily to figure out what each template represents in a physical world. These templates represent the different directions of left-hand movement in a physical space. Gestures 1, 3, 4 and 7 basically represent that the hand moves upward. By taking a closer look, gesture 1 represents *moving straight up*, gesture 7 represents *moving up and right*, gesture 4 represents *moving up and left* (i.e., opposite to gesture 7), and gesture 3 represents *moving up and forward*. Gestures 8 and 9 basically represent that the hand moves forward. While gesture 8 represents *moving forward and left*, gesture 9 represents *moving forward*. The rest of gestures are quite obvious, gesture 2 represents *moving right*, gesture 10 represents *moving left* and gesture 6 represents *moving back*. Gesture 5 represents *putting down* in which the hand movement patterns involve both *moving down* and *moving back*. It matches the natural pattern well since our arm is actually turning around the shoulder rather than going straight down when we put down our hands. Similarly for the right hand, as shown in Fig. 5, gestures 1 and 8 basically represent *moving up and left*. Gesture 7 represents *moving forward and left*. Gestures 9 and 10 represent *moving up and back*. Gesture 2 represents *moving back*. Gesture 5 represents *putting down*. Gesture 4 represents the hand movement of *moving up*. Gesture 3 represents *moving up and right*. Finally, Gesture 6 represents *moving up and forward*.

We obtain similar results for the templates of body gestures, as visualized in Fig. 6. Obviously, gestures 1–5 represent body *moving up, moving left, sitting down, moving forward* and *moving right*, respectively. It is interesting to analyze gesture

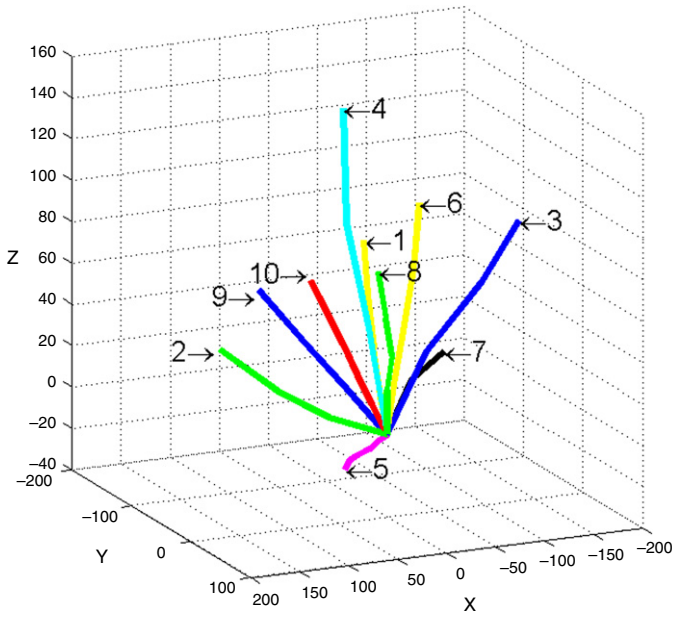

Fig. 5. Traces of templates for right hand gestures.

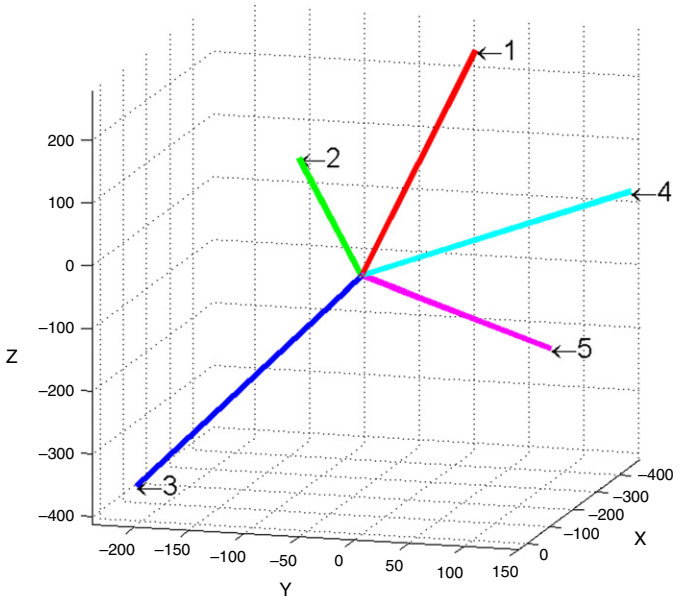

Fig. 6. Traces of templates for body gestures.

3 which involves two directions of body movement, i.e., both backward and downward. Such a movement pattern is likely to happen when we sit down. In that case, our body not only goes downward, but also goes backward when we lean our knees towards a chair.

6.6. Real-time performance evaluation

We evaluate the real-time performance of our proposed algorithm and compare it with both the EP-based single-layer approach and the HMM-based approach. Since the sliding-window size may affect both the recognition delay and accuracy, to draw a fair comparison, we first conduct extensive tests on different sliding-window sizes. We then compare the real-time performance of each approach by choosing its optimal window size. We define $d1 = 1$ and $d2 = 90$, implying that the recognition system should respond as fast as possible and making the user wait for over 90 s is unacceptable. A similar setting can be found in [21] for real-time physical activity recognition. We use ten-fold cross-validation for our evaluation.

Table 3
Real-time performances with optimal window size.

	EP two-layer	EP single-layer	HMMs
Optimal window size	1	1	2
Accuracy	82.87%	46.78%	46.41%
Delay	5.7	19.8	2.0
Utility	0.81	0.43	0.47

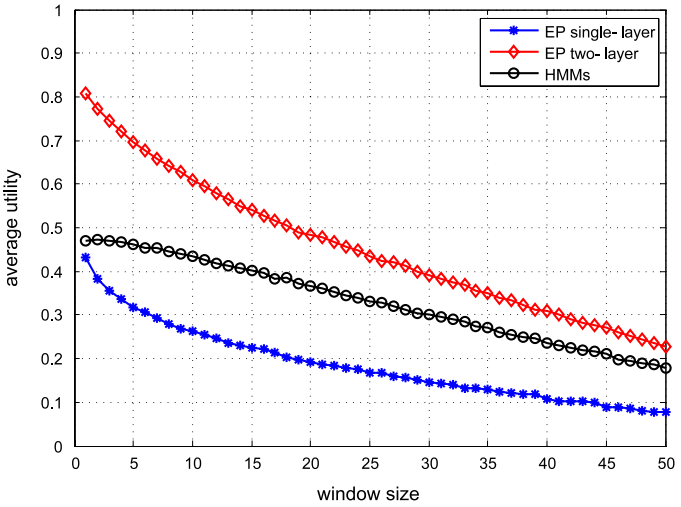

Fig. 7. Average utility gained under different window size.

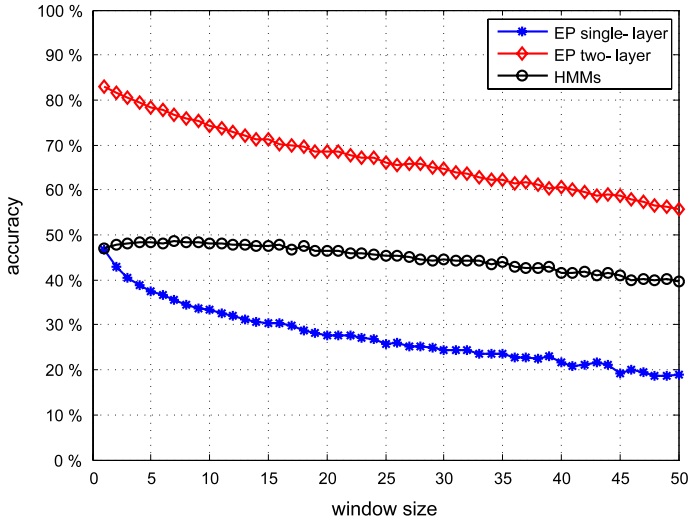

Fig. 8. Average recognition accuracy under different window size.

6.6.1. Choosing the optimal sliding-window size

We tested the performance of each algorithm by choosing the sliding-window size from 1 to 50 s. The average utility gained by each algorithm is shown in Fig. 7, the recognition accuracy is shown in Fig. 8, and the recognition delay is shown in Fig. 9.

From the figures we conclude that the optimal sliding-window size for both EP-based algorithms is one while the optimal window size for the HMM-based algorithm is two. The average recognition delay of our proposed algorithm is 5.7 sensing periods, the average recognition accuracy is 82.87%, and the average utility gained is 0.81. Table 3 summarizes the real-time performances of different algorithms by using the optimal window sizes. The results shows that the EP-based two-layer algorithm outperforms the other two algorithms by over 70% in utility gained.

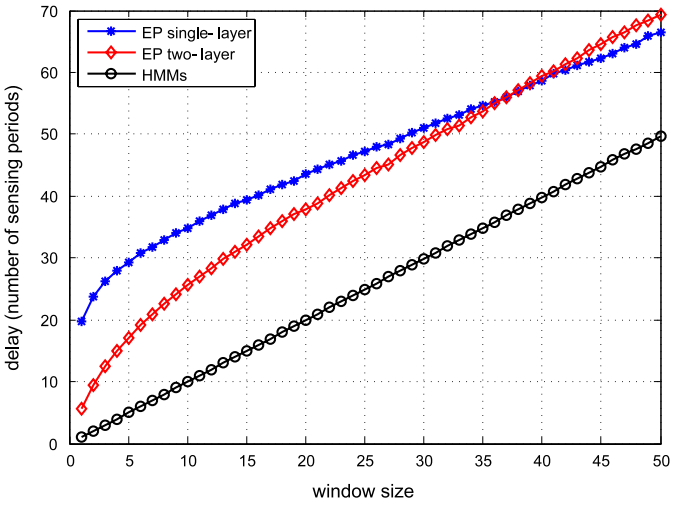

Fig. 9. Average recognition delay under different window size.

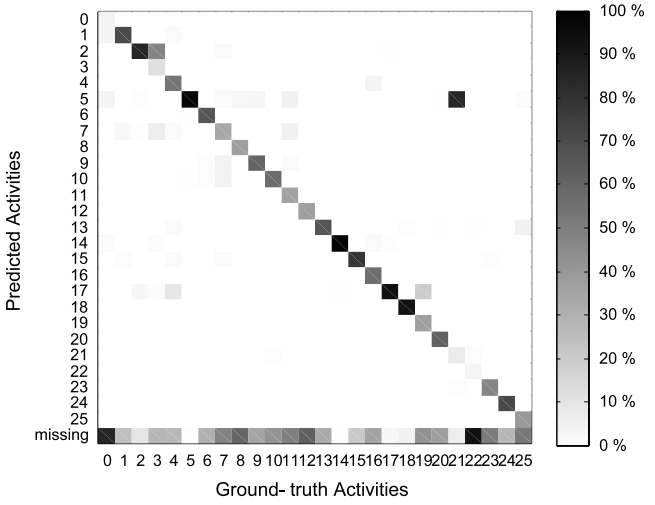

Fig. 10. Confusion matrix for the EP-based two-layer algorithm.

6.6.2. *Accuracy and delay analysis*

To reveal the accuracy of each activity, we present the confusion matrix as shown in Fig. 10. The columns show the predicted activities, and the rows show the ground-truth activities. The last row shows the percentage of instances that are miss detected. From the confusion matrix, we observe two main cases illustrated as follows.

Miss detection: This is the case which an activity instance performed by a subject is not detected. For example, for *making coffee* (i.e., activity 0 in the table), only 4.3% of the instances are correctly recognized while 84.8% of them are missed. This leads to a low accuracy for this activity. By analyzing the ground truth, we found that *making coffee* is often performed with another activity such as *making tea* or *using phone* in an interleaved manner. For example, a case in the dataset shows the phone rang in the middle of making coffee, and the subject then paused *making coffee* and went to pick up the phone. When the system recognizes the *using phone* activity, it clears the bitmap resulting in a loss of the initial data for *making coffee*. Hence, when the subject came back for making coffee again, the system may no long recognize it since some data are lost.

False detection: This is the case which an activity is recognized as another activity. For example, for *frying eggs*, 48.2% of the instances are recognized as *making oatmeal*. It probably can be explained as follows. These two activities share many common features, i.e., they are performed using similar objects, with similar gestures and in the same location. With the existing sensor features, it is difficult to discriminate these two activities. One possible solution is to make use of the sequence information of hand and body gestures and objects which we leave for our future work.

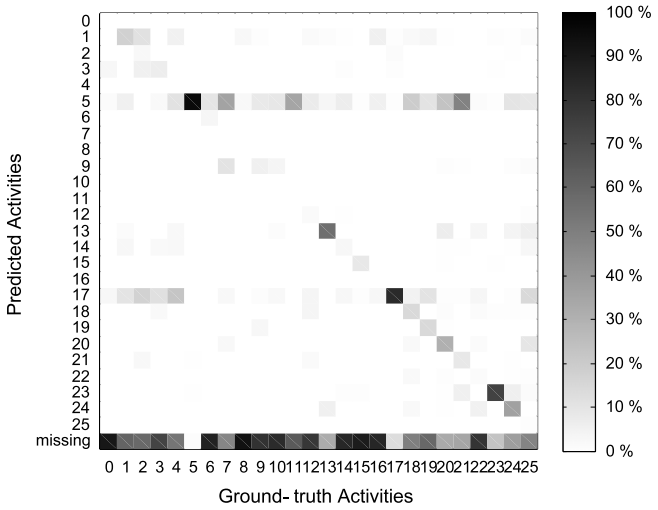

Fig. 11. Confusion matrix for the EP-based single-layer algorithm.

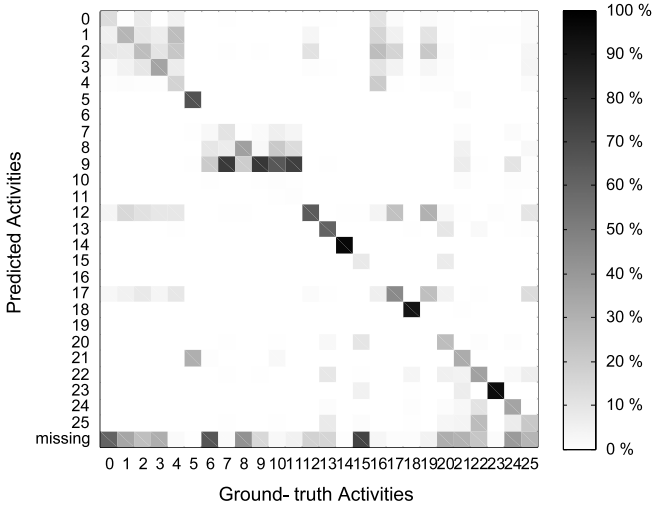

Fig. 12. Confusion matrix for the HMM-based algorithm.

We also present the confusion matrix for the EP-based single-layer approach (Fig. 11) and the HMM-based approach (Fig. 12). From the results we observe that the former one suffers from severe miss detection but has a higher accuracy on some activities such as *applying makeup* and *cleaning a dining table*; the latter one has a lower miss rate and also a lower accuracy.

We present the distribution of recognition delay for each algorithm in Fig. 13. For our proposed approach, about 83% of the predicted activities are recognized in one sensing period and over 90% of the predicted activities are recognized within 8 sensing periods. For the EP-based single-layer approach, only 38% of them are done in one sensing period and about 90% of them are recognized within 50 sensing periods. For the HMM-based approach, the algorithm is guaranteed to produce one activity for any given input data, hence its recognition delay is always two sensing periods. To this sense, the HMM-based approach outperforms our proposed approach and the EP-based single-layer approach. This result suggests that, our proposed approach sacrifices recognition delay for higher recognition accuracy. Compared with the other two methods, our algorithm still achieves the best real-time performance due to the highest utility gained.

6.7. Feature selection

In this experiment, we are interested to know how different features affect the recognition. We conduct a series of experiments by removing one of the features, and plot the result in Fig. 14. By removing the object and location the utility drops to 0.61 and 0.58 respectively, indicating that these two features are the two most important features. By removing all the gestures, the utility drops to 0.65 (approximately 19% of performance loss), which suggests the gestures also play an

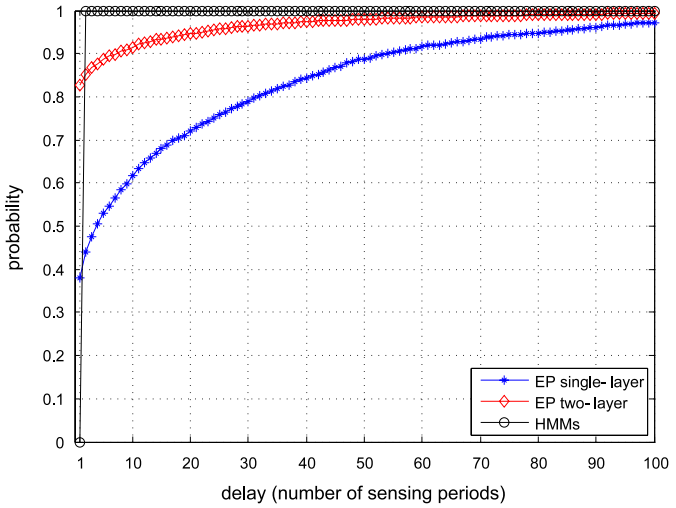

Fig. 13. The distribution of recognition delay.

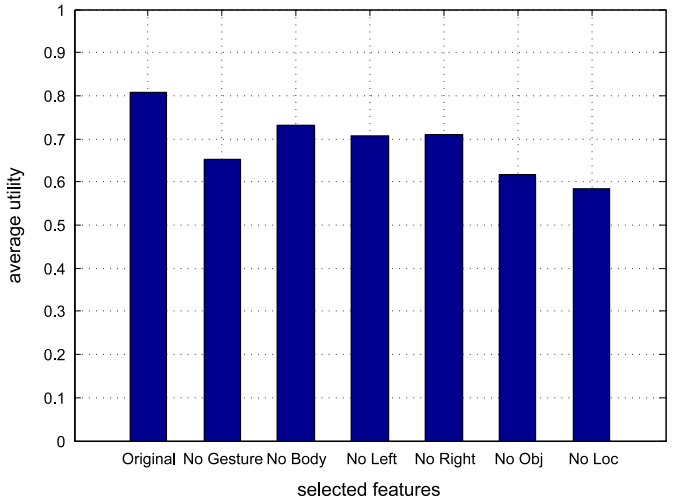

Fig. 14. Utility gained by different feature sets.

important role; similar results are observed when only one of the three gestures is removed, the utility is 0.73, 0.71, 0.71 by removing the body, left hand, and right hand gestures, respectively.

6.8. Real deployment and measurement

In this section, we report the measurement results from real deployment. The gesture recognition algorithm is implemented using C# and deployed on real IMOTE2 sensor nodes, while the high-level, complex activity recognition algorithm is implemented in Java and deployed on a smart phone as described in Section 6.4.

We first test the gesture recognition algorithm on the sensor nodes. By simulating different sampling rate of a 3-D accelerometer, Fig. 15 plots its execution time on IMOTE2 with different sampling rates. The blue dotted line shows the curve of a function that belongs to $\Theta(n^2)$. This result perfectly matches with our previous analysis on the time complexity in Section 4.4.

We also compare the execution time of our proposed approach with both the EP-based single-layer approach and the HMM-based approach. As illustrated in Fig. 16, when the sliding-window size increases from one vector to fifty vectors, the recognition time for the HMM-based method increases from 10 to 850 ms while the recognition time of EP-based approaches, both single- and two-layer, remains constantly below 10 ms. This may be explained as follows. The time complexity of HMMs is linear to the input data size while the time spent on matching the EPs with the items held in the bitmap remains constant, no matter how much input data are provided. This result brings an important feature for EP-based approaches, *the execution time of the recognition algorithm is constant given any size of input data*, which demonstrates the advantage of our EP-based approach for real-time activity recognition.

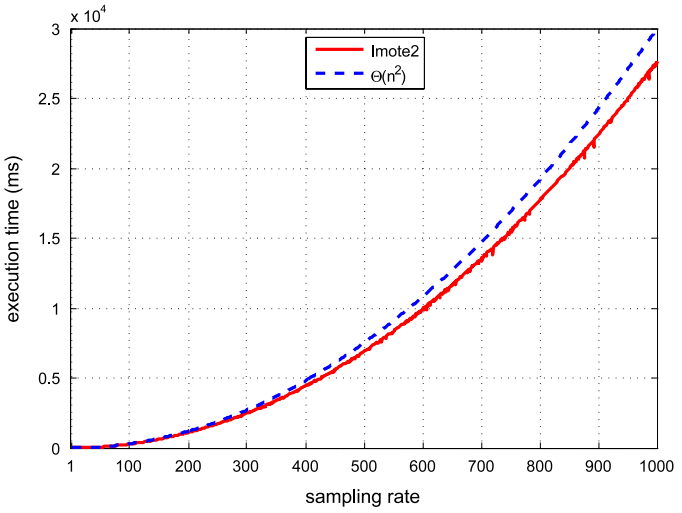

Fig. 15. Execution time of the gesture recognition algorithm.

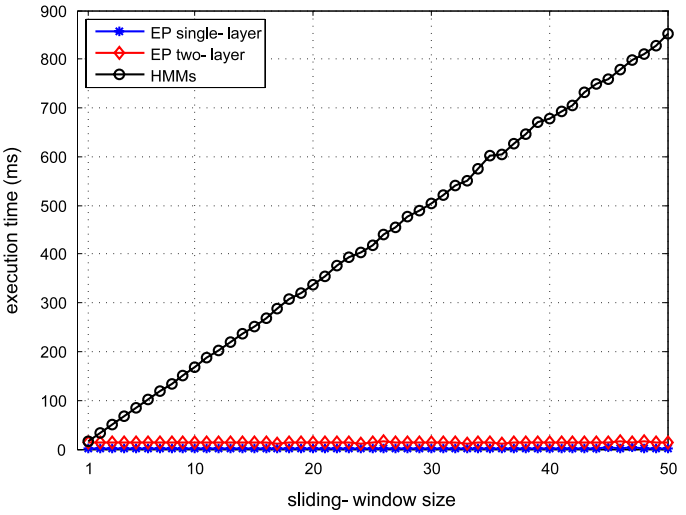

Fig. 16. Execution time of the EP-based two-layer recognition algorithm.

6.9. Communication cost analysis

In this experiment, we analyze the network communication cost for our hierarchical recognition model. We compare the recognition model with and without a hierarchical design. In a single layer model, all the sensor readings generated at each accelerometer node will be transferred over wireless links. The total amount of data transmitted on the network in one second can be computed as follows.

$$D = \sum_{i=1}^{n} \left\lceil \frac{f_i \cdot p_i}{m_i} \right\rceil \cdot (m_i + o_i) \tag{6}$$

where n is the number of sensor nodes, f_i is the sampling rate of the ith sensor node, m_i is the designed payload size for each packet, p_i is the size of each reading of the ith sensor and o_i is the overhead of sensor node i sending a packet. It is clear that $f_i \cdot p_i$ computes the total size of sensor readings of the ith sensor node that is to be transmitted for each second. By taking the ceiling of $\frac{f_i \cdot p_i}{m_i}$, we get the number of packets that is sent by the ith node in one second. Finally, by multiplying the number of the packets and the size of each packet, which can be easily computed by $m_i + o_i$, we get the total number of bits transmitted for sending the ith sensor node's data in one second. Finally, the data transmitted in the entire network in one second is the sum of the data transmitted for all sensor nodes.

We have three accelerometers, two RFID sensors and one location sensor in our system. ZigBee radio is used by the sensors for wireless data transmission. The packet header size for ZigBee/IEEE 802.15.4 protocol is 120 bits. Each accelerometer

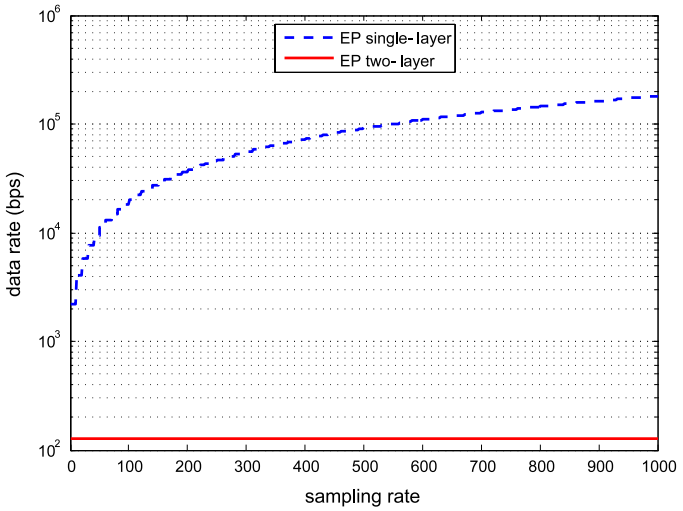

Fig. 17. Comparison of communication cost.

reading has a size of three 16-bit integers (i.e., on the three axes) which is 48 bits in total. The packet payload size is set to be 10 readings which is 480 bits. Each RFID sensor or location sensor has a sampling rate of 1 Hz. Each reading has a size of 64 bits, which is the size of one tag ID. The packet payload size is set to 64 bits. Thus, the total amount of data transmitted in the entire system in one second is 2352 bits according to Eq. (6).

In our hierarchical recognition model, we only need to transfer 1 byte of data containing a gesture label in every one second over wireless links since acceleration data are processed immediately by the gesture recognition algorithm. The total amount of data transmitted for each accelerometer in one second is 128 bits (8 bits for the gesture label and 120 bits for the packet header). While the RFID and location sensors remain the same, the total amount of bits transmitted in one second is reduced to 936 bits. Hence, we reduce the total communication cost by 60.2%. Through the above analysis, we demonstrate that a hierarchical recognition model is more appropriate for real-time activity recognition using a wireless sensor network which typically has a limited network bandwidth. Fig. 17 shows the comparison result of communication cost between the single-layer and two-layer approaches.

7. Conclusion

In conclusion, this paper proposes a real-time, hierarchical model based on a wireless body sensor network to recognize both physical, simple gestures and high-level, complex activities. At the sensor node level, acceleration data are processed immediately by a fast and lightweight gesture recognition algorithm for recognizing gestures. The recognized gestures, object and location information will be transferred to a centralized device, and then processed by an EP-based, real-time algorithm to recognize complex, high-level activities. Our experimental studies show the proposed system achieves not only good performance in accuracy and real-time recognition delay, but also better communication efficiency.

While our real-time, hierarchical model is promising, the entire system is still premature for real-life deployment. There are a number of limitations worth discussing. First, the data collection was still done in a mock scenario which did not reflect real-life situations. A more natural collection should be conducted in a real home. In addition, the current evaluation does not focus on subject independence. It is clear that in real life each individual may perform the same activity in different ways. Although in principle our system is able to be applied to any subject, it is interesting to study how the performance is affected by subject independence. We plan to extend our work in several directions. First, we will further develop our proposed algorithms to make a hard upper bound on the system's running time which guarantees the system's real-time performance. Second, we will deploy our system for real-life trials, and study its real-time behaviors and limitations in a real-life setting.

Acknowledgements

This work was supported by the Danish Council for Independent Research Natural Science under Grant 09-073281, National 973 program of China under Grant 2009CB320702, Natural Science Foundation of China under Grants 60736015, 60721002 and 61073031, and Jiangsu PanDeng Program under Grant BK2008017.

References

[1] W. Chen, D. Wei, X. Zhu, M. Uchida, S. Ding, M. Cohen, A mobile phone-based wearable vital signs monitoring system, in: The Fifth International Conference on Computer and Information Technology, 2005, CIT 2005, 2005, pp. 950–955.
[2] J. Nehmer, M. Becker, A. Karshmer, R. Lamm, Living assistance systems: an ambient intelligence approach, in: Proceedings of the 28th International Conference on Software Engineering, ACM, New York, NY, USA, 2006, pp. 43–50.

[3] R. Smith, N. Zane, F. Smoll, D. Coppel, Behavioral assessment in youth sports: coaching behaviors and children's attitudes, Medicine & Science in Sports & Exercise 15 (3) (1983) 208.
[4] H. Sakoe, S. Chiba, A dynamic programming approach to continuous speech recognition, in: Proc. 7th Int. Congress Acoust, Budapest, Hungary, Paper 20C-13, 1971.
[5] T. Gu, Z. Wu, X. Tao, H.K. Pung, J. Lu, Epsicar: an emerging patterns based approach to sequential, interleaved and concurrent activity recognition, in: Proc. of the 7th Annual IEEE International Conference on Pervasive Computing and Communications, Percom'09, Galveston, Texas, 2009.
[6] J. Modayil, T. Bai, H. Kautz, Improving the recognition of interleaved activities, in: Proc. Int'l Conf. Ubicomp, Seoul, South Korea, 2008.
[7] T. Wu, C. Lian, J. Hsu, Joint recognition of multiple concurrent activities using factorial conditional random fields, in: AAAI Workshop PAIR 2007, 2007.
[8] J. Kela, P. Korpipää, J. Mäntyjärvi, S. Kallio, G. Savino, L. Jozzo, S. Marca, Accelerometer-based gesture control for a design environment, Personal and Ubiquitous Computing 10 (5) (2006) 285–299.
[9] M. Philipose, K.P. Fishkin, M. Perkowitz, D.J. Patterson, D. Fox, H. Kautz, D. Hähnel, Inferring activities from interactions with objects, IEEE Pervasive Computing (2004).
[10] B. Logan, J. Healey, M. Philipose, E.M. Tapia, S. Intille, A long-term evaluation of sensing modalities for activity recognition, in: Proc. Int'l Conf. Ubicomp, Innsbruck, Austria, 2007.
[11] P.P. Palmes, H.K. Pung, T. Gu, W. Xue, S. Chen, Object relevance weight pattern mining for activity recognition and segmentation, Pervasive and Mobile Computing 6 (1) (2010) 43–57.
[12] E.M. Tapia, S. Intille, K. Larson, Real-time recognition of physical activities and their intensities using wireless accelerometers and a heart rate monitor, in: Proceedings of the 11th International Conference on Wearable Computers, Boston, MA, 2007.
[13] N.C. Krishnan, D. Colbry, C. Juillard, S. Panchanathan, Real time human activity recognition using tri-axial accelerometers, in: Proceedings of Sensors Signals and Information Processing Workshop, Sedona, AZ, 2008.
[14] J. He, H. Li, J. Tan, Real-time daily activity classification with wireless sensor networks using hidden Markov model, in: Proceedings of the 29th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, 22–26 August, 2007.
[15] N. Györbíró, Ákos Fábián, G. Hományi, An activity recognition system for mobile phones, Journal of Mobile Networks and Applications 14 (2009).
[16] J. Liu, Z. Wang, L. Zhong, J. Wickramasuriya, V. Vasudevan, UWave: accelerometer-based personalized gesture recognition and its applications, in: IEEE PerCom, 2009.
[17] G. Dong, J. Li, Efficient mining of emerging patterns: discovering trends and differences, in: Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, New York, NY, USA, 1999, pp. 43–52.
[18] J. Li, G. Liu, L. Wong, Mining statistically important equivalence classes and delta-discriminative emerging patterns, in: Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, New York, NY, USA, 2007, pp. 430–439.
[19] G. Dong, X. Zhang, L. Wong, J. Li, CAEP: classification by aggregating emerging patterns, Lecture Notes in Computer Science (1999) 30–42.
[20] D. Patterson, D. Fox, H. Kautz, M. Philipose, Fine-grained activity recognition by aggregating abstract object usage, in: Ninth IEEE International Symposium on Wearable Computers, 2005, Proceedings, 2005, pp. 44–51.
[21] E.M. Tapia, Using machine learning for real-time activity recognition and estimation of energy expenditure, Ph.D. Thesis, Massachusetts Institute of Technology, 2008.