A Method for Extracting Temporal Parameters Based on Hidden Markov Models in Body Sensor Networks With Inertial Sensors

Eric Guenterberg, *Student Member, IEEE*, Allen Y. Yang, *Member, IEEE*, Hassan Ghasemzadeh, *Student Member, IEEE*, Roozbeh Jafari, *Member, IEEE*, Ruzena Bajcsy, *Fellow, IEEE*, and S. Shankar Sastry, *Fellow, IEEE*

***Abstract*—Human movement models often divide movements into parts. In walking, the stride can be segmented into four different parts, and in golf and other sports, the swing is divided into sections based on the primary direction of motion. These parts are often divided based on key events, also called temporal parameters. When analyzing a movement, it is important to correctly locate these key events, and so automated techniques are needed. There exist many methods for dividing specific actions using data from specific sensors, but for new sensors or sensing positions, new techniques must be developed. We introduce a generic method for temporal parameter extraction called the hidden Markov event model based on hidden Markov models. Our method constrains the state structure to facilitate precise location of key events. This method can be quickly adapted to new movements and new sensors/sensor placements. Furthermore, it generalizes well to subjects not used for training. A multiobjective optimization technique using genetic algorithms is applied to decrease error and increase cross-subject generalizability. Further, collaborative techniques are explored. We validate this method on a walking dataset by using inertial sensors placed on various locations on a human body. Our technique is designed to be computationally complex for training, but computationally simple at runtime to allow deployment on resource-constrained sensor nodes.**

***Index Terms*—Biped locomotion, body sensor networks, hidden Markov models, intelligent sensors.**

I. Introduction

THE ANALYSIS and monitoring of human movement offers many useful applications in geriatrics, fall risk assessment, disease monitoring, law enforcement, sports training, and rehabilitation. Such applications often start with a model describing human movement for a given population. This model may be parametrized by variables, such as the severity of the disease or walking speed. Then a recognition system is created to match the model and determine useful parameters using data from a sensor or combination of sensors.

Many movement models divide an action into several parts. In sports, many swings can be divided into phases. The golf swing is separated into "takeaway," "backswing," "downswing," and "follow through" portions [1]. In walking, the human stride is marked by several events such as "initial stance" (the foot placed on the ground), "mid-stance," "initial swing" (the foot has just been lifted from the ground), and "mid-swing" which repeat indefinitely [2]. These events are referred to as temporal parameters. Various papers identify different key events for the same action depending on the application and model. Sometimes these divisions can be used directly. For instance, high standard deviation of stride time during walking is indicative of Parkinson's disease or Huntington's disease and can be used to assess the risk of falling [3], [4]. Many models define these events based on limb position. However, these events also occur at specific temporal locations in the movement sequence. Inertial sensors provide movement data directly, therefore, these events can be found by looking for patterns in the sensor data.

Systems extracting these parameters have mostly used cameras or marker-based approaches to extract limb positions and orientations [5]. However, recently a new modality has been used that employs wireless sensors mounted on a human body. This configuration is called a body sensor network (BSN). While a variety of sensors can be used, the most common are accelerometers and gyroscopes because of their small size, low power usage, and useful motion data [5]. BSNs are always with a person, and can collect data at any location the subject travels to. Sensors must be placed directly on the subjects. However, if the application permits, these sensors can be placed in cell phones and other artifacts already carried by the subjects. Researchers are experimenting with incorporating these sensors into clothing, potentially easing the wearability concerns in future [6], [7].

Methods have been developed to recognize key events for specific movements and sensor configurations. However, the rapid development of sensors, wearable electronics, and the

Manuscript received October 15, 2008; revised March 26, 2009. First published September 1, 2009; current version published November 4, 2009. This work was supported in part by The Team for Research in Ubiquitous Secure Technology, which receives support from the National Science Foundation under NSF Award CCF-0424422, Air Force Office of Scientific Research (AFOSR) (#FA9550-06-1-0244), Cisco, British Telecom, ESCHER, HP, IBM, iCAST, Intel, Microsoft, ORNL, Pirelli, Qualcomm, Sun, Symantec, Telecom Italia, and United Technologies, and in part by Army Research Office/Multidisciplinary University Research Initiative W911NF-06-1-0076.

E. Guenterberg, H. Ghasemzadeh, and R. Jafari are with Embedded Systems and Signal Processing Laboratory, Department of Electrical Engineering, University of Texas at Dallas, Dallas, TX 75080 USA (e-mail: etg062000@utdallas.edu; h.ghasemzadeh@gmail.com; rjafari@utdallas.edu).

A. Y. Yang and R. Bajcsy are with the Department of Electrical Engineering and Computer Science, University of California, Berkeley, CA 94720 USA (e-mail: yang@eecs.berkeley.edu; bajcsy@eecs.utdallas.edu).

S. S. Sastry is with the Center for Information Technology in the Interests of Society, University of California, Berkeley, CA 94720 USA (e-mail: sastry@eecs.berkeley.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TITB.2009.2028421

corresponding proliferation of new applications motivate the need for a generic system that can be rapidly retrained for new applications and sensor configurations without significant research. We envision a scenario where a designer has training data recorded from one or more subjects using the desired sensors. The desired key events occur in a specific order and are labeled in the training data. The system should then be able to train with this data to recognize the key events.

In this paper, we present such a system called the hidden Markov event model (HMEM). The HMEM uses a hidden Markov model (HMM) with a specific state structure to find key events. Certain elements of the state structure can be parametrized and the specific statistical features used to recognize key events can be selected from a potentially large list. Genetic algorithms (GAs) are used to select features and parametrize the model with the objectives of reduced error, small feature set, and increased generalizability to previously unknown subjects. The novelty of this work is the development of a generic technique for event extraction. The specific model is based on HMMs, but uses a unique structure designed to extract these events.

II. Related Work

Many models for temporal parameter extraction first estimate the position of limbs over time, then label key events based on this data. Our work uses statistical techniques that bypass the position estimation phase. However, much work in this field has concentrated on position tracking; therefore, we will look at some of these techniques in the following paragraphs. There are three primary types of position tracking systems: marker tracking, vision-based recognition, and inertial sensing, respectively.

Marker systems track the position and possibly orientation of markers placed on the subjects limbs. Ultrasonic, magnetic, or vision systems may be employed. Ultrasonic systems use markers that emit ultrasonic pulses in response to an infrared transmission from the base. The position is determined by measuring the time it takes for the sonic pulse to reach several fixed sensors in the room. Orientation cannot be determined with ultrasonics alone [8]. Other systems use magnetic markers whose position and orientation can be tracked. These systems are extremely vulnerable to interference from ferrous materials [5], [9]. Vision systems use multiple video cameras to track passive optical markers (reflectors) or active optical markers (flashing LEDs) [5]. The advantage of marker systems is their ability to get highly accurate position information for each limb segment. Unfortunately, markers must be placed on precise positions on the subject's body, and the subject is restricted to movement within a small geographical area. Some of these systems also require extensive post-processing work to correlate markers over time.

Another type of system uses cameras to track movements in a natural setting without requiring the subject to wear specialized equipment or clothing. Signal processing and pattern recognition steps must be applied to determine the number of people in a scene and their postures at any given time. Successive frames can be combined to analyze motion [10]. Such systems have been used to track activities of the elderly, identify people from their gait, detect suspicious activity, etc. [11]–[13]. The main advantage of these systems is that they scale well with the number of people in a room, and do not require the subjects to wear special clothing or markers. The disadvantages include privacy concerns, and the inability to follow subjects beyond the range of the cameras.

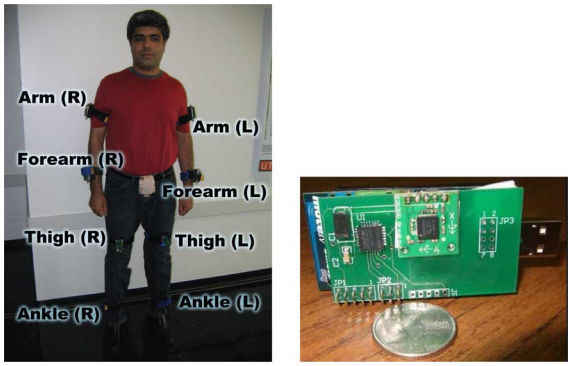
Unlike the marker and vision systems, inertial sensors cannot directly track limb position and orientation. State estimation techniques can be used to approximate joint position/orientation if the initial conditions are known, but is subject to significant drift error. Drift-free estimations have been obtained by assuming specific models of motion for specific actions [14]–[16]. But these models break down if the subject moves in a way that violates the assumptions.

For the specific application of human gait analysis, there already exist several techniques to segment the gait using data from inertial sensors. In [14], the authors derive equations for determining all four of the gait events using a gyroscope and accelerometer placed on the foot. The events are determined using the gyroscope, and the accelerometer is used to find stride length and inclination of the walking surface. Miyazaki [17] demonstrates the use of just a gyroscope placed on the thigh to extract stride time, stride length, and walking speed [17]. Using a gyroscope attached to both shanks and thigh, Aminian *et al.* [18] were able to estimate the time of each of the events and other information such as stride length. They used a simplified physical model of the shank and thigh during walking to derive equations and pick features from a wavelet transform.

These systems accurately extract not only the events, but spatial parameters such as stride length. They are all based on specific physical models of walking. If a new type of sensor is employed or the sensor is placed in a new location, a new model must be derived, and new techniques are required to extract the parameters. In contrast, our method focuses on extracting the temporal parameters, also called key events, but can be used without modification with new sensor types and positions. Furthermore, the assumptions made for HMEM apply to movements other than walking, so the same framework can capture information about a variety of actions.

Another approach is to extract model parameters without directly calculating angle and position. Researchers at the Royal Veterinary College in the University of London attached a sensor to seven horses and used a left-right HMM model to determine if a horse was galloping or not, and if it was, to separate the movement into strides [19]. Our approach bears similarity to this paper, especially with our choice of the left-right HMM for stride segmentation. However, our model differentiates itself by being able to identify multiple key events using event states, the use of collaborative processing, and explicit feature selection and parameterization using genetic algorithms to reduce error and choose only relevant features.

Quwaider and Biswas [20] apply HMMs to BSNs. They 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 energy level on each limb. For relatively quiet postures, such as



(a) (b)

Fig. 1. Sensor boards and positioning. (a) Sensors placed on body. (b) Sensor node.

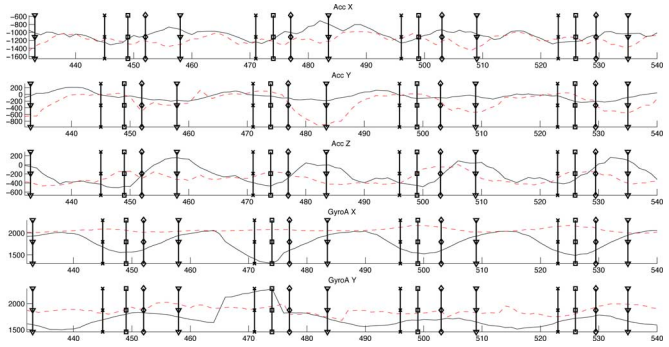


Fig. 2. Example walking signal with annotations (acceleration X, Y, Z, and angular velocity θ, ϕ). The dashed signal is from the right ankle and solid is from the right thigh. The annotations are initial swing (x), mid-swing (square), initial stance (diamond), and mid-stance (triangle).

sitting and standing, they employ an HMM used on radio signal strength (RSSI) differences between sensor nodes. In contrast, once the action is known, our model finds key events within the action using an HMM.

III. Sensing Architecture and Experimental Procedure

Data collection for our experiments revolve around the concept of a BSN composed of multiple sensor nodes. Each sensor node comprises several sensors, a processing unit, and a radio as shown in Fig. 1(b). The sensor nodes use the commercially available Tmote Sky with a custom-designed sensor board and are powered by two AA batteries. The Tmote Sky uses the 16 bit, 4 MHz TI MSP430 processor. The sensor board includes two MEMS sensors: a triaxial accelerometer and a biaxial gyroscope. The gyroscope uses the coriolis force to measure angular velocity about the two planar axes, but not the axis normal to the plane of the sensor board.

We placed eight sensor nodes on our subjects as shown in Fig. 1(a). The sensor nodes sampled data at 22 Hz and broadcast all samples to a base station node using the TDMA protocol and recorded on a PC. The sampling rate was experimentally chosen to provide sufficient resolution of human motion data while compensating for bandwidth constraints on our sensor platform. For the results in this paper, all further processing was performed in MATLAB, although the next goal of our research is to train our system in MATLAB and implement it on the sensor nodes.

A. Experimental Protocol and Data Annotation

We recruited eight subjects ages 18–23 who were directed to walk around a large conference table with approximately 100 steps each. The data were synchronized with video that was used by a volunteer to manually identify four key events in each stride. The two most important events in literature, the toe leave and heel touch can be easily recognized from the video. The volunteer used criteria based on knee and leg positions for the other two events, which may not correspond precisely to the definitions used by the bio-mechanical community, but which were consistent for all our data. An example of the walking data is shown in Fig. 2. The four events, identified relative to the right foot, are [2]:

1) *Initial swing:* the toe of the right foot leaves the ground.
2) *Mid-swing:* the middle of the swing.
3) *Terminal swing/intial stance:* the heel touches the ground.
4) *Mid-stance:* the leg is oriented vertically in stance.

One problem with statistical methods is choosing the complexity of the model. The more complex a model, the better it describes the given data, but the more likely the model is conforming to specific noise and eccentricities present in the training data. This is called the overfitting problem. Cross-validation is a classic way to avoid over-fitting [21]. For this method, training data is divided into two sets, the cross-validation set and the training set. The sets are mutually exclusive. Several models of varying complexity are trained using the training data, then tested on the cross-validation dataset. The model with the lowest cross-validation error is considered to be the correct model.

For our data, three unique subjects are used for training, three for cross-validation, and two for testing. Half the data from the cross-validation and training subjects is reserved for testing. The testing set is used to report data on the performance of our system, and so cannot be involved in training at all. All three sets are mutually exclusive. The use of different subjects in the training and cross-validation set encourages generalizability across subjects.

IV. HMEM Training and Use

The HMEM is the name of our key event labeling system, which uses an HMM with a specific state structure and a modified training procedure designed to find key events. The model also adds a feature selection and model parametrization system based on GAs. The HMEM makes several assumptions about the underlying data:

1) there are a number of different event types;
2) the events always occur in a specific order and for cyclical movements they repeat;
3) every single event type is represented in every action;
4) there are a number of unlabeled samples between two adjacent events.

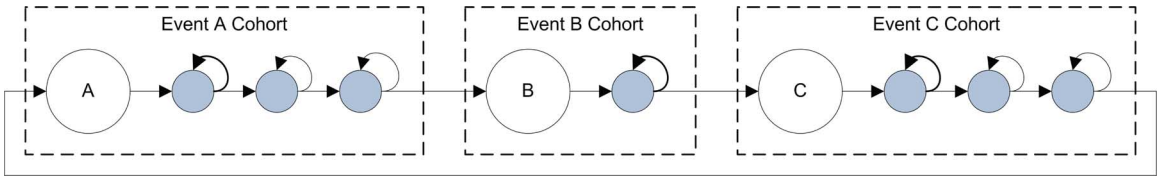


Fig. 3. HMEM model and structure.

A traditional pattern recognition technique used for time-varying signals is the HMM. A basic HMM describes a discrete-time Markov process. At a particular moment the process is in just one state. At fixed time intervals the process produces an output and then transitions to another state (or remains in the current state). The transitions and outputs are probabilistic and based exclusively on the present state. The process generates a sequence of outputs, and a corresponding sequence of states. The states cannot be directly observed, and are thus hidden. An HMM is completely described by initial state probabilities, state transition probabilities, and output probabilities. Algorithms exist to:

1) build an HMM to describe a given set of output sequences;
2) choose which of several models best describe an output sequence;
3) find the most likely current state of an HMM given the output sequence up until now;
4) the most likely state sequence associated with a particular output sequence for a specified HMM [22], [23].

For a general HMM, it is possible for any state to transition to any other state. This is called an ergodic model. Another variant is to enforce a specific ordering for the states: each state can only transition to itself or state to the "right" of it in the ordering. This is called a left-right model [23].

Each key event can be represented by a unique state. Ideal events occur at a specific time but have no duration. However, given the idea that the key event might be associated with unique features in the observation sequence, the key event state should have a one sample duration. The HMEM encodes this concept into an HMM by removing the self-transition from states associated with key events, forcing a transition after one sample. The samples between key events are represented by transition states which support both self-transitions and forward transitions, as seen in Fig. 3. States are grouped into cohorts that start with a key event state and end with the last transition state before the next key event state. For any observation sequence in the training data, the positions of the key event states are known. This means that training each cohort independently is identical to training the whole system at once.

A. Overview

There are several stages required to train the HMEM as shown in Fig. 4.

1) Preprocessing and Feature Extraction: The signal is filtered with a five-point moving average to remove high-frequency noise. Then, it is normalized by subtracting a large-window mean and dividing by a large-window standard deviation. This window size is 101 samples in our system. Several parameters representing the action data inside the signal, referred to as features, are extracted at each sample. These features are further quantized with a ten-level uniform quantizer based on the range of the features in the training data. The details for this are discussed in Section IV-B and IV-C.

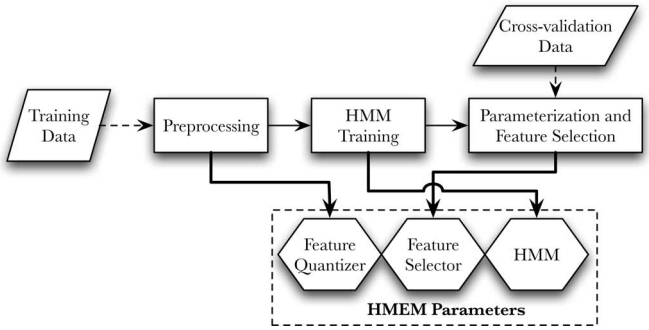


Fig. 4. HMEM training procedure.

2) HMM Training: The HMM is effectively a finite-state machine with probabilistic transitions and certain emission probabilities. These probabilities must be specified as part of the mathematical model defining an HMM. The exact locations of all key events states and what observations they emit are known from the annotations in the training data. However, the number of transition states and their transition and emission probabilities are unknown and must be trained. There are several well-known techniques for training HMMs, including Baum–Welch and Viterbi path counting (VPC) [22], [24].

The training data are segmented using the canonical annotations. Each cohort is trained independently using a set of segments that start with a sample that should be labeled with the cohorts event and end just before the next labeled event. According to our model, the first state must be the cohort's event state, and the last sample must be associated with the last transition state in the cohort. During training, it is important to make sure that all considered paths meet this constraint. VPC produces a single path for each event that can be edited to meet the constraints if necessary, while Baum–Welch can also be constrained in this way VPC is much faster, which is important given the already high training times. This feature is one of the primary reasons for choosing VPC over Baum–Welch. The details of the training process are discussed in Section IV-D.

3) Parametrization and Feature Selection: HMMs are trained to represent a process, not to minimize segmentation error. It is possible to explicitly attempt to increase classification accuracy by choosing model parameters with that goal in mind. The parameters we tune are selected features and number of transition states for each cohort. We use a genetic algorithm

with uniform crossover to train the model. The population fitness is evaluated using the training model, and then at the end, the model that gives the best results for the cross-validation data is selected. Further discussion follows in Section IV-E.

B. Feature Extraction

The HMM must have a new observation $o \in X$ every sample. In our system, the observation is actually a multidimensional vector, called a feature vector $\vec{f}$. Features are essentially numbers summarizing the data in some way. The most basic feature vector would include the value of each sensor at time of observation. However, other types of features, such as the derivative, might better represent important information for recognizing key events.

The feature vector for a given sample is composed of features extracted from each sensor, and perhaps features extracted from the past and the future. The feature extraction functions used are described shortly. For sensor and node at time i, the feature vector is $\vec{F}_i'' = \langle v_i, d_i, dd_i, p_i \rangle$.

1) *Sample value v_i:* The value of the preprocessed signal at time i.
2) *Derivative d_i:* The approximation of the derivative for time i. $d_i = (v_{i+1} - v_{i-1})/2$
3) *Second derivative dd_i:* $dd_i = (d_{i+1} - d_{i-1})/2$
4) *Peak detector p_i:* This is the current sample value divided by the maximum value in a window centered on the current value (we use a 15-sample window). For a peak in the signal, $p_i = 1$. The value is less for nonpeaks

$$p_i = \frac{v_i}{\max_{j=-w/2}^{w/2} v_{i+j}}.$$

The feature vectors from each sensor and each sensor node are concatenated to form the feature vector $\vec{F}_i'$.

1) Temporally Proximate Features: A variation tried was to use features from the near past and future of the current sample to compose a larger feature vector. This is based on the idea that a key event may actually occur a few samples before or after a distinguishable characteristic appears in feature space. This leads to the vector $\vec{F}_i = \langle \vec{F}_{i-w/2}', \vec{F}_{i-w/2+1}', \ldots, \vec{F}_i', \ldots, \vec{F}_{i+w/2}' \rangle$. We use a window size of 11 samples. Section VI will show the increase in accuracy gained from this approach. A feature vector that includes these future and past values is said to contain temporally proximate features.

C. Feature Quantization

The observations may be continuous or taken from a finite alphabet. In either case, the probability density function (PDF) of the model needs to be estimated for each state. During application, the probability must be computed. A common model is the Gaussian mixture model that uses multiple weighted normal distributions to represent the probability distribution. However, such a model would greatly tax the processing resources of a sensor node during application. For this reason, the features were quantized and the probability for each of the values estimated and stored. Features are quantized by observing the range of each feature in the training set and uniformly dividing the range into K levels. All features are quantized like this before proceeding to the next step. We used $K = 10$. Uniform quantization is very simple to compute, which is why it was chosen. This technique seems naive, but it was effective.

D. HMM Training and the Viterbi Algorithm

A Markov process has N states $S = \{s_1, s_2, \ldots, s_N\}$, and can emit M observations $X = \{x_1, x_2, \ldots, x_M\}$. For a given observation sequence $O_T = (o_1, o_2, \ldots, o_T)$ with T observations, there is a corresponding state sequence $Q_{(T)} = (q_1, q_2, \ldots, q_T)$. The HMM $\lambda = \{\pi, A, B\}$ is defined by three sets of probabilities: initial state probabilities $\pi = \{\pi_i | \pi_i = P(q_o = s_i)\}$, state transition probabilities $A = \{a_{ij} | a_{ij} = P(q = s_j | q_{\text{prev}} = s_i)\}$, and observation probabilities $B = \{b_j(k) | b_j(k) = P(o = x_k | q = s_j)\}$.

The most common training algorithm is the Baum–Welch algorithm [22]; however, a newer algorithm, the VPC algorithm is more appropriate for our paper [24]. Both algorithms follow the training procedure shown in Algorithm 1. They start with a fixed number of states and an initial set of model parameters, then extract probabilistically weighted state sequences using the model parameters. Next, the transition and emission probabilities are updated based on the transitions and observations associated with each state sequence. The initial model is updated with these new probabilities. This process repeats until some desired level of convergence is reached. It will implicitly converge to a local minima.

The difference between the two training approaches lies in the state sequences considered. The Baum–Welch approach considers all possible state sequences for each observation sequence,[1] while the VPC approach considers only the most likely state sequence for each observation sequence. The reasoning behind VPC is that in a classification scenario, the most likely sequence, given the current HMM model parameters, is chosen and all others are discarded, so it is best to train the system exclusively using the most likely sequence. The other reason we chose VPC is that various state sequences might not end on the final state in the event cohort as required. Using VPC, we can edit the state sequences to enforce the requirement before re-estimating the probability. We choose initial parameters by splitting the samples evenly among all states as done in [19].

The key to VPC is extracting the most likely state sequence.

$$Q_{(T)\max} = \arg\max_{Q_{(T)} \in S^T} P\left(Q_{(T)}, O_{(T)}\right). \quad (1)$$

A dynamic programing algorithm, called the Viterbi algorithm [22], solves this problem. Using the most likely state sequence extracted using the Viterbi algorithm, the transition and emission probabilities are found simply by counting the occurrences in all the most likely state sequences for each observation sequence in the training set. Since we are training one cohort at a time, each observation sequence is a sequence starting on the key event and ending right before the next key event.

[1]Because of certain properties of Markovian processes, the probabilities can be determined without literally examining every state sequence, even though all are considered.

Algorithm 1 HMM Training Procedure

Require: λ_0, $\mathcal{O} = \{O^1_{(T_1)}, O^2_{(T_2)}, \cdots, O^Y_{(T_Y)}\}$
1: $\lambda \leftarrow \lambda_0$
2: **for** $i \leftarrow 1$ to K **do** {estimates from $\mathcal{Q}$}
3: $\quad \mathcal{Q} \leftarrow \emptyset$
4: $\quad$ **for all** $O_{(T)} \in \mathcal{O}$ **do**
5: $\quad\quad Q_{(T)} \leftarrow$ collect weighted state sequences using λ
6: $\quad\quad \mathcal{Q} \leftarrow \mathcal{Q} \cup Q_{(T)}$
7: $\quad$ **end for**
8: $\quad \bar{\pi}_i = \frac{\text{number of times in state } s_i \text{ at time 1}}{\text{number of times at time 1}}$
9: $\quad \bar{a}_{ij} = \frac{\text{number of transitions from state } s_i \text{ to state } s_j}{\text{number of transitions from state } s_i}$
10: $\quad \bar{b}_j(k) = \frac{\text{number of times in state } s_j \text{ observing symbol } x_k}{\text{number of times in state } s_j}$
11: $\quad \lambda \leftarrow \{\bar{\pi}, \bar{A}, \bar{B}\}$
12: **end for**

One of the model parameters is the number of states. This training method requires the number of states to be known, so each cohort is trained five times with one to five transition states.

Maximizing the probability is equivalent to maximizing the log probability; therefore, we use log probabilities to prevent numerical underflow and facilitate faster computing.

1) Algorithmic Complexity: The order of the Viterbi algorithm for a left-right model with independent features is $O(\text{Viterbi}) = O(TxN)O(Pr_{\text{est}})$, where $O(Pr_{\text{est}})$ is the order of algorithm required to estimate probability. This order is constant time for the state transition probability, but based on the number of features for the observation probability estimation, as explained in the next section. This means $O(\text{Viterbi}) = O(TxNx|\vec{F}|)$.

2) Estimating Observation Probability: The observation probability at time t is $P(o_t|q_t)$. There are F features and K quantized feature levels. There are then $F \times K$ possible observable values. In our data, using $F = 4$ features $\times\, 5$ sensors $= 20$ and $K = 10$, we get 200 observable values. This is of the same order as the size of our training set, making proper probability estimation difficult. This is called the small sample size problem [25]. We adopt one of the suggested solutions: we consider each feature to be independent of all other features. This means

$$\log P(o_t|q_t) = \sum_f \log P\left(o_t(f)|q_t\right). \tag{2}$$

The order of this estimate is linear with respect to the number of features.

E. Feature Selection and Model Parametrization Using Genetic Algorithms

The choice of whether or not to include each feature and the choice of the number of transition states for the cohorts are all tunable parameters of the HMEM. The feature selection $\Psi = \{\psi_k|\psi_k \in \{0,1\}, i = 1, \ldots, |F|\}$ represents a choice of which features out of an exhaustive list are to be included and which are to be discarded. The number of selected features is $|\Psi| = \sum_k \psi_k$. Feature k is selected if $\psi_k = 1$ and is discarded if $\psi_k = 0$. The other parameter is the number of transitions states for each cohort $\Omega = \{\omega_e|\omega_e \in \{1, \ldots, 5\}, e = 1, \ldots, E\}$, where E is the number of key events. The full HMEM model is represented by $\lambda_{\text{HMEM}} = \{\lambda_{\text{HMM}}, \Psi, \Omega\}$.

Algorithm 2 Select Operator

1: *function* ρ_{out} **= select($\mathcal{F}$, $\mathcal{P}$)**
2: $X \leftarrow \sum_{\zeta \in \mathcal{F}} \zeta$ { Compute total fitness}
3: **for** $j \leftarrow 1$ to $|\mathcal{F}|$ **do**
4: $\quad p_j \leftarrow \frac{\zeta_j}{X}$
5: **end for**
6: $i \leftarrow$ choose i from 1 to $|\mathcal{F}|$ weighted by p_j
7: **return** ρ_i taken from $\mathcal{P}$
8: **end function**

In essence, parametrization consists of choosing several good models based on one or more objective functions applied to the training set. These models are then compared against each other using the same objective function(s) applied to the cross-validation set. The best model on the cross-validation set is chosen. Because the cross-validation set exclusively contains data from subjects not in the training set, generalizability of the models to new subjects is improved. Further objective functions are discussed below. GAs are used to generate the list of "good" models.

1) GAs: GAs are a class of random optimization problems. Like other optimization problems, they attempt to minimize an objective function. GAs do not by nature converge to a solution, but try to find a population that contains the solution. Given infinite time, GAs will find the global optimum [26].

Our objective function is η, as described in Section VI-A. Each solution must be representable by a chromosome ρ. For our system, this chromosome is $\rho = \langle \Psi, \Omega \rangle$. The basic idea behind GAs is that a set of parametrizations is created randomly by assigning values to the genes. This set is called a population, $\mathcal{P}$. At each generation, each gene ρ_i in the population is assigned a fitness ζ_i in the fitness vector $\mathcal{F}$ corresponding to population $\mathcal{P}$. Members of one generation are randomly selected for cloning, mutation, or crossover (two-parent reproduction) to populate the next generation. A higher fitness value ensures a greater chance of reproduction by one of these means. The population evolves over many generations until some stopping criteria is reached [27]. The algorithms are shown in Algorithms 2–4.

The best solutions from each generation are saved, and members are selected for crossover and mutation using a Roulette wheel approach in which the probability of selection is directly related to fitness. Our GA employs uniform crossover [27] as adjacent features have no relation to each other. In addition, feature set union and intersect operators were used for mating.

With this simple type of genetic algorithm, the parametrized HMEM generally performed worse than the original HMEM as a result of overfitting. The main culprit seemed to be weakness at generalizing to all subjects. Furthermore, the population tended to converge to similar feature selections so that there was not enough diversity in the final population to guarantee a good choice when testing against the cross-validation set. The solution to both problems was the NGSA algorithm, which looks at multiple optimization objectives and uses sharing to increase diversity in the population.

Algorithm 3 Mutation Operator

function ρ_{out} = **mutate**(ρ_{in})
$\langle \Psi, \Omega \rangle \leftarrow \rho_{in}$
$\text{cflip}_0 \leftarrow 0.1$ {The chance of a feature being deselected}
$\text{cflip}_1 \leftarrow 0.1$ {The chance of a feature being selected}
{Mutate the feature selection}
for all $\psi_j \in \Psi$ **do**
  if $\text{rand}(0 \rightarrow 1) \leq \text{cflip}_0$ **then**
    $\psi_j \leftarrow 0$
  else if $\text{rand}(0 \rightarrow 1) \leq \text{cflip}_1$ **then**
    $\psi_j \leftarrow 1$
  end if
end for
{Change the number of transitions in at most one cohort}
if $\text{rand}(0 \rightarrow 1) \leq 0.5$ **then** {Only a 50% chance of mutation}
  randomly select a cohort $\omega \in \Omega$
  $\omega \leftarrow$ number drawn uniformly from $\{1, \ldots, 5\}$
end if
return $\langle \Psi, \Omega \rangle$
end function

Algorithm 4 Crossover Operator

function ρ_{out} = **crossover**(ρ_a, ρ_b)
$\langle \Psi_a, \Omega_a \rangle \leftarrow \rho_a$
$\langle \Psi_b, \Omega_b \rangle \leftarrow \rho_b$
{Crossover for the feature selection}
$s \leftarrow \text{rand}(0 \rightarrow 1)$
if $s \leq \frac{1}{3}$ **then** {Select the union operator}
  $\Psi_{out} \leftarrow \Psi_a \cup \Psi_b$
else if $s \leq \frac{2}{3}$ **then** {Select the intersect operator}
  $\Psi_{out} \leftarrow \Psi_a \cap \Psi_b$
else {Uniform crossover operator}
  for $j \leftarrow 1$ to $|\Psi_a|$ **do**
    if $\text{rand}(0 \rightarrow 1) \leq 0.5$ **then**
      $\psi_{out,j} \leftarrow \psi_{a,j}$
    else
      $\psi_{out,j} \leftarrow \psi_{b,j}$
    end if
  end for
end if
{Crossover for number of transitions per cohort}
for $j \leftarrow 1$ to $|\Omega|$ **do**
  if $\text{rand}(0 \rightarrow 1)$ **then**
    $\omega_{out,j} \leftarrow \omega_{a,j}$
  else
    $\omega_{out,j} \leftarrow \omega_{b,j}$
  end if
end for
return $\langle \Psi_{out}, \Omega_{out} \rangle$
end function

2) Multiobjective GA: Single-objective optimization defines a point as optimal if there is no other point that performs better with regard to the objective function. For multiobjective optimization, a point is considered optimal if there is no point at least as good for all but one objective, and better for at least one objective. Such a point is called pareto-optimal. Multiple pareto-optimal points can exist for a problem. This collection of points forms a hypersurface known as the Pareto front. Multiobjective genetic algorithms (MOGA) are one possible approach. Oliveira *et al.* [28] suggested using multiple objectives to assign fitness for feature selection, using the NGSA algorithm detailed in [29]. Essentially, the Pareto front from the population is assigned a certain fitness value, then the Pareto front chosen by ignoring the points in the first Pareto front is assigned a lower fitness value. This repeats until all points in the population have been accounted for.

This initial method suffers from the problem that the population tends to converge to a few points on the Pareto front as the number of generations increase. NGSA reduces this problem by reducing the probability of similar solutions being chosen [29].

The objectives are minimization of total error, maximum per-subject error, and maximum per-annotation-type error. With the more diverse population of solutions, the GA-based feature selection and model parametrization outperformed the default "no feature selection." The final parametrization was chosen by selecting the individual with the highest performance for the "total error" objective on the cross-validation data. Membership in the Pareto front was determined using Yi Cao's Pareto front MATLAB script citeyicao.

3) GA Parameters: The population was initialized with 40 individuals with randomly created chromosomes. Each generation, the best 20 individuals were carried over to the new generation, up to 20 additional individuals were selected using the Roulette wheel approach (any that were the same as existing members of the population were discarded). There were 15 mutated, and 25 pairs chosen for crossover. The whole process was carried out for 40 generations. Relative to existing literature, we have a much smaller population and fewer generations. The objective function involved evaluating the actual error of the HMEM for all our training data. This took approximately 1 s per individual on a 2.4 GHz Intel Core 2 Duo iMac with 2 GB of RAM running MATLAB; therefore, running the genetic algorithm for a single sensor node took approximately 20 min. An increase in the population or generations would have made it take a prohibitively long amount of time to evaluate different scenarios.

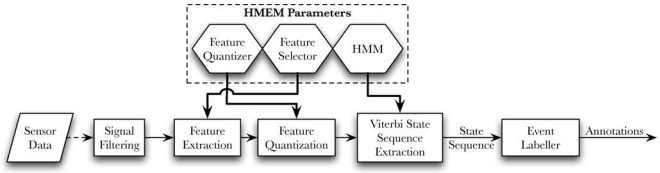


Fig. 5. Application of the HMEM as described in Section IV-F.

F. HMEM Application Procedure

After the HMEM is trained, it can be used to find key events in a data stream for the movement it has been trained on. The data flow for the algorithm is shown in Fig. 5. First, the data are filtered using the procedure described earlier, then features are extracted and quantized. Next, the feature selection is applied, and finally the most likely state sequence is extracted using the Viterbi algorithm. The annotation converter finds all the event states in this sequence and outputs an ordered set where each element consists of a time and event label.

V. Collaborative Segmentation

One of the main properties of a BSN is the capability of distributed sensing and collaboration. The design of the HMEM so far has been based on individual sensor nodes labeling the key events. An interesting question is "how much do the segmentation results improve by allowing collaboration between nodes?" A simple form of collaboration splits the nodes into a coordinator node and several operator nodes. It requires each operator node to extract the key events independently, then transmit the annotations to the coordinator node for final event labeling. This scheme has a relatively lowcommunication overhead, and the star-topography also introduces less delay than a more general directed acyclic graph (DAG) collaboration scheme might.

HMEM already provides a framework for extracting key events that can be extended for use in collaborative segmentation. All the operator nodes independently label the events using an HMEM that can observe the features extracted from their sensors. The coordinator node has an HMEM that not only observes the data from its own sensors, but also the labels assigned by the operator nodes. Since the HMEM requires an observation for each sample, samples unlabeled by a specific operator node receive a label of "0."

VI. Results

We performed the experiments as described in Section III-A. The results are reported with precision (*P*), recall (*R*), quality [root mean square error (RMSE)]. The first and last annotations were ignored because the algorithm needs context to determine annotations, and we are interested in the steady-state performance only. For each comparison, the results from the default parametrization (greedy choice of transition states, all features), and the GA feature/parameter selection are shown. We show error for each mote using just the accelerometer readings, look at per-subject error for a poor performing sensor node and a well-performing sensor node. Also, various combinations of sensors are explored.

A. Error Measures

The quality of the results can be evaluated against two primary criteria: 1) how accurately can the events be labeled?; and 2) how close in time are the accurate event labels to the actual event? Precision (*P*) and recall (*R*) can assess accuracy of labeling. Precision is the percentage of labeled events which are actual events, and recall is the percent of actual events which are labeled. The second criterion can be addressed with the RMSE of difference in time between the estimated time of correctly identified key events and the actual time those events occurred. Equations (3), (4), and (10) give precise mathematical definitions for these concepts based on true positives (tp), false positives (fp), and false negatives (fn). The functions used to define RMSE are explained in the following paragraphs:

$$P = \frac{\text{tp}}{\text{tp} + \text{fp}} \tag{3}$$

$$R = \frac{\text{tp}}{\text{tp} + \text{fn}}. \tag{4}$$

It is necessary to describe a method for determining whether an annotation is a true positive, false negative, or false positive. This method should find a mapping from the canonical to the generated annotations. Any mapped annotation is a true positive, and the unmapped ones are either false negative or false positive depending on whether they are in the generated or canonical annotation set. More formally, there is a set of canonical annotations C, and a set of generated annotations G, and an injective function $c = f(g)$ that maps generated annotations to corresponding canonical annotations. $\mathrm{dom}(f) \subseteq C$ and $\mathrm{ran}(f) \subseteq G$, and there is also a function $t(e), e \in (G \cup C)$ that determines the time the given event occurred. To define f, we first define a helper function, $h(e, E)$, which finds the closest event to the annotation e in the annotation set E.

$$h(e, E) = \min_{e_i \in E} |t(e) - t(e_i)| \tag{5}$$

$$f(g) = \begin{cases} h(g, C), & \text{if } g = h\left(h\left(g, C\right), G\right) \text{ and} \\ & \quad W \leq |t(g) - t\left(h\left(g, C\right)\right)| \\ \text{undefined}, & \text{otherwise.} \end{cases} \tag{6}$$

In other words, $f(g)$ produces a mapping between a pair of annotations c and g if the closest generated annotation to c in G is g and the closest canonical annotation to g in C is c and the temporal interval between them is less than W.

$$tp = \|\mathrm{ran}(f)\| \tag{7}$$

$$fp = \|G \cap (\mathrm{ran}(f))^c\| \tag{8}$$

$$fn = \|C \cap (\mathrm{dom}(f))^c\|. \tag{9}$$

Now that the measure for accuracy has been thoroughly addressed, a measure for quality must be presented. We use the popular RMSE.

$$\mathrm{RMSE} = \sqrt{\frac{\sum_{g \in \mathrm{dom}(f)} [t(g) - t(f(g))]^2}{tp}}. \tag{10}$$

Finally, an error measure is needed for training. The error measure should be unified (one measure that includes quality and accuracy), and should penalize a model more for false positives and false negatives than for a small deviation from the actual position. The following measure satisfies these requirements.

$$\eta = \frac{\sum_{c \in C} [t(c) - t(h(c, G))]^2 + \sum [t(g) - t(h(g, C))]^2}{2 \min(\|C\|, \|G\|)}. \tag{11}$$

B. Event Annotation Using Only Accelerometer Readings

Accelerometers are commonly available in many devices including laptops, cell phones, and PDAs. This means that if our algorithm can extract the key events using accelerometer data exclusively from positions these devices are worn, the device could be used to compute gait parameters. Because of this, many of our experimental results consider only the accelerometer readings, and discard the gyroscopic data. Since people commonly wear cell phones on their thigh, this data shows the potential for using cell phones to segment gait in the background. In Table I, features from the present sample as well as features up to five

TABLE I
RESULTS USING ONLY ACCELEROMETER WITH TP FEATURES

	Default (132 Features)			Genetic Algorithm			
Node	**P**	**R**	**RMSE**	**P**	**R**	**RMSE**	**Fsel**
Waist	99.7	99.7	2.33	100	100	2.21	26
R Arm	96.9	97.5	4.33	96.0	93.8	3.78	16
R Forearm	90.3	93.5	3.61	96.5	94.4	3.41	10
L Arm	97.2	99.9	3.25	98.3	98.5	3.01	23
L Forearm	99.6	98.6	2.57	99.6	98.8	2.40	11
R Thigh	100	100	1.97	100	100	1.92	38
R Leg	99.9	99.9	1.52	99.9	99.9	1.50	107
L Thigh	100	100	1.81	100	100	1.73	22
L Leg	100	100	2.00	100	99.6	1.85	15

TABLE II
USING ONLY ACCELEROMETER WITHOUT TP FEATURES

	Default (132 Features)			Genetic Algorithm			
Node	**P**	**R**	**RMSE**	**P**	**R**	**RMSE**	**Fsel**
Waist	99.8	99.8	2.29	99.9	99.5	2.36	5
R Arm	96.3	96.9	4.23	97.8	96.8	2.93	5
R Forearm	91.1	94.4	3.65	95.2	92.8	3.74	4
L Arm	99.0	99.8	3.48	98.9	96.3	2.92	6
L Forearm	97.9	99.5	3.15	99.5	98.3	2.64	9
R Thigh	99.0	99.9	2.25	99.9	99.5	2.06	2
R Leg	99.9	99.9	1.77	100	99.6	1.49	4
L Thigh	100	100	1.97	100	100	1.95	9
L Leg	99.5	99.9	2.43	100	99.6	1.98	2

samples in the future and five samples in the past are used. These are called temporally proximate (TP) features. The results show the best performance on the legs and thighs, with decent performance on the waist. When looking at diseases that can influence inter-leg coordination, each sensor node should be trained to look for temporal parameters related to the limb on which it relies. In this case, all sensor nodes are trying to extract parameters from the right leg. Another interesting thing is that the arms even do a reasonable job, indicating relatively strong coordination between the arms and the legs. Finally, it is clear that using genetic algorithms improves performance and reduces the number of features. In most cases, the performance increase is modest, but the number of features is drastically reduced, which can lead to a significant performance increase.

Table II shows the results from the same sensors without the use of temporally proximate features. The performance is noticeably degraded from the "TP" case, but the advantage is that no feature buffer is needed, thus decreasing the time it takes to produce an answer. Once again the GA produces lower error for most cases with fewer features.

C. Examination of Per-Subject Error

One of the goals of the HMEM system is good generalization to new subjects. Table III shows per-subject error for the sensor on the right thigh. Initially subjects 2–4 were in training, 5–7 in cross-validation, and 8–9 in test. However, subjects 5 and 7 have walking patterns that differ significantly from the others, but are similar to each other. Therefore, subjects 4 and 5 were exchanged. It is likely that with a larger dataset the system could generalize better to such subjects. All the results are shown from the portion of the subjects' data that was in the test dataset.

The sensor on the right thigh, as shown in Table III, performs well. Subjects 8 and 9 perform a little worse than subjects in

TABLE III
SUBJECT ERROR FOR R THIGH WITH ACCEL AND TP

	Default (132 Features)			GA (38 Features)		
Subject	**P**	**R**	**RMSE**	**P**	**R**	**RMSE**
Sub 2 Train	100	100	1.44	100	100	1.62
Sub 3 Train	100	100	1.33	100	100	1.11
Sub 4 Cross	100	100	1.36	100	100	1.38
Sub 5 Train	100	100	3.54	100	100	3.43
Sub 6 Cross	100	100	1.03	100	100	1.02
Sub 7 Cross	100	100	3.25	100	100	3.09
Sub 8 Test	100	100	1.74	100	100	1.76
Sub 9 Test	100	100	1.55	100	100	1.66

TABLE IV
CROSS-VALIDATED SUBJECT ERROR (R THIGH)

Subject	**P**	**R**	**RMSE**	**Fsel**
Sub 2	100	100	1.47	6.7
Sub 3	100	100	1.38	8.7
Sub 4	100	100	1.68	4.7
Sub 5	99.8	99.8	3.55	4.7
Sub 6	100	100	1.20	6.7
Sub 7	100	100	3.09	28.7
Sub 8	100	100	1.52	10.7
Sub 9	100	100	1.59	16.0

TABLE V
SENSOR TYPES WITH TP FEATURES ON RIGHT THIGH

	Default			Genetic Algorithm			
Set	**P**	**R**	**RMSE**	**P**	**R**	**RMSE**	**Fsel**
Accel (132)	100	100	1.97	100	100	1.92	38
Gyro (88)	100	100	2.14	100	100	2.14	9
All (220)	100	100	1.84	100	100	1.84	33
Acc Mag (44)	99.4	99.8	2.02	99.9	100	1.97	5

the training and cross-validation sets. The worst per-subject error comes from subjects 5 and 7. The reason for this is not entirely clear. The use of the GA does not significantly reduce the discrepancy in per subject error. Since the final selection criteria for the solution is minimum total error, not minimum worst-case subject error, this is not surprising.

Manual partitioning of the data into training, cross-validation, and testing sets can artificially bias the results. Therefore, we performed an experiment for each subject where the subject was placed exclusively in the testing set, and the training and cross-validation sets were selected randomly from the remaining subjects. The results shown in Table IV are the average of three tests after the GA. These results demonstrate that the model has good generalization to many subjects, but performs poorly on some. It would be interesting as a future work to investigate the features of those subjects that cause the model to perform poorly.

D. Exploration of Different Sensor Types

Another goal for HMEM is the ability to use new sensors and combination of sensors without having to develop new methods to extract the key events. To simulate this, we examine the HMEM trained with different subsets of sensors, as shown in Table V. The sensor types considered were:

1) accelerometer only;
2) gyroscope only
3) all sensors; and
4) just the magnitude from the accelerometer.

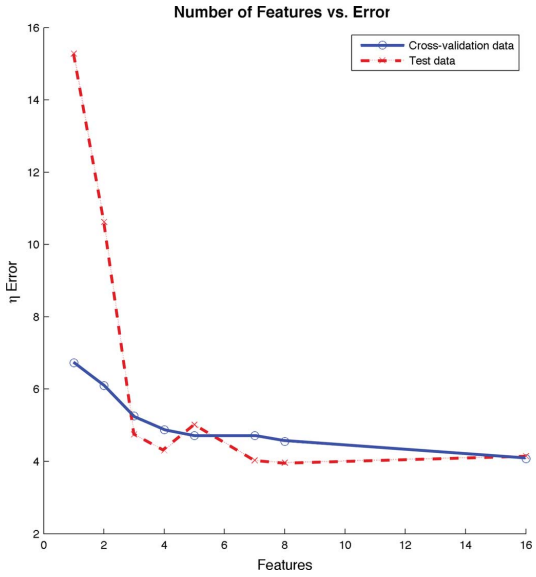


Fig. 6. Number of features versus error for the right thigh.

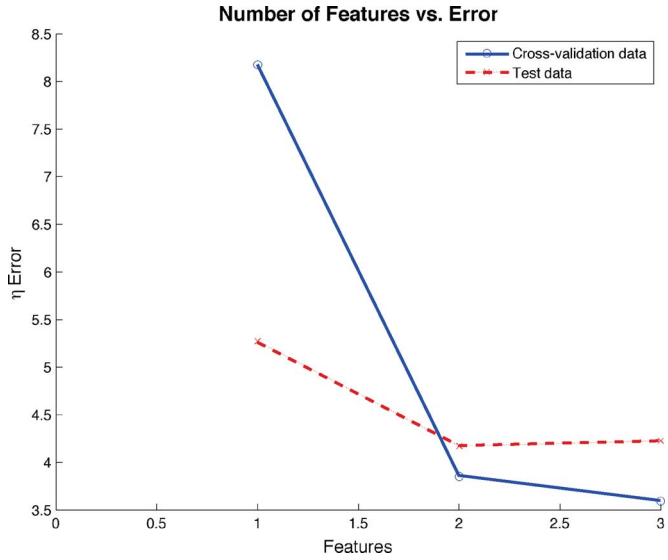


Fig. 7. Features versus error for the right thigh with expansive elitism.

The accelerometer performs better than the gyroscope and a combination of all the sensors has the best performance. However, the most interesting test is using just the magnitude of the accelerometer. The magnitude of the accelerometer would be invariant for any rotation of the sensor. This could be especially important if the sensor is a cellphone in the subject's pocket. Even though exclusive reliance on the magnitude of acceleration gives the worse results, the performance is still reasonable.

E. Explicit Feature Reduction

Feature selection can be used to explicitly reduce the number of features. The algorithmic time for the HMEM event annotation algorithm increases linearly with the number of features; so, feature reduction can improve performance considerably. The NSGA framework can be used as described earlier with the objectives of global error minimization and feature minimization.

The two objectives imply a two-dimensional Pareto front. We take the population at the final generation and select the Pareto front when the population error is judged using the cross-validation set. The same trained models are then judged against the test set. The results from both the test and cross-validation sets are shown in Figs. 6 and 7. The error is reported using the training error η, which is approximately equal to the square of the RMSE. Fig. 6 shows the results where each generation saves just the Pareto front for the training data. The GA used to generate Fig. 7 saved several successive fronts so that at least 20 of the best were saved each generation, resulting in lower error. Further, using just the two features found in Fig. 7 results in performance almost as good as with no feature reduction. The starting number of features was 132; so, this results in a performance increase of approximately 66 times.

Moreover, while the performance on the cross-validation set and the test set are different, both have an "elbow" at the same place, where the error increases drastically with an increased number of features. This suggests that an effective way of picking the best HMEM is to pick the HMEM right before the cross-validation error starts increasing significantly.

TABLE VI
RESULTS FROM COLLABORATIVE EVENT LABELING

	Default (220 Features)			Genetic Algorithm			
Coordinator	**P**	**R**	**RMSE**	**P**	**R**	**RMSE**	**Fsel**
Waist	100	100	1.70	100	100	1.46	74
R Arm	99.1	100	1.61	100	100	1.40	61
R Forearm	98.7	100	1.55	100	100	1.36	124
L Arm	99.7	100	1.65	100	100	1.33	46
L Forearm	99.4	100	1.66	100	100	1.38	26
R Thigh	100	100	1.59	99.8	100	1.34	27
R Leg	100	100	1.33	100	100	1.30	116
L Thigh	100	100	1.55	100	100	1.34	61
L Leg	100	100	1.64	100	100	1.38	17

F. Collaborative Event Extraction

The collaborative scheme outlined in Section V requires one node to be chosen as the coordinator, and the rest become operator nodes. We ran an experiment for each possible choice of the coordinator nodes and show the results. The data fed into each node come from using just the accelerometer and TP features without GA parametrization. Results are shown in Table VI for a coordinator using default parametrization as well GA parametrization to select features.

These results exhibit several interesting properties. The results for any mote as coordinator are not guaranteed to be as good as just picking the results from the best operator node. However, the quality of the results obtained by using a given node as the coordinator are always better than using the results from that node individually. Furthermore, the ranking of each of each of the nodes as coordinator is the same as the independent ranking. This suggests that a good strategy for picking the coordinator node is to pick the best individual node and make it the coordinator.

When using GA parametrization, these patterns no longer hold true. There is no longer a predictable correlation between individual rank and coordinator rank. In all but one case, the re-

sults of collaboration are better than the best performance of any individual node. The best performance from a GA parametrized node is not significantly better than the best performance from a default parametrization, but the average performance is much better.

Using a collaborative scheme poses communication overhead with associated higher power costs. The improvements even for the best individual node are large, however, not all problems may require that high precision. It also is reasonable to assume that a sensor node mounted on the right shin or right thigh is picking up cues from the sensors directly related to right leg movement. However, a sensor mounted on the left leg or left arm is likely picking up cues related to limb coordination in the training population. If a system's goal is to label events accurately even if the subject has unusual limb coordination, such as in a disease scenario, then relying on specific types of coordination could introduce fairly error compared with just using results from a sensor mounted on the right leg.

VII. Conclusion and Future Work

The HMEM was designed to label key temporal events within an action. It is meant to be trainable on a wide variety of sensor types and a wide variety of actions, and to generalize well to new subjects not encountered during training. We proposed using algorithms and node collaboration to further improve the accuracy of the system. The results demonstrate that this model effectively meets the design goals. One surprise was that the GA, improved accuracy, but not dramatically. Instead, the GA appears to be effective when applied to feature reduction goals.

Another factor in the design was algorithmic simplicity so that the HMEM could be reasonably implemented on a sensor node. Our research efforts are currently focused on porting a version of HMEM to the TelosB platform.

The initial goal of our research was to see how effective an HMM-based solution would be at event labeling. Specifically, given the knowledge that the subject is walking, the HMEM can do stride labeling. The next important step is determining whether or not the subject is walking. HMMs have long been used in this way for continuous speech recognition, and we believe, we can adapt some of their approaches to the problem of human movement monitoring. It is also important that we determine what scenarios would cause a trained HMEM to be ineffective. For example, would an HMEM trained on a set of young people be able to accurately extract walking events on older subjects?

We believe that HMEM provides an important set of features for human movement monitoring. Its simplicity and generalizability to new sensors and new subjects make it attractive for many problems, and with some improvements it can be used for event recognition as well as event labeling.

References

[1] S. Newell and E. Els, *The Golf Instruction Manual*. London, U.K.: Dorling Kindersley, 2001.

[2] J. Perry, *Gait Analysis: Normal and Pathological Function*. Thorofare, NJ: SLACK Incorporated, 1992.

[3] J. Hausdorff, M. Cudkowicz, R. Firtion, J. Wei, and A. Goldberger, "Gait variability and basal ganglia disorders: Stride to-stride variations of gait cycle timing in Parkinson's disease and Huntington's disease," *Movement Disord.*, vol. 13, no. 3, pp. 428–437, 1998.

[4] J. Hausdorff, D. Rios, and H. Edelberg, "Gait variability and fall risk in community-living older adults: A 1-year prospective study," *Archives Phys. Med. Rehabil.*, vol. 82, no. 8, pp. 1050–1056, 2001.

[5] K. Aminian and B. Najafi, "Capturing human motion using body-fixed sensors: Outdoor measurement and clinical applications," *Comput. Animation Virtual Worlds*, vol. 15, no. 2, pp. 79–94, 2004.

[6] F. Axisa, P. Schmitt, C. Gehin, G. Delhomme, E. McAdams, and A. Dittmar, "Flexible technologies and smart clothing for citizen medicine, home healthcare, and disease prevention," *IEEE Trans. Inf. Technol. Biomed.*, vol. 9, no. 3, pp. 325–336, Sep. 2005.

[7] R. Jafari, F. Dabiri, P. Brisk, and M. Sarrafzadeh, "Adaptive and fault tolerant medical vest for life-critical medical monitoring," in *Proc. 2005 ACM Symp. Appl. Comput.*. New York, NY: ACM, 2005, pp. 272–279.

[8] R. Dickstein, N. Abulaffio, I. Gelernter, and T. Pillar, "An ultrasonic-operated kinematic measurement system for assessment of stance balance in the clinic," *Clin. Biomech.*, vol. 11, no. 3, pp. 173–175, 1996.

[9] F. Raab, E. Blood, T. Steiner, and H. Jones, "Magnetic position and orientation tracking system," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 15, no. 5, pp. 709–718, Sep. 1979.

[10] J. Aggarwal and Q. Cai, "Human motion analysis: A review," in *Proc. IEEE Nonrigid Articulated Motion Workshop*, 1997, pp. 90–102.

[11] N. Vaswani and A. Chowdhury, "Activity recognition using the dynamics of the configuration of interacting objects," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog. (CVPR 2003)*, pp. 633–640.

[12] H. Nait-Charif and S. McKenna, "Activity summarisation and fall detection in a supportive home environment," in *Proc. 17th Int. Conf. Pattern Recog. (ICPR 2004)*, vol. 4, pp. 323–326.

[13] L. Lee and W. Grimson, "Gait analysis for recognition and classification," in *Proc. 5th IEEE Int. Conf. Autom. Face Gesture Recog.*, 2002, pp. 148–155.

[14] A. Sabatini, C. Martelloni, S. Scapellato, F. Cavallo, S. Sant'Anna, and I. Pisa, "Assessment of walking features from foot inertial sensing," *IEEE Trans. Biomed. Eng.*, vol. 52, no. 3, pp. 486–494, Mar. 2005.

[15] T. Sakaguchi, T. Kanamori, H. Katayose, K. Sato, and S. Inokuchi, "Human motion capture by integrating gyroscopes and accelerometers," in *Proc. IEEE/SICE/RSJ Int. Conf. Multisensor Fusion Integr. Intell. Syst.*, 1996, pp. 470–475.

[16] R. Zhu and Z. Zhou, "A real-time articulated human motion tracking using tri-axis inertial/magnetic sensors package," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 12, no. 2, pp. 295–302, Jun. 2004.

[17] S. Miyazaki, "Long-term unrestrained measurement of stride length and walking velocity utilizing a piezoelectric gyroscope," *IEEE Trans. Biomed. Eng.*, vol. 44, no. 8, pp. 753–759, Aug. 1997.

[18] K. Aminian, B. Najafi, C. Büla, P. Leyvraz, and P. Robert, "Spatio-temporal parameters of gait measured by an ambulatory system using miniature gyroscopes," *J. Biomech.*, vol. 35, no. 5, pp. 689–699, 2002.

[19] T. Pfau, M. Ferrari, K. Parsons, and A. Wilson, "A hidden Markov model-based stride segmentation technique applied to equine inertial sensor trunk movement data," *J. Biomech.*, vol. 41, no. 1, pp. 216–220, 2008.

[20] M. Quwaider and S. Biswas, "Body posture identification using hidden Markov model with a wearable sensor network," presented at the ICST 3rd Int. Conf. Body Area Netw. Table Contents. Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, Brussels, Belgium, 2008.

[21] R. Duda, P. Hart, and D. Stork, *Pattern Classification*, 2nd ed. New York: Wiley, 2001.

[22] L. Rabiner and B. Juang, "An introduction to hidden Markov models," *IEEE ASSP Mag.*, vol. 3, no. 1, Part 1, pp. 4–16, Jan. 1986.

[23] L. Rabiner, "A tutorial on hidden Markov models and selected applications inspeech recognition," *Proc. IEEE*, vol. 77, no. 2, pp. 257–286, Feb. 1989.

[24] N. Liu, B. Lovell, and P. Kootsookos, "Evaluation of HMM training algorithms for letter hand gesture recognition," in *Proc. 3rd IEEE Int. Symp. Signal Process. Inf. Technol. (ISSPIT 2003)*, pp. 648–651.

[25] S. Raudys and A. Jain, "Small sample size effects in statistical pattern recognition: Recommendations for practitioners," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 13, no. 3, pp. 252–264, Mar. 1991.

[26] G. Rudolph, "Convergence analysis of canonical genetic algorithms," *IEEE Trans. Neural Netw.*, vol. 5, no. 1, pp. 96–101, Jan. 1994.

[27] M. Srinivas, L. Patnaik, "Genetic algorithms: A survey," *Computer*, vol. 27, no. 6, pp. 17–26, 1994.

[28] L. Oliveira, R. Sabourin, F. Bortolozzi, and C. Suen, "Feature selection using multi-objective genetic algorithms for handwritten digit recognition," in *Proc. 16th ICPR*, 2002, pp. 568–571.

[29] N. Srinivas, K. Deb, "Muiltiobjective optimization using nondominated sorting in genetic algorithms," *Evol. Comput.*, vol. 2, no. 3, pp. 221–248, 1994.

Eric Guenterberg (S'07) received the B.S. degree in electrical engineering from the University of California, Los Angeles, in 2007. He is currently working toward the Ph.D. degree in electrical engineering at the University of Texas at Dallas, Dallas.

He is a Research Assistant at the Embedded Systems and Signal Processing Laboratory, Department of Electrical Engineering, University of Texas at Dallas. His current research interests include primarily distributed pattern recognition and signal processing with an emphasis on medical/biological applications.

Allen Y. Yang (M'07) received the B.Eng. degree in computer science from the University of Science and Technology of China (USTC), Hefei, China, in 2001, the M.S. degree in electrical engineering, in 2003, the M.S. degree in mathematics, in 2005, and the Ph.D. degree in electrical and computer engineering, in 2006, all from the University of Illinois at Urbana-Champaign (UIUC), Urbana.

Currently, he is a Research Scientist at the Department of Electrical Engineering and Computer Science, University of California, Berkeley. His current research interests include pattern analysis of geometric and statistical models in very high-dimensional data spaces, and applications in motion segmentation, image segmentation, face recognition, and signal processing in heterogeneous sensor networks. He is the author of six journal papers and more than ten conference papers. He is the co-inventor of three US patent applications.

Dr. Yang received the Best Bachelor's Thesis Award from the USTC and the Henry Ford II Scholar Award from the UIUC.

Hassan Ghasemzadeh (S'07) received the B.Sc. degree in computer engineering from Sharif University of Technology, Tehran, Iran, and the M.Sc. degree in computer engineering from the University of Tehran, Tehran, in 1998 and 2001, respectively. Currently, he is working toward the Ph.D. degree in computer engineering at the University of Texas at Dallas, Dallas, where his research is focussed on collaborative signal processing, reconfigurable computing, and algorithm design for medical embedded systems.

He was with Azad University, Damavand, Iran, where he served as a Faculty Member and the Director of undergraduate studies. His research interests include different aspects of embedded systems.

Roozbeh Jafari (M'06) received the B.Sc. degree in electrical engineering from Sharif University of Technology, Tehran, Iran, in 2000. He received the M.S. degree in electrical engineering from the State University of New York at Buffalo, Buffalo, the M.S. and Ph.D. degrees in computer science from the University of California, Los Angeles, in 2002, 2004, and 2006, respectively.

From 2006–2007, he was a Post-Doctoral Researcher with the Department of Electrical Engineering and Computer Sciences, University of California, Berkeley Currently, he is an Assistant Professor and Director at the Embedded Systems and Signal Processing Laboratory, Department of Electrical Engineering, University of Texas at Dallas, Dallas. His research interests include networked embedded system design and reconfigurable computing with emphasis on medical/biological applications, their signal processing and algorithm design.

Ruzena Bajcsy (M'81–SM'88–F'92) received the masters and Ph.D. degrees in electrical engineering from Slovak Technical University, Bratislava, Slovakia, in 1957 and 1967, respectively, and the Ph.D. degree in computer science from Stanford University, Palo Alto, CA, in 1972. She received the honorary doctorate degree from the University of Ljubljana, Ljubljana, Slovenia and from Lehigh University, Bethlehem, PA, in 2001.

She was the Director of Center for Information Technology Research in the Interest of Society and Professor at the Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, during 2001. During 1998–2001, she was an Assistant Director at the Computer Information Science and Engineering Directorate (CISE), Arlington, VA. As Head of National Science Foundations (NSF) CISE Directorate, she managed a $500 million annual budget. She joined NSF from the University of Pennsylvania, Philadelphia, where she was a Professor of Computer Science and Engineering since 1972. Currently, she is with the Department of Electrical Engineering and Computer Sciences, University of California. She has done seminal research in the areas of human-centered computer control, cognitive science, robotics, computerized radiological/medical image processing, and artificial vision.

Dr. Bajcsy is a Member of the National Academy of Engineering, as well as the Institute of Medicine. She is a recipient of the Franklin Medal 2009.

S. Shankar Sastry (F'94) received the M.A. degree (honoris causa) from Harvard University, Cambridge, MA, in 1994, and the Ph.D. degree from the University of California, Berkeley, in 1981.

From 1980 to 1982, he was an Assistant Professor.at Massachusetts Institute of Technology (MIT), Cambridge. He was a Chaired Gordon Mc Kay Professor at Harvard University, Cambridge, MA, during 1994. From 2001 to 2004, he was the Chairman of the Department of Electrical Engineering and Computer Science (EECS), University of California, Berkeley. In 2000, he was the Director of the Information Technology Office at Defense Advanced Research Projects Agency, Arlington, VA. He is the NEC Distinguished Professor of EECS and a Professor of Bioengineering. Currently, he is the Director of Center for Information Technology in the Interests of Society, University of California.

Dr. Sastry received the President of India Gold Medal in 1977, the IBM Faculty Development Award for 1983–1985, the US National Science Foundation Presidential Young Investigator Award in 1985, the Eckman Award of the American Automatic Control Council in 1990, the distinguished Alumnus Award of the Indian Institute of Technology, in 1999, and the Ragazzini Award for Excellence in Education by the American Control Council in 2005. He is on the US Air Force Science Board and is Chairman of the Board of the International Computer Science Institute.