Automatic Segmentation and Recognition in Body Sensor Networks Using a Hidden Markov Model

ERIC GUENTERBERG, HASSAN GHASEMZADEH, and ROOZBEH JAFARI,
University of Texas at Dallas

One important application of body sensor networks is action recognition. Action recognition often implicitly requires partitioning sensor data into intervals, then labeling the partitions according to the action that each represents or as a non-action. The temporal partitioning stage is called segmentation, and the labeling is called classification. While many effective methods exist for classification, segmentation remains problematic. We present a technique inspired by continuous speech recognition that combines segmentation and classification using hidden Markov models. This technique is distributed across several sensor nodes. We show the results of this technique and the bandwidth savings over full data transmission.

Categories and Subject Descriptors: I.5.1 [**Pattern Recognition**]: Models—*Statistical*; I.5.3 [**Pattern Recognition**]: Clustering—*Algorithms*; C.3 [**Computer Systems Organization**]: Special-Purpose and Application-Based Systems—*Real-time and embedded systems*

General Terms: Algorithms, Experimentation, Performance

Additional Key Words and Phrases: Body sensor networks, hidden Markov models, action recognition

ACM Reference Format:
Guenterberg, E., Ghasemzadeh, H., and Jafari, R. 2012. Automatic segmentation and recognition in body sensor networks using a hidden Markov model. ACM Trans. Embed. Comput. Syst. 11, S2, Article 46 (August 2012), 19 pages.
DOI = 10.1145/2331147.2331156 http://doi.acm.org/10.1145/2331147.2331156

1. INTRODUCTION

The capabilities of small electronic devices have been increasing exponentially as their sizes and prices have dropped. Uses that may have seemed frivolous or expensive are becoming practical and even cheap. For instance, cell phones can now record videos and images and transmit them wirelessly to personal websites in real time, and cars can automatically notify paramedics of a crash. One exciting platform with similar potential is the body sensor network (BSN) in which several intelligent sensing devices are placed on the human body and can perform collaborative sensing and signal processing for various applications.

Currently, these sensing devices are large enough that they are too cumbersome for casual use. However, the threshold for wearability depends on the application. For instance, stride variability is associated with the occurrence of Alzheimer's [Hausdorff et al. 1998; Sheridan et al. 2003]. If a patient could wear a sensor on his leg that could help a doctor evaluate the effectiveness of medication in a naturalistic setting, the

Author's address: R. Jafari, ESSP Lab, University of Texas at Dallas, 800 W. Campbell Rd, MS EC33, Richardson, TX 75080; email: rjafari@utdallas.edu.

DOI 10.1145/2331147.2331156 http://doi.acm.org/10.1145/2331147.2331156

inconvenience might be worth it. Further, these devices are getting smaller and more powerful every year, so wearability is unlikely to remain a long-term problem.

Therefore, now is the time to investigate applications so that hardware designers can optimize their devices for more useful applications. One use of BSNs is action recognition in which the actions of the person wearing the sensors are identified. This has several applications. For instance, techniques exist for extracting stride variability, but the output is only correct if the person is walking [Aminian et al. 2002]. Also, action recognition could be used to develop an activity log for a person to help him or doctors assess health [Choudhury et al. 2008; Nait-Charif and McKenna 2004; Ouchi et al. 2004] or avoid dangerous actions, which might be useful for RSI sufferers. Action recognition could even be used to help provide contextual interfaces to other devices [Castelli et al. 2007].

Recognition is a two-step process: (1) locate temporal regions that contain a single action, and (2) recognize the action. Both these problems are well studied for image and motion capture systems. However, many of these techniques are dependent on spatial information provided by these systems and are not appropriate to resource-constrained environments. This has led BSN researchers to adapt techniques from more basic pattern-recognition approaches and from speech recognition. Much of the work in the BSN community has focused on the recognition portion, while segmentation in BSN recognition is still largely an open problem. Popular approaches include segmenting on fixed time slices, manual segmentation, and exhaustive search. Fixed time slices may capture multiple movements or only part of movements, manual segmentation is impractical for a deployed system, and exhaustive techniques are resource intensive.

In previous work, we looked at segmentation using an efficient energy-based approach [Guenterberg et al. 2009]. We found that the method was effective for actions separated by rests. However, in a non-lab setting, actions often occur in rapid succession with no rest between them. This problem is called continuous action recognition and is addressed by applying a hidden Markov model (HMM)-based approach adapted from the speech recognition community. The HMM can both segment and recognize simultaneously. The novelty of our work is a system which (a) provides continuous action recognition from inertial sensor data, (b) uses unsupervised clustering on a per-node basis to reduce the communication from each node, and (c) provides postural information to eliminate impossible actions (e.g., walking from a sitting position). The hidden Markov model allows full signal segmentation of continuous actions with a low computational-order algorithm. To the best of our knowledge, this capability is not addressed in the literature. Since the sole goal of segmentation is to facilitate recognition, the quality of segmentation will be judged entirely by presenting recognition accuracy. Further, the results will be compared to k-NN classifier provided with manual segmentation as a conceptual upper bound.

2. RELATED WORK

Several approaches to action recognition have been proposed. A common problem is not the recognition but the segmentation of data into actions. Often in image recognition this is done without specific knowledge of what the image contains, but for action recognition using inertial sensors, it is generally not possible to infer a segmentation without some knowledge of what is being segmented. Various approaches address this problem in different ways.

Ward et al. recognized several workshop activities, such as taking wood out of a drawer, putting it into the vice, getting out a hammer, and more. They avoided the problem of segmenting accelerometer data by segmenting the data using the presence or absence of sound and then identified the action using accelerometer data and an

HMM classifier [Ward et al. 2006]. Their results showed the effectiveness of this technique for a shop, but in many other situations, actions are not correlated with sounds.

Another approach used a k-NN classifier and several statistical features to classify actions using a minimum number of sensor nodes [Ghasemzadeh et al. 2008]. Manual segmentation was used to avoid introducing errors from segmentation. This is a good technique for isolating the performance of various parts of a system, but for a deployed system, a satisfactory segmentation scheme is necessary.

Alternatively, it is possible to try a number of segmentations and choose the best. Lv and Nevatia [2006] use 3-D motion capture data. Given a start and end time, each joint uses an HMM to identify the action. AdaBoost is then used to make a global decision using the HMMs as weak classifiers. A dynamic programming algorithm chooses the best segmentation according to their maximum-likelihood function in $O(T^3)$ time. This scheme performs well if all computation is done on a single machine, but when each HMM is employed on a separate sensor node, the communication overhead required to try the different segmentations is quite high.

Several authors classify fixed-size segments independently of each other [Bao and Intille 2004]. This can result in outliers and discontinuities. Many methods involve some sort of smoothing function [Bao 2003; Courses et al. 2008; Van Laerhoven and Gellersen 2004]. One such method uses AdaBoost to enhance several single-feature weak classifiers. An HMM uses the confidence output of the AdaBoost classifier as input. A separate HMM is trained for each action class, and the overall segmentation/classification is chosen based on the maximum likelihood among the various HMMs [Lester et al. 2005].

This is somewhat similar to our approach, except our model is based on a single HMM, which allows us to rule out impossible sequences of actions and to avoid outliers that could result from one model temporarily having higher probability than the others. The main contribution of our algorithm is efficiently producing a segmentation and classification and performing this processing on a distributed platform.

Quwaider and Biswas [2008] divide actions, which they refer to as postures, based on the activity level measured with accelerometers. With high-activity postures, such as running, the postures are identified based on the energy level on each limb. For relatively quiet postures, such as sitting and standing, they employ a hidden Markov model used on radio signal strength (RSSI) differences between sensor nodes. With this, they can differentiate between sitting and standing postures. Our technique also identifies postures as key to recognizing actions, but we use an inertial sensor approach to achieve recognition and explicitly model actions as sequences of motions. This allows us to differentiate between actions that are different but may have a similar level of activity, such as turning counter clockwise and turning clockwise.

Another HMM-based segmentation technique from our research group subdivides a single action [Guenterberg et al. 2009]. This method is able to find the time of certain key events within a known and possibly repeating action but is unable to determine the action. The present article describes a method for segmenting and identifying actions from a stream of sensor data. Therefore, these works are complementary but different in terms of both goals and methods. The method described in Guenterberg et al. [2009] is based on a left-right HMM and is able to determine the time when certain events occur within an action, such as heel touch and toe raise during walking. While both that work and the present work rely on HMMs to process inertial data of human movement, the present work uses a unique state structure built from posture states and independently trained left-right models for each movement. Further, we introduce the concept of a node-level unsupervised clustering technique, called motion transcripts, to reduce communication load and model order. Finally, this work uses data from multiple sensor nodes for most action recognition problems, while the

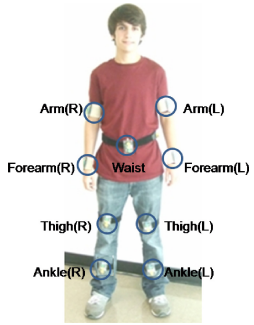


Fig. 1. Sensor placement.

Fig. 2. Sensor node.

previous work was aimed at data from a single sensor location (even though the possibility of multiple locations was explored).

3. DATA COLLECTION HARDWARE

This article presents a scenario in which the actions a subject performs are identified from continuous data provided by a BSN. The sensor nodes are embedded computing and sensing platforms with inertial sensors, wireless communication capability, a battery, and limited processing capabilities. Sensor nodes must be placed at multiple locations on the body to capture sufficient information to accurately determine the action. For instance, the action *placing something on a shelf* and *standing still* produce similar sensor data on the leg but different data on the arms, while *turning to look behind* and *turn* 90° show similarities from the shoulder but differences from the legs. The nodes communicate with a base station where a final conclusion is reached.

3.1. Sensing Hardware and Body Placement

Figure 2 shows one of the sensor nodes used to collect data for this article. The sensor nodes use the commercially available TelosB mote with a custom-designed sensor board and are powered by two AA batteries. The processor is a 16-bit, 4 MHz TI MSP430. The sensor board includes a tri-axial accelerometer and a bi-axial gyroscope. Data is collected from each sensor at 20 Hz. This frequency was chosen empirically as a compromise between sampling rate and packet loss.

The sensor nodes are placed on the body, as shown in Figure 1. Placement was chosen so that each major body segment is monitored with a sensor. While we expect that nodes placed at a subset of these locations would be sufficient for accurate classification of all considered actions, no formal procedure was performed to select such a reduced set. Discovering such procedures could prove to be a fertile area for future research.

3.2. Constraints and Deployment Architecture

The goal of this research was to find a computationally realistic algorithm for segmenting and classifying actions. Actually implementing this technique on sensor nodes is the subject of future research. To this end, data collected on each sensor node was broadcast to a base station and recorded for later processing in the MATLAB environment. This gave us the most flexibility for developing and testing different signal processing and classification schemes.

The algorithms presented here assume the following deployment architecture: the sensor nodes are placed on the body, as shown in Figure 1. Each can communicate

Table I. Actions Captured

ID	Initial Posture	Action	Final Posture
1	Stand	Stand to Sit (Armchair)	Sit
2	Sit	Sit to Stand (Armchair)	Stand
3	Stand	Stand to Sit (Dining Chair)	Sit
4	Sit	Sit to Stand (Dining Chair)	Stand
5	Sit	Sit to Lie	Lie
6	Lie	Lie to Sit	Sit
7	Stand	Bend and Grasp from Ground (R Hand)	Stand
8	Stand	Bend and Grasp from Ground (L Hand)	Stand
9	Stand	Bend and Grasp from Coffee Table (R Hand)	Stand
10	Stand	Bend and Grasp from Coffee Table (L Hand)	Stand
11a	Stand	Turn Clockwise 90°	Stand
11b	Stand	Return from 11a	Stand
12a	Stand	Turn Counterclockwise 90°	Stand
12b	Stand	Return from 12a	Stand
13	Stand	Look Back Clockwise and Return	Stand
14	Stand	Look Back Counterclockwise and Return	Stand
15a	Stand	Kneeling (R Leg First)	Kneel
15b	Kneel	Return from 15a	Stand
16a	Stand	Kneeling (L Leg First)	Kneel
16b	Kneel	Return from 16a	Stand
17a	Stand	Move Forward 1 Step (R leg)	Stand
17b	Stand	Move L Leg beside R Leg	Stand
18a	Stand	Move Forward 1 Step (L Leg)	Stand
18b	Stand	Move R Leg beside L Leg	Stand
19a	Stand	Reach up to Cabinet (R Hand)	Stand
19b	Stand	Return from 19a	Stand
20a	Stand	Reach up to Cabinet (L Hand)	Stand
20b	Stand	Return from 20a	Stand
21a	Stand	Reach up to Cabinet (Both Hands)	Stand
21b	Stand	Return from 21a	Stand
22	Stand	Grasp an Object (1 Hand), Turn 90° and Release	Stand
23	Stand	Grasp an Object (Both Hands), Turn 90° and Release	Stand
24	Stand	Turn Clockwise 360°	Stand
25	Stand	Turn Counterclockwise 360°	Stand

directly with the base station. The nodes have a limited power supply and must last a long time between recharges, so power must be conserved. The base station is a cell phone or PDA that has greater processing capabilities and can use significantly more power. Wherever the final classification occurs, it must be transmitted to the base station for storage or long-range communication.

Communication uses significantly more power than processing [Akyildiz et al. 2002; Polastre et al. 2005], so limiting communication is key to conserving power. Also, while the base station is more powerful than a sensor node, it is not as powerful as desktop computers, so algorithms designed to run on the base station should be of low computational order.

3.3. Actions Collected

The actions considered are mostly transitional actions. Each starts and ends with a posture. The actions are shown in Table I. Not all postures sharing the same label are exactly the same. For instance, for *stand* at the end of actions 19a, 20a, and 21a, one or both hands are on a shelf, whereas for most other cases *stand* means standing with hands resting at the side. The probabilistic nature of HMMs allows both postures to be represented by the same state.

The actions were collected from three subjects who each performed the action ten times. These actions were manually segmented to label the start and the end of each action. This manual segmentation is used to create sequences of known actions to train the model and represents the ground truth. These are referred to as canonical annotations. The data is divided into a training set and a testing set, with approximately half the trials used for training and half for testing. The testing data has also been manually annotated to allow for comparison between the segmentation and labeling automatically generated by our system and the ground truth (manual segmentation).

4. CLASSIFICATION MODEL

One of the most difficult problems in classification is trying to label all the actions in a continuous stream of data in which both the timing of actions and the labels are unknown. This problem has been considered many times in speech recognition and is called continuous speech recognition [Rabiner and Juang 1986]. The problem of speech recognition is similar enough to action recognition that many techniques used for speech recognition can be applied with appropriate modifications to action recognition tasks. Jurafsky et al. [2000] present a model based on hidden Markov models (HMMs) where each word is represented by a separate left-right HMM.[1] These are combined into a single HMM by creating a null state which generates no output. Each word starts from this null state and ends on the null state. A very similar approach is used for gesture recognition from hand sensors in Lee and Kim [1999].

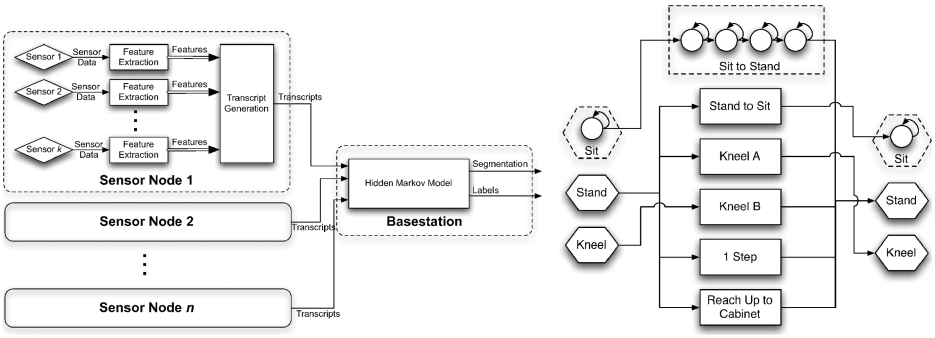
We took this model and adapted it to fit within the constraints imposed by our BSN configuration and to more effectively solve the problem of action recognition. One particular change is that each action is assumed to start with some posture such as *kneeling* or *standing* and end with a posture. The postures are not null states, that is, there is an output associated with a posture, and a posture may persist for a period of time. The input for this system is a subject performing movements. These movements can be in an arbitrary order. The output is a segmentation and a set of labels for each segment.

4.1. Overview

Classification requires a number of signal processing steps that execute on the sensor nodes and the base station, as shown in Figure 3(a). The system is designed to accurately classify actions with limited communication and use of processing power. The data is processed on a moving window centered on the current sample. The window moves forward one sample at a time.

(1) *Sensor Data.* Data from five sensors is collected. The accelerometer senses three axes of acceleration: a_x, a_y, and a_z. The gyroscope senses angular velocity in two axes: θ and ϕ. There is no angular velocity from the axis orthogonal to the plane of the sensor board.
(2) *Feature Extraction.* For each sample time, a feature vector is generated. The following features are extracted from a five-sample window for each sensor: *mean*, *standard deviation*, *rms*, *first derivative*, and *second derivative*.

[1]Hidden Markov models assume a process starting in a state which, at each discrete time, generates an output and then transitions to a new state. The state transition and output are probabilistic and based exclusively on the current state. The output can be observed but not the state. Algorithms exist to train a model to a given set of output sequences and to infer the state sequence given an output sequence and model. A left-right model restricts transitions to self transitions or the next state in sequence. See Rabiner and Juang [1986] for more information.



(a) Signal processing model. (b) HMM for action recognition.

Fig. 3. Signal processing and recognition models for continuous action recognition.

(3) *Transcript Generation*. Instead of transmitting the feature vector to the base station, each sensor node labels a sample using a single character from a small alphabet. Each sensor has a unique alphabet with between two to nine characters. Characters often repeat for several samples allowing for significant compression. The sequence of labels produced are *motion transcripts*. Transcript generation uses Gaussian mixture models (GMM) to label samples based on clusters generated from the training data.
(4) *Hidden Markov Model*. The HMM uses the model shown in Figure 3(b). In the middle are actions which are modeled as left-right HMMs with between $M_w = 1, 2, \cdots, 10$ states. The postures on the left and right are each modeled using a single state. The duplicated postures represent the same state. Postures and actions are connected as shown to form a single HMM.
(5) *Generating Output.* When trying to segment and classify data, the Viterbi algorithm [Rabiner and Juang 1986] is used to find the most likely state sequence for the given output. The Viterbi is used because it finds the optimal sequence efficiently. Each sample is labeled with the name of the action or posture of the associated state. This output is the *generated annotations*.

4.2. Transcript Generation

Reducing data before transmission can save considerable power in a BSN. Transcripts do this by reducing the multidimensional per-sample observations on a sensor node to a single character taken from a small alphabet. Transcripts are inspired by the idea that actions can be represented by a sequence of motions. Motions can be identified from a small interval of observations. A single motion or position is likely to persist for some time, allowing run-length encoding to further reduce transmitted data.

We have no canonical list of motions; therefore, a technique is needed that does not require human input. One solution is unsupervised clustering which automatically groups points based on underlying patterns in the data. Once these groups are created from training data, later observations may be assigned to one of the existing groups. In our system, the *points* are feature vectors in F-dimensional space.

The most common clustering techniques include hierarchical clustering [Johnson 1967], k-means clustering [Hartigan and Wong 1979], and model-based clustering [Figueiredo and Jain 2002; Fraley and Raftery 1998]. Model-based clustering assumes that all points have been generated from a set of distributions. The Gaussian mixture model (GMM) is a model-based clustering using Gaussian distributions. Many realistic processes actually generate output based on Gaussian distributions, and many

more can be approximated by a small number of Gaussian distributions. This causes GMMs to often outperform other methods of clustering [Fraley and Raftery 1998]. For these reasons, GMMs are used for transcript generation. We assume a diagonal covariance matrix to reduce computational complexity for labeling and to alleviate errors in estimation due to the small sample size problem [Raudys and Jain 1991].

There is an independent transcript, $\mathcal{T}_i$, for every node. Each frame of data from the node can be labeled with one of the C_i symbols, each of which is represented by a GMM, λ_j. λ_j has M_j Gaussian distributions. As shown in Equation (1), each mixture is represented by three parameters, that is, the mixing parameters p, which represent the prior probability of the distribution generating a given point. μ and Σ are the Gaussian mean and covariance matrices.

$$\lambda = \left\{ (p_1, \mu_1, \Sigma_1), \cdots, \left(p_{M_j}, \mu_{M_j}, \Sigma_{M_j}\right) \right\}. \tag{1}$$

The probability that a given observation x was generated by the mixture λ_j is

$$p(x|\lambda) = \sum_{i=1}^{M_j} p_i p(x|\mu_i, \Sigma_i). \tag{2}$$

Training a set of clusters involves choosing clusters which best model the training data while having a low enough order to avoid overfitting to the training set at the expense of good generalization. The maximum likelihood (ML) criterion for the best cluster model for a given set of observed points $x \in X$ is

$$\lambda_{\max} = \arg \max_{\lambda=\lambda_1}^{\lambda_{C_i}} P(X|\lambda) = \arg \max_{\lambda=\lambda_1}^{\lambda_{C_i}} \prod_{t=1}^{T} p(x_t|\lambda). \tag{3}$$

This cannot be analytically calculated, so frequently, the expectation maximization (EM) procedure is used [Figueiredo and Jain 2002; Reynolds and Rose 1995]. This iteratively converges to a locally optimal clustering. EM starts with an initial solution then iteratitively improves the solution with the following steps.

(1) *Expectation.* For each point and mixture component, calculate the probability that the point was generated by the component's distribution.
(2) *Maximize.* Update the model parameters to maximize the likelihood function using the membership probabilities calculated in the preceding expectation step.

The initial model and the choice of the number of mixtures (M) affect the final quality of the clusters. A common method of choosing M is to use EM to train several models using different values of M and starting distributions. Then the models can be compared using some measure, and the best is selected. We use the Bayesian information criterion to compare models [Fraley and Raftery 1998]. The clustering is performed independently on each node, and each node may have a different number of clusters.

4.2.1. Assigning New Observations to Clusters. Transcripts for an observed action are generated by assigning each sample to a single cluster. These cluster assignments are used to label each sample, resulting in a string from a finite alphabet representing the motion, as observed from a single sensor node. Using Equation (4), a feature vector for a given sample is assigned the cluster most likely to have generated it.

$$c(x) = \arg \max_{i=1}^{M} p_i p(x|\mu_i, \Sigma_i). \tag{4}$$

4.2.2. Implementation on Sensor Nodes. Cluster assignment in a deployed system will run on a sensor node, so efficiency is very important. One way to achieve this is to use log probabilities to assign the clusters. Specifically, by extending Equation (4), the following relationship is achieved.

$$c(x) = \arg\max_{i=1}^{M} p_i p(x|\mu_i, \Sigma_i) \tag{5}$$

$$= \arg\max_{i=1}^{M} \left[\log|p_i| + \log\left|p(x|\mu_i, \Sigma_i)\right|\right] \tag{6}$$

$$= \arg\max_{i=1}^{M} \left[\log|p_i| + \log|\alpha| - \frac{1}{2}\sum_{k=1}^{F}\frac{(x_{ik} - \mu_{ik})^2}{\sigma_{ik}^2}\right]. \tag{7}$$

Equation (6) follows from the logarithmic function being monotonic and strictly increasing. The expansion of the log Gaussian probability in Equation (7) as a summation is the consequence of using a diagonal covariance matrix. The variable $\alpha = \frac{1}{2\pi^{F/2}|\Sigma|^{1/2}}$ and represents the normalization factor of the distribution.

Furthermore, given the lack of a floating point unit on most sensor node hardware, calculations should be done with fixed point arithmetic.

4.3. Hidden Markov Model

After each sensor node assigns a character to each sample, the actions can be determined on the base station using the HMM shown in Figure 3(b).

4.3.1. Training the Model. An HMM has M states and is defined by the model λ, consisting of three sets of probabilities.

$$\lambda = \{\pi_i, a_{ij}, b_j(k)\}. \tag{8}$$

The probability that a sequence begins with state s_i is π_i. The transition probability a_{ij} is the probability that a state transitions to state s_j after starting on state s_i. For discrete observations, $b_j(k)$ gives the probability that observation v_k is emitted at state s_j.

For our system, the left-right model for each action and posture is trained independently, then all actions and postures are joined to form a single HMM. The model for each action is trained by starting with an initial model and iteratively improving it using the Baum-Welch procedure [Rabiner and Juang 1986]. The Baum-Welch procedure finds a local minimum. By trying a number of variations and selecting the best model according to some measure, the likelihood of finding a global minimum is increased. As with GMMs, a common technique for model selection is BIC [Biem 2003; Stoica and Selen 2004]. We try $M_w \in 1, 2, \cdots, 10$. For each of these fixed-state models, we start by dividing samples in each sequence evenly among states, then iterating the EM algorithm five times to converge the model. For the next nine tries for the current M_w, each state is initially assigned a random number of samples. The best of these 100 models is used to represent the action.

Each component of the HMM model in Equation (8) must be trained. Since the action must start at the first state, m_1,

$$\pi_i = \begin{cases} 1 & \text{if } i = 1 \\ 0 & \text{otherwise.} \end{cases} \tag{9}$$

a_{ij} and $b_j(k)$ are trained using the Baum-Welch algorithm, as described in Rabiner and Juang [1986]. The Baum-Welch algorithm is another EM algorithm. At each step, the state sequence probabilities are computed, and this is used to update a_{ij} and $b_j(k)$

based on the expected transitions and observations corresponding to each state, as seen in Equations (10) and (11).

$$a_{ij} = \frac{E\left[\text{number of transitions from } s_i \text{ to } s_j\right]}{E\left[\text{number of transitions from } s_i\right]}. \tag{10}$$

$$b_j^{(f)}(k) = \frac{E\left[\text{number of times in } s_j \text{ and observed symbol } v_k\right]}{E\left[\text{number of times in } s_j\right]}. \tag{11}$$

The observation probabilities for each sensor node are considered independent of other nodes f. This means that $b^{(f)}(k)_j$ is computed for each sensor node separately, and the overall observation probability is.

$$b_j(k) = \sqrt[M_w]{\prod_{f \in F} b_j^{(f)}(k)}. \tag{12}$$

In speech recognition, a scaling factor, called the language model scaling factor (LMSF) is used to compensate for an incorrect assumption of independence [Wessel et al. 1998] between observations. We adapt this concept by taking the M_wth root of the observation probabilities.

Right after the action finishes, it is expected to immediately transition to the proceeding posture. Therefore, during training, only state sequences ending in the final state, s_{M_w} should be considered. This can be accomplished simply if the observation probability takes the sample number into account and makes the probability 0 for all observations from a state other than the final state for the final observation in a given sequence.

$$b_j'(k,t) = \begin{cases} 0 & \text{if} j \neq M_w \text{ and } t = T \\ b_j(k) & \text{otherwise.} \end{cases} \tag{13}$$

4.3.2. Bayesian Information Criterion. Selecting the best among several models is a common problem in statistical pattern recognition. In general, a model with a greater number of parameters will better fit any set of training data but runs the risk of fitting eccentricities in the training data not present in the test data. The Bayesian information criterion [Biem 2003] is a method that only requires training data and has strong probabilistic properties.

$$BIC(\lambda) = \log p\left(X|\lambda, \hat{\theta}\right) - \alpha \frac{K}{2} \log T. \tag{14}$$

For HMMs, $p(X|\lambda, \hat{\theta})$ can be computed in a straightforward manner, as given in Rabiner and Juang [1986]. T is the total number of samples in the training set, and α is a regularizing term that allows for control over the overfitting penalty. We chose

a value of $\alpha = 0.05$ in which each action was represented with an average of three states. K, the number of free parameters, is

$$K = 2(M-1) + M \sum_{i=1}^{n_{nodes}} c_i - 1. \tag{15}$$

M is the total number of states, and c_i is the number of clusters on the given sensor node. The first term represents the number of transition probabilities, and the second the number of emission probabilities.

4.4. Joining Models

The models for each action are joined into a single HMM. Posture self-transition probabilities are derived from the training data, while the probability of transition from a posture to all associated actions are considered to be equal probability.

4.5. Sequence Extraction for Classification

After the model is fully deployed, it will be deployed on a BSN. The clustering operates on individual nodes, and the cluster sequences are transmitted to the base station, where the HMM is used to segment and classify the actions. For this, the most likely state sequence is extracted using the Viterbi algorithm [Rabiner and Juang 1986]. The states in the sequence are grouped by action; the individual state progression within an action is considered unimportant. The model dictates that each occurrence of an action is separated by at least one posture state. This means that even if an action were to repeat many times, the number of repetitions can be easily counted.

4.6. Runtime Order and Comparison to Other Methods

At runtime, there are two primary stages: transcript generation and HMM-based recognition. For transcripts, the probability that the feature vector at each sample was generated by each cluster is calculated. The sample receives the label of the highest probability cluster. Because a diagonal covariance matrix is used, calculating probabilities is a linear function of the number of features.

$$O(\text{Clustering}) = O(T \cdot F \cdot C_i), \tag{16}$$

where T is the number of samples considered, F is the number of features in the feature vector, and C_i is the number of clusters on sensor node i.

The HMM uses the Viterbi algorithm [Rabiner and Juang 1986] for classification and segmentation. Since the HMM consists of several joined left-right models, the simplified Viterbi runs more efficiently.

$$O(\text{Classifier}) = O(T \cdot 2M_a \cdot K). \tag{17}$$

In Equation (17), the total number of action states is M_a, and K is the number of sensor nodes.

This method must be compared to other methods that segment and classify data and cannot be compared with methods that rely on external data to segment the data. The method in Lv and Nevatia [2006] is $O(T^3)$ and so has a lower runtime efficiency: the approach is not designed for sensor networks and thus implies a large number of transmissions. Yang et al. [2009] propose a system that can be either used on fixed segments or can adaptively choose a segmentation. Both are linear with respect to the number of samples, however, there are large constant factors, such as the number of length hypotheses, and repeated computation of an inverse matrix for each hypothesis.

Finally, there are a number of methods based on fixed segmentation. These also can do no better than linear time. However, the associated constant factors may be smaller

Table II. Results for All Subjects Trained on One Model

Subject	*k*-NN	Full Samples Accuracy	Clustering Accuracy
1	96.5	78.1%	92.6%
2	93.2	38.1%	82.6%
3	99.4	10.9%	83.2%

than for our model. These methods do not take temporal characteristics into account, so actions that use the same motions for a portion of time will be indistinguishable, even if the sequence is unique for each action.

5. RESULTS

For our experiment, three subjects performed the actions listed in Table I using the body sensor configuration in Section 3. The system was trained using approximately half the data. An action is considered properly labeled if over 50% of the action was labeled as the action and the rest was labeled as either the start or the end posture. If any part of an action was labeled as a different action, then the action is considered to be incorrectly labeled.

5.1. Comparison to Other Methods

The advantage of the classification technique developed in this article is the ability to both segment and classify data. In this section, the accuracy of our method is explored. However, it is useful to compare this with other techniques. First, we look at the accuracy when using *k*-NN classification with manually performed segmentation, as proposed in Ghasemzadeh et al. [2008]. The implementation uses data fusion to make a decision based on a full feature vector containing data from each node instead of the decision fusion technique outlined. Because the *k*-NN test uses manual segmentation and features extracted from all the data instead of from the transcripts, the *k*-NN results represent a conceptual upper bound on the accuracy.

The second method we choose for comparison is the HMM outlined in this article, but using features linearly quantized into nine levels. This has the potential for higher accuracy than clustering, since clustering potentially discards useful information. For our results, this method resulted in considerably higher error, probably due to a combination of overfitting, over-simple quantization, and lack of a feature-selection technique. The need for feature selection is especially likely, as there are only five training trials for each subject and over 20 features, some of which may be fairly useless or highly correlated with other features.

5.2. Classification Accuracy

Table II shows the result for each subject when a single model was trained on all subjects. With clustering, accuracy for each subject is reasonable, especially given the similarity of the movements. As expected, *k*-NN on an ideal (human-generated) segmentation outperforms the HMM. The lower accuracy of the HMM is compensated by significantly less required data transmission, as well as the automatic segmentation provided by the HMM.

Visual inspection of the transcripts shows consistency within subjects but marked differences between subjects for the same movements. This is expected because of the differences in the way subjects perform movements, that is, everybody sits down on the bed slightly differently. Subject 1 exhibited much higher intertrial consistency than the other two subjects, leading to Subject 1 dominating the trained model. This is the reason for Subject 1 having 10% higher accuracy than the other two subjects. By independently training the model on a per-subject basis, the bias towards a

Table III. Results for All Subjects Trained Individually

Subject	*k*-NN	Full Samples Accuracy	Clustering Accuracy
1	97.5	95.1%	90%
2	91.5	48.9%	94%
3	99.4	47.4%	94%

particularly consistent subject can be eliminated. The results of this approach are shown in Table III. The improvements for Subjects 2 and 3 are considerable, while the accuracy of Subject 1 actually dropped. A clue to this drop can be found in the training accuracy: Subject 1 has 100% accuracy over the training set. This is a symptom of overfitting. The classic solution to overfitting is increasing the number and variation of training trials. Subject 1's consistency is the primary reason that the subject had an overfitting problem sooner than the other subjects. The biggest disadvantage in this approach compared with training one model for all subjects is the requirement of more training data for each subject.

Once again, clustering outperforms full use of all samples but generally fails to beat *k*-NN, although the HMM with clustering approach produced the best results of any of the approaches for Subject 2.

Detailed results for Subject 1, as displayed in Table III, are shown in Table IV. Of special interest is the confusion column. Some of the misclassifications are expected. For instance, picking an object off the ground and off a coffee table are very similar, and so the confusion of Actions 10 and 8 make sense. Similarly, turning counterclockwise 90° is quite similar to returning from a clockwise turn. However, the confusion of *reaching up to a cabinet with left hand* and *move forward one step* makes little sense and so represents a true error.

A visual representation of the segmentation and classification process for Subject 1 is shown in Figure 4(a), and for the same movement with Subject 3 in Figure 4(b). The clusters are on the bottom. The labels in red are the canonical annotations (ground truth), while the ones above in blue are generated annotations (system output). The grayscale bar at the top represents the progression of states. Movements 11a and 11b are *turn counterclockwise* 90° and return, respectively. Movements 12a and 12b are *turn clockwise* 90° and *return*, respectively. The clusters from the left thigh show a very consistent pattern, while the clusters from the waist and right arm show significant variation. The HMM is able to accurately identify these actions from among all possible actions, as can be seen from the labeling at the top. Sometimes Subject 1 misidentifies a clockwise turn as the return from a counterclockwise turn, which is not necessarily even a mistake: the two may be impossible to distinguish even for a person. The transcripts for each subject are markedly different, even though both come from the same action. These figures give clear motivation to prepare separate models for each subject.

5.3. Bandwidth Savings from Clustering

The primary reason for choosing clustering instead of transmitting samples directly was decreasing transmissions. In Table V, the savings are shown. In the first column, the uncompressed, 12-bit per sensor data is transmitted, with results shown in Bytes per second. For the next column, sensor data is first quantized (with nine possible bins per sensor) then represented by two pieces of information: the quantized label and duration of samples of that value. The results shown are the average entropy per original sample. Coding methods, such as Huffman's, come close to achieving entropy, so this is a reasonable estimate of bandwidth. The final column is similar, except instead of quantized sensor data, clustering is performed.

Table IV. Results for an Independently Trained Subject 1

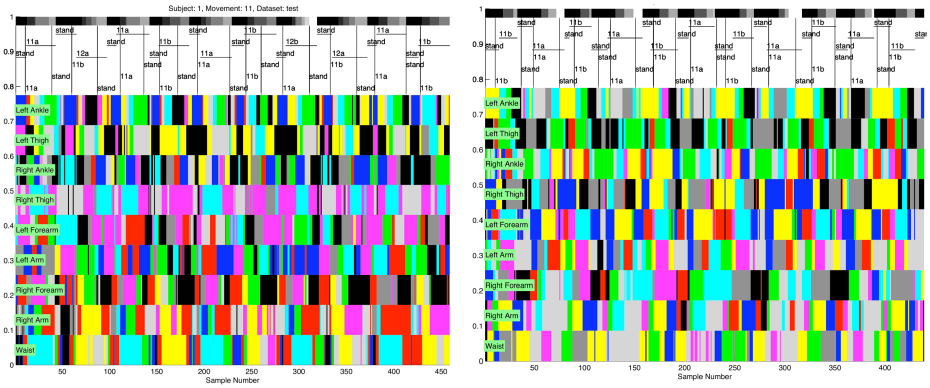
Action	# States	Accuracy	# Actions	Confusion
1	6	100%	12	
2	5	100%	12	
3	4	100%	11	
4	4	100%	11	
5	3	100%	10	
6	5	100%	10	
7	4	100%	5	
8	4	100%	5	
9	3	100%	5	
10	3	80%	5	8:1
11a	4	60%	5	12b:2
11b	3	100%	5	
12a	3	0%	5	11b:5
12b	3	100%	5	
13	6	100%	5	
14	7	67%	6	19a:1, 19b:1
15a	6	100%	5	
15b	3	100%	5	
16a	4	100%	5	
16b	4	100%	5	
17a	3	100%	6	
17b	3	60%	5	18a:2
18a	1	80%	5	17b:1
18b	3	100%	5	
19a	2	80%	5	11b:1
19b	3	100%	5	
20a	2	60%	5	18a:1, 21a:1
20b	2	20%	5	10:1, 17a:1, 21b:2
21a	2	100%	5	
21b	3	100%	5	
22	10	100%	6	
24	4	100%	5	
25	4	100%	5	
Total		90%		

The savings are most dramatic when compression of any kind is applied; however, clustering still reduces the bandwidth by about 75%.

5.4. Rejection Criterion

One application of this work is the life-logging application in which a participant wears sensors for several days, and the system automatically creates a diary of actions performed. Most classification systems, including our hidden Markov model, try to classify an action as being one of several possible actions. For life logging, many actions are novel and should not be labeled as one of the training actions. This *none of the above* labeling is also called a rejection criterion. There are several rejection strategies. The most basic is to compute a confidence for each action and if the measure is under a threshold, the action is rejected as not belonging to any known class [Kashi et al. 1998]. Another strategy is to train one or more rejection classes on a variety of data that is to be rejected. The prior probability of this class can be changed to raise or lower the rejection threshold [Rosenberg et al. 1998]. In our work, we investigate threshold-based methods.

As the model in this article already extracts the most likely sequence using the Viterbi algorithm, a natural confidence measure is the log-likelihood probability of a given action in the most likely sequence. The likelihood of the most likely state



(a) Subject 1. (b) Subject 3.

Fig. 4. Classification results for the action *turn Clockwise* 90°.

Table V. Data Savings from Clustering

Subject	Uncompressed (B/s)	Samples Cmp. (B/s)	Clustering Cmp. (B/s)
1	165.00	10.91	2.78
2	165.00	11.93	2.97
3	165.00	13.44	3.21

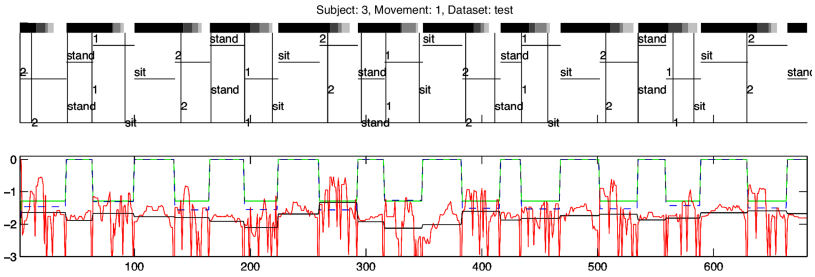
sequence decreases as the length of the sequence increases; therefore, it should be normalized by the path length [Kashi et al. 1998].

As will be seen in the results, this rejection criterion using a fixed threshold led to many false rejections and false recognitions. Looking at the state progression, as seen in the state progression bar in Figure 4(a), correctly recognized actions tended to spend approximately equal time in each state within the action, while incorrectly recognized actions tended to linger on one state for most of the time, then go through a rapid set of transitions to get to the next action. A simple way to quantify this is to look at the entropy of the observed chain compared to the theoretical entropy of the model. Entropy is maximized if all states have equal representation. Maximum entropy is $H_{\max}(x) = \log|X|$ for $x \in X$. We set a threshold for negative entropy. The action is accepted if

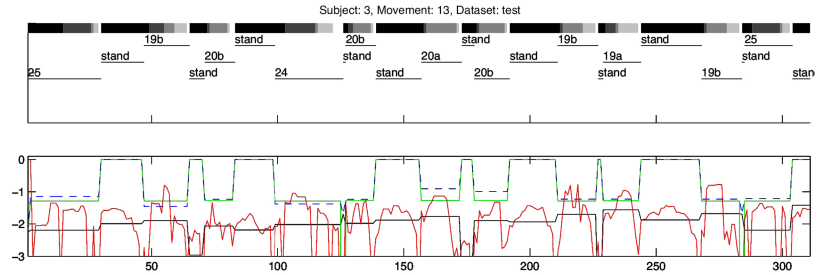
$$-k \cdot \log|X| \geq -H(S_i), \tag{18}$$

where $H(S_i)$ is the entropy of the state sequence extracted by the Viterbi algorithm.

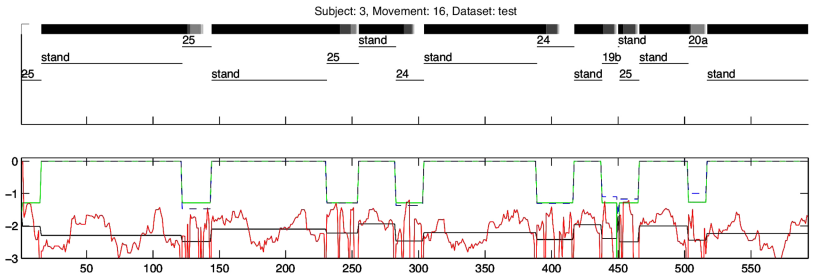
For testing several possibilities, movements 13–18b in Table I were removed from the training set to provide a set of actions to be rejected. The highest possible entropy is 0 for one state and goes up with the number of total states. To facilitate this measure, the models were constrained all to have five states per action. The results are shown in Figure 5. In this diagram, the top portion is identical to the clustering diagrams. The bottom shows various rejection criteria. The red lines are the likelihood of the current state and transition in the chain. The solid black lines are the average likelihood probabilities for the entire action. The solid green line is the entropy rejection threshold and the dashed blue line is the observed entropy. The entropy threshold is 80% of the expected entropy based on the trained model. For Figures 5(b) and 5(c), the actions were not part of the training set. For this reason, the data all theoretically represents non-actions (except for the postures); therefore, no ground truth data is shown. In these figures, we expect the log likelihood and entropy measures to go down.



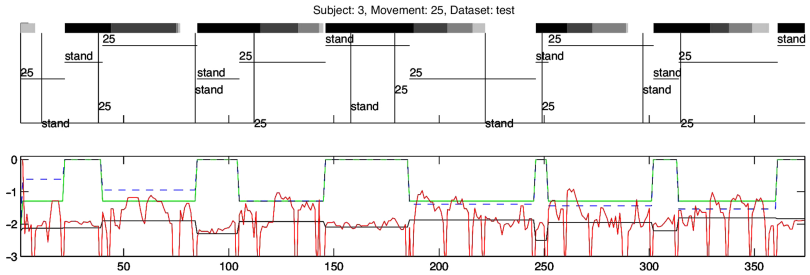
(a) Movement 1: stand to sit.



(b) Movement 13: look back ccw (should reject).



(c) Movement 16: kneeling (should reject).



(d) Movement 25: turn counterclockwise 360°.

Fig. 5. Rejection thresholds.

In Figure 5(a), the negative entropy is always below the threshold, so all relevant actions are correctly identified. The actions in Figures 5(b) and 5(c) are part of the rejection threshold. Some of the actions will be rejected by the entropy criterion,

but others will not. Looking at the average log likelihood combined with the entropy measure, it looks like sufficient evidence exists to reject. Movement 25 is shown in Figure 5(d) to show the thresholds for a correctly identified movement in contrast to the incorrect identifications for rejection class Movement 13.

5.5. Hardware Implementation

We recently implemented transcript generation on a set of sensor nodes based around the hardware described in Section 3.1. The transcripts produced were consistent with those produced in MATLAB.

6. CONCLUSION AND FUTURE WORK

In this article, we presented an action recognition framework based on an HMM which is capable of both segmenting and classifying continuous movements. It is specifically designed for the distributed architecture of body sensor networks and has modest runtime requirements, which is essential for resource-limited sensor nodes. The accuracy is consistent with results reported from similar experiments described in literature, but below that of a k-NN system using manual segmentation. We also examined the possibility of using thresholds based on log-probability and entropy to reject unknown movements. The methods are promising, but further work is needed to perfect them.

For deployment, several additional steps must be taken. First, in a system designed to monitor a subject throughout the day, many actions performed by the subject will not represent any of the trained actions. The system will need to not only recognize known actions but reject unknown actions. [Yoon et al. 2002] suggest a method based on rejection thresholds that could be used. Second, this system needs to be implemented on a BSN. Since our MATLAB tests proved successful, this is our next major step.

REFERENCES

AKYILDIZ, I., SU, W., SANKARASUBRAMANIAM, Y., AND CAYIRCI, E. 2002. Wireless sensor networks: a survey. *Comput. Netw. 38,* 4, 393–422.

AMINIAN, K., NAJAFI, B., BÜLA, C., LEYVRAZ, P., AND ROBERT, P. 2002. Spatio-temporal parameters of gait measured by an ambulatory system using miniature gyroscopes. *J. Biomech. 35,* 5, 689–699.

BAO, L. 2003. Physical activity recognition from acceleration data under semi-naturalistic conditions. Ph.D. dissutation, Massachusetts Institute of Technology, Cambridge, MA.

BAO, L. AND INTILLE, S. 2004. Activity recognition from user-annotated acceleration data. In *Proceedings of the 2nd International Conference on Lecture Notes in Computer Science Pervasive Computing*. vol. 3001, Springer, Berlin, 1–17.

BIEM, A. 2003. A model selection criterion for classification: Application to hmm topology optimization. In *Proceedings of the 7th International Conference on Document Analysis and Recognition*. 104–108.

CASTELLI, G., ROSI, A., MAMEI, M., AND ZAMBONELLI, F. 2007. A simple model and infrastructure for context-aware browsing of the world. In *Proceedings of the IEEE International Conference on Pervasive Computing and Communications*. 229–238.

CHOUDHURY, T., BORRIELLO, G., CONSOLVO, S., HAEHNEL, D., HARRISON, B., HEMINGWAY, B., HIGHTOWER, J., KLASNJA, P., KOSCHER, K., LAMARCA, A., LANDAY, J. A. and LESTER, J. 2008. The mobile sensing platform: An embedded system for capturing and recognizing human activities. *IEEE Pervasive Mag. Special Issue on Activity-Based Computing*.

COURSES, E., SURVEYS, T., AND VIEW, T. 2008. Analysis of low resolution accelerometer data for continuous human activity recognition. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP'08).* 3337–3340.

FIGUEIREDO, M. AND JAIN, A. 2002. Unsupervised learning of finite mixture models. *IEEE Tran. Pattern Anal. Mach. Intell.* 24, 3, 381–396.

FRALEY, C. AND RAFTERY, A. 1998. How many clusters? Which clustering method? Answers via model-based cluster analysis. *Comput. J. 41,* 8, 578–588.

GHASEMZADEH, H., GUENTERBERG, E., GILANI, K., AND JAFARI, R. 2008. Action coverage formulation for power optimization in body sensor networks. In *Proceedings of the Asia and South Pacific Design Automation Conference (ASPDAC'08).* 446–451.

GUENTERBERG, E., OSTADABBAS, S., GHASEMZADEH, H., AND JAFARI, R. 2009. An automatic segmentation technique in body sensor networks based on signal energy. In *Proceedings of the 4th International Conference of Body Area Networks (BodyNets'09)*.

GUENTERBERG, E., YANG, A. Y., GHASEMZADEH, H., JAFARI, R., BAJCSY, R., AND SASTRY, S. S. 2009. A method for extracting temporal parameters based on hidden Markov models in body sensor networks with inertial sensors. *IEEE Trans. Inform. Technol. Biomed*. *13*, 6, 1019–10300.

HARTIGAN, J. AND WONG, M. 1979. A K-means clustering algorithm. *JR Stat. Soc. Ser. C-Appl. Stat. 28*, 100–108.

HAUSDORFF, J., CUDKOWICZ, M., FIRTION, R., WEI, J., AND GOLDBERGER, A. 1998. Gait variability and basal ganglia disorders: Stride-to-stride variations of gait cycle timing in Parkinson's disease and Huntington's disease. *Mo. Disord. 13,* 3, 428–37.

JOHNSON, S. 1967. Hierarchical clustering schemes. *Psychometrika 32,* 3, 241–254.

JURAFSKY, D., MARTIN, J., KEHLER, A., VANDER LINDEN, K., AND WARD, N. 2000. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. MIT Press, Cambridge, MA.

KASHI, R., HU, J., NELSON, W., AND TURIN, W. 1998. A hidden Markov model approach to online handwritten signature verification. *Int. J. Doc. Anal. Recog. 1,* 2, 102–109.

LEE, H. AND KIM, J. 1999. An HMM-based threshold model approach for gesture recognition. *IEEE Trans. Pattern Anal. Mach. Intell. 21*, 10, 961–973.

LESTER, J., CHOUDHURY, T., KERN, N., BORRIELLO, G., AND HANNAFORD, B. 2005. A hybrid discriminative/generative approach for modeling human activities. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI'05)*.

LV, F. AND NEVATIA, R. 2006. Recognition and segmentation of 3-D human action using HMM and multiclass AdaBoost. In *Proceedings of the 9th European Conference on Computer Vision (ECCV'06)*. Lecture Notes in Computer Science vol. 3954, Springer, Berlin, 359.

NAIT-CHARIF, H. AND MCKENNA, S. 2004. Activity summarisation and fall detection in a supportive home environment. In *Proceedings of the 17th International Conference on Pattern Recognition (ICPR'04)*.

OUCHI, K., SUZUKI, T., AND DOI, M. 2004. LifeMinder: A wearable healthcare support system with timely instruction based on the user's context. In *Proceedings of the 8th IEEE International Workshop on Advanced Motion Control (AMC'04)*. 445–450.

POLASTRE, J., SZEWCZYK, R., AND CULLER, D. 2005. Telos: Enabling ultra-low power wireless research. In *Proceedings of the 4th International Symposium on Information Processing in Sensor Networks*.

QUWAIDER, M. AND BISWAS, S. 2008. Body posture identification using hidden Markov model with a wearable sensor network. In *Proceedings of the ICST 3rd International Conference on Body Area Networks*.

RABINER, L. AND JUANG, B. 1986. An introduction to hidden Markov models. *ASSP Mag. 3,* 1, 4–16. See also IEEE Signal Process. Mag.

RAUDYS, S. AND JAIN, A. 1991. Small sample size effects in statistical pattern recognition: Recommendations for practitioners. *IEEE Trans. Pattern Anal. Mach. Intell. 13,* 3, 252–264.

REYNOLDS, D. AND ROSE, R. 1995. Robust text-independent speaker identification using Gaussianmixture speaker models. *IEEE Trans. Speech Audio Process. 3,* 1, 72–83.

ROSENBERG, A., SIOHAN, O., AND PARATHASARATHY, S. 1998. Speaker verification using minimum verification error training. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*.

SHERIDAN, P., SOLOMONT, J., KOWALL, N., AND HAUSDORFF, J. 2003. Influence of executive function on locomotor function: Divided attention increases gait variability in Alzheimer's disease. *J. Am. Geriatrics Soc. 51,* 11, 1633–1637.

STOICA, P. AND SELEN, Y. 2004. Model-order selection: A review of information criterion rules. *IEEE Signal Process. Mag. 21,* 4, 36–47.

VAN LAERHOVEN, K. AND GELLERSEN, H. 2004. Spine versus Porcupine: A study in distributed wearable activity recognition. In *Proceedings of the 8th IEEE International Symposium on Wearable Computers*. 142–149.

WARD, J., LUKOWICZ, P., TRÖSTER, G., AND STARNER, T. 2006. Activity recognition of assembly tasks using body-worn microphones and accelerometers. *IEEE Trans. Pattern Anal. Mach. Intell.*, *28,* 10, 1553–1567.

WESSEL, F., MACHEREY, K., SCHLUTER, R., FUR INF, L., AND AACHEN, T. 1998. Using word probabilities as confidence measures. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*.

YANG, A., JAFARI, R., SASTRY, S., AND BAJCSY, R. 2009. Distributed recognition of human actions using wearable motion sensor networks. *J. Ambient Intell. Smart Environ. 1*, 1–5.

YOON, H., LEE, J., AND YANG, H. 2002. An online signature verification system using hidden Markov model in polar space. In *Proceedings of the 8th International Workshop on Frontiers in Handwriting Recognition*. 329–333.

Received October 2009; revised June 2010; accepted September 2010