Structural Action Recognition in Body Sensor Networks: Distributed Classification Based on String Matching

Hassan Ghasemzadeh, *Student Member, IEEE,* Vitali Loseu, *Student Member, IEEE,*
Roozbeh Jafari, *Member, IEEE,*

***Abstract*—Mobile sensor-based systems are emerging as promising platforms for healthcare monitoring. An important goal of these systems is to extract physiological information about the subject wearing the network. Such information can be used for life-logging, quality of life measures, fall detection, extraction of contextual information, and many other applications. Data collected by these sensor nodes are overwhelming, and hence an efficient data processing technique is essential. In this paper, we present a system using inexpensive, off-the-shelf inertial sensor nodes that constructs motion transcripts from biomedical signals and identifies movements by taking collaboration between the nodes into consideration. Transcripts are built of motion primitives and aim to reduce the complexity of the original data. We then label each primitive with a unique symbol and generate a sequence of symbols, known as motion template, representing a particular action. This model leads to a distributed algorithm for action recognition using edit distance with respect to motion templates. The algorithm reduces the number of active nodes during every classification decision. We present our results using data collected from five normal subjects performing transitional movements. The results clearly illustrate the effectiveness of our framework. In particular, we obtain a classification accuracy of 84.13% with only one sensor node involved in the classification process.**

***Index Terms*—Body Sensor Networks, Physical Movement Monitoring, Collaborative Signal Processing, Motion Primitives, Distributed Computing.**

I. Introduction

ADVANCES in wireless communication, sensor design and microelectronics have enabled the development of tiny sensor platforms that can be integrated with the physical environment of our daily lives. The new generation of wireless sensor networks, formally known as body sensor networks (BSNs), is promising to revolutionize healthcare system by providing continuous and ambulatory health monitoring. They find applications in rehabilitation, sports medicine, geriatric care, gait analysis, and balance evaluation.

Many movement monitoring applications require knowledge of what movement the subject is performing. This knowledge can be divided into three categories based on the level of abstraction of the conclusion: 1) motion, 2) action, and 3) activity. The most tangible category is the motions which represents the position, velocity, and acceleration of all body parts at a given time. These characteristics can be directly observed by BSNs; however they do not provide a high enough level of abstraction for realistic applications. Actions belong to a higher level category, and refer to the basic motion sequences or static postures. Actions are generally sequential and rather consistent; examples include standing, moving from sitting to standing, walking, and jumping. While actions provide more information than motions, they lack realization of intelligent intent in human behavior. This role is filled with the highest level of motion abstraction called activity. Common activities include cooking, talking with friends, teaching, and brushing teeth.

The additive hierarchical representation of human movements is very similar to the representation of human speech: raw sounds are divided into phonemes, which are further grouped into words, which are grouped into sentences [1]. Phonology exclusively focuses on sound, ignoring physical movement of the tongue and throat and cues from facial expressions. Similarly, raw sensor data can be used to build sequences of motions, which can be further grouped into actions and then activities.

We are primarily concerned with the accuracy of action recognition while respecting the inherent limitations of our sensing platform. In BSNs, sensor nodes are usually arranged in a star topology with a base station responsible for processing sensing data. Approaches that make this assumption employ centralized algorithms with the base station as the coordinator to reduce computational stress on individual sensor nodes. This can result in nodes forwarding a significant amount of data to the base station for signal processing. There are two problems with this approach. First, communication generally consumes more energy than local computation [2]. From the energy preservation point it is more beneficial to have signal processing on individual nodes. Second, the level of interference in wireless networks depends on the amount of data that needs to be transferred [3]. Thus, decreasing the amount of data sent over the network can lower the number of retransmissions and increase the system lifetime. Both of those factors warrant the need for creating a distributed model, where nodes classify test data locally and make the overall decision based on a subset of local decisions.

This study presents a novel framework for action recognition based on the edit distance concept. The framework relies on motion transcripts. Each movement is divided into several segments each with a consistent physical pattern. The algorithm to create motion transcripts will maintain temporal

Authors are with the Department of Electrical Engineering at the University of Texas at Dallas, Richrdson, TX 75080 USA (phone: 972-259-7816, email: {h.ghasemzadeh, vitali.loseu, rjafari}@utdallas.edu).

and structural properties of the observed sensor readings. This representation of human movements is used for local classification. A distributed algorithm is then utilized that produces a global classification decision based on a subset of results generated by individual sensor nodes. The distributed nature of the algorithm along with the compact representation of movements using transcripts enable our system to lower the amount of information stored at individual nodes, and to minimize the amount of data passed in the network. Furthermore, with the dynamic selection of the nodes needed for classification, the overall number of active nodes is reduced.

II. Related Work

Reducing the amount of active nodes is a common approach for power optimization and wearability enhancement in BSNs. As recognition systems grow in size and expand to distinguish many physical actions, so grows the number of sensors that those systems require. However, only a subset of sensors is needed to recognize most of the individual actions. Keeping other sensors operational while their input is not considered by the system is wasteful. Zappi et al. [4] propose to optimize the system energy consumption by selecting the required subset of sensors with the help of the meta-classifier sensor fusion. As a result sensors are awakened only when their input is needed to satisfy correctness property. In [5], Ghasemzadeh et al. formulate coverage problem in the context of movement monitoring using inertial on-body sensors. Their technique focuses on the minimum number of nodes that produces full action coverage set. Another way to reduce the number of active nodes is to keep track of the performed motions and pay attention only to a subset of sensors that can observe transition out of the current motion [6]. Previous studies done in [7] show that only a few sensor nodes are required to correctly classify a small action set. An additional sensor node needs to be added to the system if a new action is added to the action set and it cannot be uniquely identified by existing sensors. While it is easy to analyze any given action set and come up with an optimal number of sensors and sensor placement, the task is not trivial for a generic action set. To potentially be able to classify a large number of actions and keep the number of active nodes low a distributed classification scheme can be employed. Instead of collecting information from all of the sensor nodes to make the final decision a distributed algorithm collects information only from a subset of sensor nodes.

The concept of primitives has provided an efficient representation of human movements both in computer vision and in wireless sensor domains. Using motion primitives as building blocks, Guerra-Filho et al. [8] study decomposing angles of body segments, calculated from cameras, into a well-representative language called HAL (Human Activity Language). As another example, authors in [9] investigate construction of context-dependent grammar known as DCG (Discrete Clause Grammar) by combining atomic motions. DCGs enable rules to be formed using simple logic statements. The authors form a hierarchy of abstraction that begins with feature extraction and uses unsupervised classification at each step to group lower-level primitives into higher-level primitives. The idea of unsupervised learning in a recognition system based on motion primitives is also discussed in [10, 11], where authors try to identify action primitives from motion capture data. Finally, authors in [12] introduce a statistical technique for synthesizing walking patterns where the motion is expressed as a sequence of primitives extracted using a Hidden Markov Model (HMM). To simplify computation further primitives can be represented as string templates. This idea is explored in [13] where authors use edit distance to distinguish between motion primitive in 3D movement classification task. Similar idea is used in [14] where authors use edit distance for action/posture classification in a BSN. The authors further argue in favor of this approach by stating that only integer arithmetic operations are required for successful algorithm deployment, making it suitable to run on sensor nodes.

TABLE I
Commonly Used Terms

Name	Symbol	Definition
Action	A_j	A transitional movement observed by the system
Observation	O_{ij}	A specific view of action A_j by node s_i
Primitive		Basic set of motions defined by grouping similar signal readings
Cluster		Set of signal readings that have consistent physical behavior representing a primitive
Alphabet	$\sum_i$	A set of symbols assigned to primitives at each node s_i
Transcript	T_{ij}	A sequence of motion primitives assigned to action A_j by node s_i
Template	TPL_j	A concatenation of transcripts of different nodes assigned to action A_j
Class	C_{ij}	Set of observations of the same action A_j made by nodes s_i

We propose a concept of combining primitives extracted from inertial data into transcripts that maintain temporal and structural properties of the observed sensor readings. Based on properties extracted from edit distance calculation, we define a novel distributed algorithm for action recognition. To the best of our knowledge no previous work has been done on development of a distributed classification algorithm based on the properties of motion transcripts.

III. System Overview

In this section, we briefly describe the architecture of our system and signal processing flow for action recognition. Table I defines some of the terms that are used throughout this paper.

A. Sensing Platform

Our system consists of several XBow® TelosB [15] motes with custom-designed motes. Each sensor board has a tri-axial accelerometer and a bi-axial gyroscope. The accelerometers are LIS3LV02DQ with 1024 *LSb/g* sensitivity and are used in $2g$ mode for the experiments. The IDG-300 gyroscopes used for this study have a $2\ mV/\circ/s$ sensitivity. Each node is powered by a Li-Ion battery and samples the sensors at a certain rate, performs local processing and can transmit collected data wirelessly to other nodes. In particular, each mote can send the data to a base station. For our experiments, the base station is a node without a sensor board that forwards the data to a PC via USB. Furthermore, two Logitech® webcams are used to record video of all trials. The video

is used only as a gold standard to mark the start and end times associated with movements. For the prototype that will be developed in this paper, the sensor readings and video are collected and synchronized in MATLAB.

B. Signal Processing

Our signal processing consists of the following steps.

Data collection: Data from each body-worn sensor is obtained at 50 Hz. The sampling rate is chosen to satisfy the Nyquist criterion [16].

Preprocessing: The data collected at each node is locally filtered using a five-point moving average to reduce high frequency noise [5].

Segmentation: The signal is partitioned into segments that represent a complete action [17].

Feature extraction: Features are extracted from a small moving window centered about each point of the signal segment. The features include *mean*, *standard deviation*, *root mean square*, *first* and *second derivatives*. Intuition behind choosing this set of features is that the aforementioned features are computationally inexpensive that can be executed on our lightweight sensor nodes. Furthermore, their effectiveness in capturing structural patterns of motion data has been previously established by experimental studies [18].

Primitive construction: Each point is clustered based on the features calculated for the window surrounding it. Each cluster represents a movement primitive.

Per-node transcript generation: A transcript is built by noting where each primitive begins and ends based on the membership of the data points to a cluster. The transcript is then transformed into a sequence of characters over a finite alphabet.

Template generation and pattern recognition: Transcripts generated by individual nodes are sent to the base station where a template is generated for each trial by combining transcripts received from all sensor nodes. A central classifier is then built which makes a global decision on the current movement occurred in the system. Deployment of a central classifier is not efficient in terms of communication power and bandwidth. Despite its inefficiency, we will explore certain properties within the central classification strategy which would enable the development of an effective and fast distributed algorithm.

IV. Motion Transcripts

A physical movement can be divided into a sequence of several smaller motions. A transcript of this movement, with regard to the motions, would record order and timing of the motions. For example, a transcript for the foot during walking could consist of 1) lifting the foot, 2) moving the foot forward, 3) placing the foot on the ground, and 4) bearing weight on the foot, with certain periods of time associated with each primitive. The pattern repeats as long as walking continues. At the same time, a transcript for the hip consists of 1) rotate clockwise, 2) rotate counterclockwise, repeatedly.

Movement transcripts consist of adjacent, non-overlapping segments labeled as a particular motion primitive. One way to generate movement transcripts is to independently label each sample as a given motion primitive. We determine the characteristics for each data point in our signal by extracting features described in Section III-B from a moving window centered about the current point. The motion primitives should be found without specific knowledge of the movements, but based on patterns in the signal. Lack of prior knowledge of the structure of the dataset makes construction of primitives a challenging task. A well-studied technique for grouping similar observations is clustering [19]. We use clustering analysis to group data points with consistent features to form a primitive.

Our model employs two steps for generating movement transcripts: 1) Clustering of each data point in a movement to find the set of primitives 2) Labeling to map each primitive to a character over an alphabet. In the following subsections, we elaborate on these steps.

A. Primitive Construction and Labeling

Clustering deals with the problem of finding patterns in a dataset in an unsupervised manner. Data points (represented by a feature vector) in a cluster are similar and points in different clusters are distinct. Several clustering methods such as K-means [20], hierarchical [21] and probabilistic model based [22] clustering have been developed. The K-means algorithm starts by selecting K centroids, either randomly or using heuristic initialization. It assigns each point to a cluster it is nearest to and recalculates the cluster centroids. It repeats the previous procedure until some convergence criterion is met. In agglomerative hierarchical clustering, each data point is initially considered as a cluster. At each stage of the algorithm similar clusters are grouped together based on some distance measure. On the other hand, model based clustering [23] assumes that data are generated by a mixture of probability distributions in which each component represents a different cluster. In particular, Gaussian Mixture Models (GMM) creates clusters by representing the probability density function of the data points as a mixture of multivariate Gaussian distribution. GMM is a powerful probabilistic model that has been widely used in speech processing. Because our transcript generation approach is similar to speech processing, we use GMM to define the primitives from a set of training actions.

Primitives of movements are created by mapping extracted features into K clusters. The kth primitive is associated with a cluster ω_k in the model which has a mean vector μ_k. Each cluster generates data from a Gaussian with mean μ_k and covariance matrix $\sigma_k^2 I$. Given an observation O_i (ith feature vector), GMM finds the cluster corresponding to that vector. It computes $\Re_{ik}$, the probability of the cluster k's responsibility for accommodating observation O_i. This probability is given by (1).

$$\Re_{ik} = \mathrm{P}(k|O_i) = \frac{\mathrm{P}(O_i|k)\mathrm{P}(k)}{\mathrm{P}(O_i)} \tag{1}$$

where $\mathrm{P}(O_i|k)$ is the Gaussian function for cluster k and is defined by

$$\mathrm{P}(O_i|k) = g(O_i; \mu_k, \sigma_k) \tag{2}$$

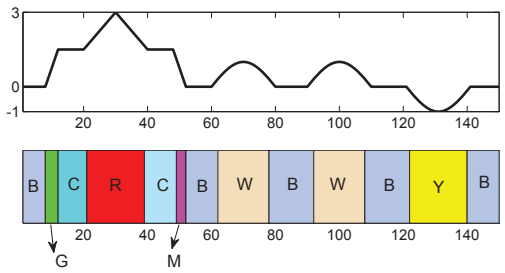


Fig. 1. An example of motion transcripts generated for a one-dimensional synthetic signal.

and P(O_i) represents the prior probability that can be calculated by marginalization of joint probabilities. P(k) is the mixing parameter for component k in the model which is equal to the number of observations belong to that cluster divided by number of all observations. This value is given by

$$\mathrm{P}(k) = \frac{\sum_i (\pi_i; i = k)}{\sum_i \pi_i} \tag{3}$$

where π_i is a binary value indicating membership for observation O_i.

Therefore, the responsibility probability can be written as

$$\Re_{ik} = \frac{g(O_i; \mu_k, \sigma_k)\mathrm{P}(k)}{\sum_k \mathrm{P}(O_i, k)} \tag{4}$$

and calculated for each observation, using a combination of Gaussian and mixing parameters. The process can be repeatedly executed to assign probability to all observations.

We use Expectation Maximization (EM) [24] to find the parameters of the mixture model. This algorithm is iterative and has two steps: *expectation* and *maximization*. The algorithm starts with initial guess for parameters of the mixing model. In the expectation step, partial membership of each observation is computed using expected measures for the membership values of each observation. The probability $\Re_{ik}$ is then calculated based on estimated parameters. In the maximization step, the Gaussian and mixing parameters are updated by maximizing the *Maximum Likelihood* function. As the number of components is unknown *a priori*, we perform multiple runs of the EM algorithm for different values of K. The optimal number of clusters and the problem of choosing the best model are evaluated based on the Bayesian Information Criterion [23]. Each data point can be assigned to a primitive by selecting the cluster that maximizes the posterior probability. We construct a transcript of movement by noting where each primitive begins and ends based on the membership of data points to a cluster.

The second step in our transcript generation is to assign labels to the primitives. Each movement can be described as a series of primitives. We label each primitive with a unique symbol. The transcript is then transformed to a sequence of symbols over a certain alphabet, which is unique for each sensor node.

Fig. 1 shows the transcript for a synthetic one-dimensional signal. The primitives are generated with GMM approach, labeled and colored. For example, primitive 'G' corresponds to a portion of the signal with a positive slope and 'W' represents a portion with positive value of the second derivative. Note that each primitive maintains its temporal characteristics. Since duration of both 'G' and 'M' is short in the original signal, the same is true in the transcript. This example clearly verifies that primitives can capture signal segments that exhibit consistent patterns.

***Definition** 1:* Given an observation O_{ij} of action A_j made by sensor node s_i, a *transcript* T_{ij} is generated by our techniques and is defined as a finite sequence of symbols from an alphabet Σ_i.

Each sensor node builds its transcripts independent of the patterns observed by other sensor nodes. Therefore, each node requires a separate alphabet Σ_i ($i \in \{1, \ldots, n\}$). Thus, symbols used for different nodes are mutually exclusive.

B. Template Generation

Making a decision on the current movement occurring in the system requires a predefined classifier which operates at the base station. Such a classifier receives data from several sensor nodes and makes a decision by combining the data using a fusion scheme. Similarly, when developing a distributed algorithm for action recognition, each node receives data from another node, performs local processing and forwards its results to the next node. As a result, a fusion policy is still employed by individual nodes to provide more informative observations to other nodes in the network. Traditional methods of constructing the data fusion function over a feature space take input from different sources. In our framework, however, each sensor node generates a 1-dimensional feature space in the form of transcripts. To enable the use of traditional classifiers, e.g. k-NN (k-Nearest-Neighbor) [19], a fusion technique is required to represent each trial of a movement by integrating spatially distributed transcripts. For this reason, we make a *template* for each trial, by combining its transcripts from all sensor nodes. Our template generation is a simple algorithm which produces a new transcript, i.e. template, by concatenating all transcripts of the same trial. In the following, we present a more formal definition of a template.

***Definition** 2:* The concatenation of n given strings S_1, S_2, $\ldots S_n$ yields another string S where all symbols of S_i follow by all symbols of S_{i+1}.

$$S = Concat(S_1, S_2, \ldots, S_n) \tag{5}$$

***Definition** 3:* Given a set of n transcripts $T_{1j}, T_{2j}, \ldots, T_{nj}$ associated with a certain trial of movement A_j and generated by n sensor nodes, the trial is represented by the template TPL_j=Concat(T_{1j}, T_{2j}, …, T_{nj}).

Each transcript T_{ij} is associated with a length $\ell(T_{ij})$ which is equal to the total number of symbols that form the transcript. Therefore, the length function is additive with respect to the string concatenation.

V. Action Recognition

Action recognition aims at classifying human movements into predefined actions. Movements are mainly postural motions such as 'Sit to Stand', 'stand to sit', 'kneel' and 'sit to lie' which can be specified by the start and the end of the

signal real. Taking into consideration the compact representation of physical movements using motion transcripts, a new movement can be classified in two ways. In the first method a central classifier is designed at the base station where the movement is labeled according to an existing training model. The second approach, however, uses in-network processing to make a final decision on the current movement by combining data from most informative nodes and converging to a final decision. In this section we elaborate on both techniques.

A. Centralized Architecture

For an observation associated with action A_j, each sensor node s_i generates a transcript T_{ij} over an alphabet Σ_i. In a centralized architecture, all sensor nodes transmit their local transcripts to a base station. For each observation of action A_j, a template TPL_j is then obtained by the base station. On observing an unknown action, a classification algorithm is used by the central node to classify that action as one of the movement based on which the classifier is already trained.

A number of classification algorithms have been used in the field of pattern recognition and machine learning. The (k-NN) [25] is a simple and scalable algorithm which assigns an unknown sample to its closest class according to a distance measure. Distance measure can quantify the level of similarity between an unknown movement and each sample in the training set. Euclidean distance is widely used as the similarity measure when the training set is constructed based on numerical values. In our system, however, each movement is represented by a sequence of characters. Therefore, a similarity metric is required to find the difference between two strings. The *edit distance* [26] is a well-known metric for measuring the amount of difference between two character sequences. The edit distance between two strings is given by the minimum number of operations needed to transform one string into the other, where an operation is defined as an insertion, deletion, or substitution of a single character.

Let T_{ij} be a transcript generated by a node s_i for an unknown action A_j. For each class C_{ij}, let T_{iq} be the closest transcript to T_{ij}. A 1-NN classifier assigns A_j to the class $A_{\hat{q}}$ such that:

$$\hat{q} = \arg\min_{q} \quad \delta(T_{ij}, T_{iq}) \tag{6}$$

where $\delta(T_{ij}, T_{iq})$ represents the value of the edit distance between transcripts T_{ij} and T_{iq}.

B. Distributed Paradigm

In the centralized architecture described earlier, when an unknown action occurs, all sensor nodes must transmit their local transcripts to the central node for the purpose of global classification. In a distributed scenario, however, each node makes a local decision on the target movement and may decide to propagate its local results to the next node in the network. The amount of data transmitted over the network can be reduced to only a subset of the nodes that contribute to the classification of the movement. In this section, we develop a distributed algorithm for action recognition which needs a smaller number of the nodes to make a decision while maintains classification accuracy comparable to the centralized architecture. We explore an important property of classifying movements using transcript representation prior to describing our distributed algorithm.

1) Additive Property: This property implies that the summation of edit distances computed locally is equal to the edit distance of the overall corresponding template. We note that nodes s_i and s_k construct their transcripts using separate alphabets Σ_i and Σ_k. Edit distance increases as a result of insertion of a character, deletion of a character, or substitution of an existing character with another. It can be shown that the edit distance is additive under each one of the above operations. Furthermore, edit distance calculation proceeds linearly and increases the sum by only 1 at a time (based on the operation performed), which means that any combination of the operations described above is also additive.

A direct consequence of the additive property of the edit distance is that a global decision can be made by calculating edit distances locally and adding them in the network to find the most similar movement. Recall from equation (6), assume a 1-NN classifier is employed to assign an unknown movement to one of pre-specified actions. Each sensor node makes its own training model by generating local transcripts. When an unknown action A_q occurs, node s_i generates a transcript T_{iq} representing a new action which we need to classify. In contrast with the centralized model, this time, each node s_i measures distance between the new movement T_{iq} and nearest trial within each local class. For each class C_{ij} within the model, assume T_{ij} represents the nearest trial to T_{iq}. The classifier then operates according to (7).

$$\hat{j} = \arg\min_{j} \quad \sum_{i=1}^{n} \delta(T_{iq}, T_{ij}) \tag{7}$$

It then assigns the new movement A_q to $C_{i\hat{j}}$. Target action is identified by adding edit distances up from all sensor nodes and finding the movement for which the summation has smallest value, pointing to the nearest class to the test trial.

The idea behind our distributed algorithm is similar to the basic principle behind classification. If the classification works correctly the value of only one movement's classifier will be below the threshold. This means that once the summation of distance values from a subset of nodes exceeds the threshold the corresponding classifier is bad and no further computation for it is needed. To capitalize on this property we create the ordering where the largest distance values are added first. They are more likely to make the summation exceed the threshold and invalidate bad classifiers early.

2) Algorithm: The algorithm assumes that each node processes data locally, generates transcripts and measures distance between an unknown trial and every class of movements. Each node assesses reliability of its own decision. Communication is initiated by the node that has the most reliable information for classification. The computation is executed by a series of the nodes until the solution converges. Each sensor node maintains a data structure, including its local computation as well as statistics received from other nodes. In particular,

each node s_i maintains a timer variable τ_i which represents the time left for the node to initiate communication. It also keeps track of recognition convergence by a variable *Target Movement Vector* (TMV) which initially contains all actions as possible target movements. As the algorithm proceeds, each node may decide to discard some movements from the *TMV*. Furthermore, each node s_i maintains a *Distance Vector* (DV) to evaluate confidence level of classification. This vector stores the distance between the new action and all classes within that node, and is gradually updated as a node receives corresponding distances from other nodes. The algorithm takes several steps as follows.

Step 1 (Initialization): Each sensor node s_i classifies an unknown movement A_q as $A_{\hat{j}}$ and forms its distance vector DV_i. It further sets a timer τ_i to have an inverse relationship with the average of distances between T_{iq} and all classes C_{ij} , excluding the target class $C_{i\hat{j}}$. Once τ_i expires the node starts transmitting its local statistics. These operations are formulated in (8) through (11).

$$DV_i = \{\delta(T_{iq}, T_{i1}), ..., \delta(T_{iq}, T_{im})\} \tag{8}$$

$$\hat{j} = \arg\min_j \quad \delta(T_{iq}, T_{ij}) \tag{9}$$

$$\Delta_i = \frac{1}{m-1} \sum_{j \neq \hat{j}} \delta(T_{iq}, T_{ij}) \tag{10}$$

$$\tau_i \propto \frac{1}{\Delta_i} \tag{11}$$

Our choice of Δ_i is inspired by confidence estimation of classification in machine learning and pattern recognition. The confidence measure is usually defined based on the minimum distance for which the class prediction changes [27]. In a 1-NN classifier it is equal to the distance to the second closest class. However, our pruning-based distributed classifier aims to reduce the number of nodes contributing in classification. Therefore, the distance measure Δ_i must be chosen to prune larger number of actions per node. The intuition is that large Δ_i corresponds to a set of large distances between T_{iq} and existing classes. A large distance between T_{iq} and a class T_{ij} suggests that it is less likely that T_{ij} is the target class.

Step 2 (Transmission): When the value of the timer τ_i becomes zero, the node s_i starts broadcasting its local statistics including DV_i and TMV_i. This node will never need to transmit again for detecting current action. Therefore, it can turn its radio off saving power until a new action occurs.

Step 3 (Update): On receiving data, each node s_k first terminates its timer to avoid the scheduled transmission. It then updates its local distance vector DV_k by adding corresponding values from TMV_i provided by the sender node s_i. The receiver further updates the Target Movement Vector TMV_k by rejecting the movements that are far enough from the target class. To do so, the node s_k discards those movements A_j that have an accumulate distance greater than or equal to a threshold ϵ_j. The receiver also checks conditions for termination. Specifically, it checks the convergence vector TMV_k which contains possible movements left. If only one movement is left in the vector, the node declares a convergence and reports that movement as the target action. It then broadcasts a message to all the remaining nodes to stop their scheduled transmission. However, if more than one action is left in TMV_k, the node would schedule a transmission by resetting its timer as discussed previously. These operations are summarized in Algorithm 1. The algorithm proceeds through Steps 2 and 3 until it uniquely identifies an action as target movement.

Algorithm 1 Updating TMV_k at node s_k

```
if δ(T_kq, T_kj) ≥ ε_j then
    remove action A_j from TMV_k
end if
if |TMV_k| = 1 then
    declare A_ĵ as target movement
else
    set timer τ_k as in equation (11)
end if
```

Algorithm 2 Updating rejection criteria for faster convergence

```
if δ(T_kq, T_kj) ≥ ((n_v + b)/n) ε_j then
    remove action A_j from TMV_k
end if
```

3) Choice of Epsilon: Our distribute algorithm considers a complete list of movements when it starts. As it goes after different nodes, the system tends to disqualify those actions that have a large distance to the test trial. The pruning decision described in Algorithm 1 is made according to the value of ϵ_j which is defined for every movement A_j as in (12).

$$\epsilon_j = \sum_{i=1}^{n} \left[\frac{1}{M_j} \sum_{s \neq t} \delta(T_{ij}^s, T_{ij}^t) \right] \tag{12}$$

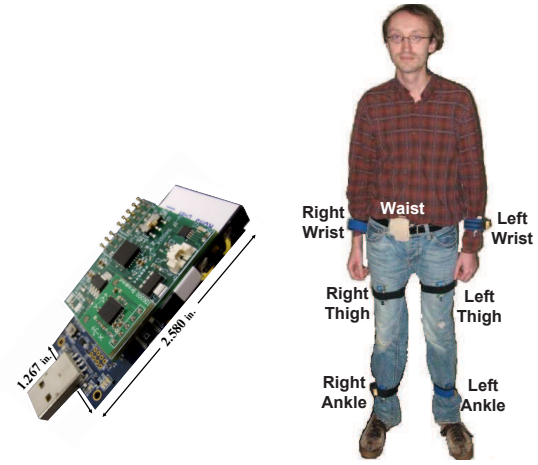
where M_j represents the number of samples in class C_{ij} and for every training trial 't', trial 's' refers to the trial that has minimum distance with 't'. For each movement, we calculate the distance between each trial and its closest sample. By taking an average over all such pairs of the trials and adding value from all the nodes, we compute the maximum value we expect to get when an unknown movement is classified as this class. During system training, ϵ_j is calculated for every training class. During classification, once the summation of distance values from a subset of nodes exceeds this threshold the corresponding movement is disqualified and no further computation for it is needed. The choice of epsilon would determine the performance of the classifier. However, our proposed technique can be applied independent of choice of epsilon.

4) Augmenting Classification: The criterion $\delta(T_{kq}, T_{kj}) \geq \epsilon_j$ in Algorithm 1 for rejecting movements from further processing is determined in a cautious way. This method may require the algorithm to go after more sensor nodes that it actually needs to make a classification decision. However, the criterion can be modified for a faster convergence. Depending on the classification accuracy, designer may decide to use different criteria. Algorithm 2 shows one of the approaches which updates the value of ϵ_j dynamically based on the number of nodes already visited.

where n_v represents the number of nodes already considered for classification, and n is the total number of sensor nodes,

TABLE II
EXPERIMENTAL MOVEMENTS

No.	Movement
1	Stand to sit
2	Sit to lie
3	Bend and grasp
4	Kneel
5	Turn counter clockwise
6	Look back clockwise
7	Move forward (1 step)
8	Move to the side (1 step)
9	Reach up to cabinet
10	Jump



(a) A mote with custom-designed inertial sensor board and battery (b) Subject wearing seven sensor nodes

Fig. 2. Sensor node and subject

and b is a tunable parameter which can be adjusted by the designer to obtained desired classification accuracy. $(n_v+b)/n$ represents the fraction of ϵ_j that is required for classification termination.

VI. SYSTEM PROTOTYPE

In this section, we present procedures for developing our action recognition framework. Moreover, we demonstrate the effectiveness of our system using a prototype developed in our research laboratory.

A. Data Collection

We developed our trial product for identifying 10 transitional movements listed in Table II. The experiments were carried out on five subjects, three males and two females, all between the ages of 25 to 55 and in good health condition. Seven sensor nodes were placed on the subjects as shown in Fig. 2(b). Subjects were asked to repeatedly perform each specific motion 10 times.

1) Node Placement: Current node placement methodologies are either intuitive or analytical. Intuitive methods [17] position sensor nodes according to the set of movements of interest and physical model representing human body. Analytical approaches [4, 5] strive to find optimal positioning of the sensor nodes by searching minimum number of nodes, among an exhaustively distributed sensor nodes. In this paper, we follow an intuitive approach, and therefore, distribute seven sensor nodes on different body segments as shown in Fig. 2(b).

2) Sampling Frequency: The motes were programmed to sample sensors (accelerometer and gyroscope) at 50 Hz. The sampling frequency was chosen to satisfy the Nyquist criterion. For estimation of the Nyquist frequency, the power spectrum of the sampled signals was examined. From the power spectrum graphs, the highest frequency of the signal was 8.5 Hz which means that a sampling frequency of 17 Hz would suffice to meet the Nyquist frequency.

The sampled data were sent wirelessly to a base station using a TDMA protocol. The base station was connected to a laptop via USB to deliver received data to our data collector tool.

For data collection, each mote was programmed to transmit data in packets of size 41 bytes, containing two consecutive samples of accelerometer and gyroscope readings. Given the 250 Kbps bandwidth available on TelosB motes [15], the base station can theoretically support up to 31 sensor nodes at 50 Hz.

The communication system for data collection did not use a retransmission mechanism for lost packets. However, a MATLAB tool was developed to estimate missing samples using linear interpolation. Throughout collecting data of different nodes, subjects, movements and trials, the packet lost never exceeded 20%.

B. Data Processing

For each movement, 50% of the trials were used to generate the training model, and the rest were used to verify the action recognition technique. For each trial, the raw sensor readings were passed through a five-point moving average filter to reduce high frequency noise. The filtered data were reported at the middle of the window. The five-point moving average filter is a low pass filter with a cutoff frequency of 2.4 Hz. The cutoff frequency was obtained by conducting a discrete Fourier transform analysis in MATLAB. Given the 50 Hz sampling frequency, the Periodogram analysis was used to estimate the Power Spectral Density (PSD) of the signal. The cutoff frequency was determined as the frequency corresponding to 3 db below the first peak in the plot of the signal PSD.

The choice of the window size for the moving average filter relies on two objectives 1) the cutoff frequency needs to be low enough to effectively bypass unnecessary motions such as tremors that occur at higher frequencies than usually movements. 2) the cutoff frequency must be high enough to maintain significant data. With these objectives, different filters with varying window sizes ranging from 3 to 13 were examined. The cutoff frequencies ranged from 4.3 for three-point filter to 1.1 for thirteen-point filter. The three-point filter was pruned out because it had a cutoff frequency of 4.3 Hz which is within the range of undesirable motions. Tremors in patients with Parkinson's disease occur at frequencies 4-5.3 Hz [28]. Among the remaining filters, the filter that generated highest quality clusters during transcript generation was chosen. Practically, a very large window would cause the transcripts to miss key details and too small windows would produce irrelevant and misleading clusters. The Silhouette measure [29] reported the highest value for the five-point moving average filter.

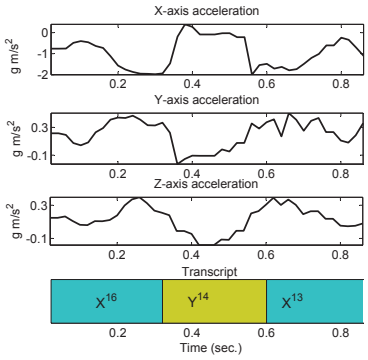


Fig. 3. Transcript for a trial of 'Jump' generated by the right-wrist node

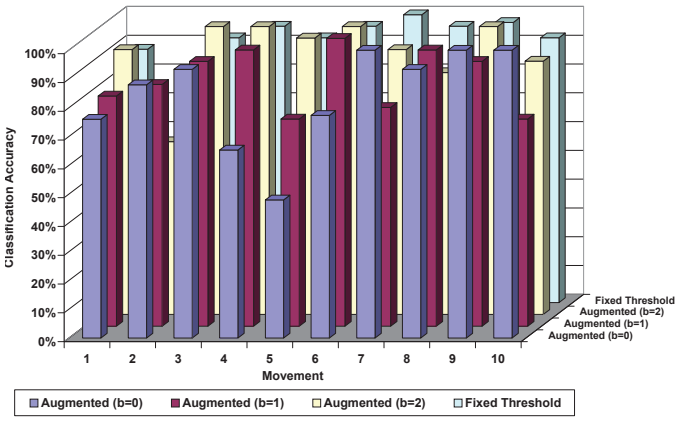


Fig. 4. Classification accuracy per movement for different experimental categories.

To capture parts of the signal that correspond to a complete action, we used the video data which was recorded during the data collection. Using video, we found the start and the end of each trial and ignored non-activity parts in subsequent processing.

The next step in our signal processing was feature extraction. The five statistical features described in Section III-B were extracted from a moving window centered about each sample. These features were calculated for all training trials. The features were then used for GMM clustering which aimed to construct primitives of the movements. Motion transcripts were generated by individual nodes using separate alphabets. Fig. 3 illustrates transcript of the 'Jump' movement, generated by the node placed on the right-wrist. For visualization, only accelerometer readings are shown, but both accelerometer and gyroscope readings were used for clustering. Acceleration is measured with respect to the gravitational acceleration, g, as shown on X-axis. Each movement is divided into several segments, each representing a primitive. A string α^L denotes L instances of primitive α mapped to the same cluster. For example, X^{16} in Fig. 3 accounts for the same classification for the first 16 points.

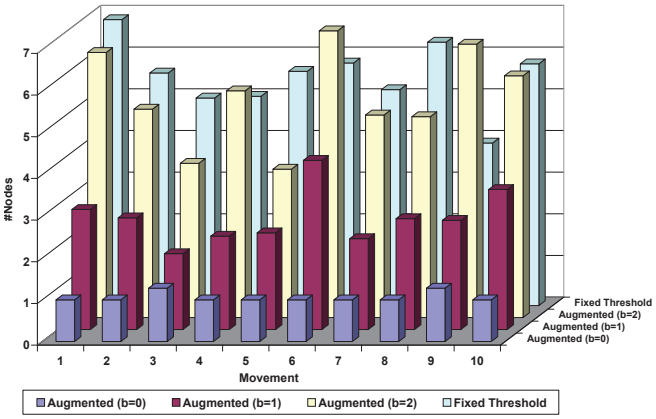


Fig. 5. Average number of active nodes for detecting each movement

C. Classification Accuracy

As mentioned earlier, we used 50% of the trials to validate the effectiveness of our distributed action recognition technique. For each test trial, our local processing proceeded to generate transcripts at each node. With the transcripts, we computed the distance between the test trial and each movement class. Based on the resulting distance vector, a sensor node might decide to transmit its local statistics as describes in Section V-B. The algorithm could eventually output an action as the target movement. We compared this output with the actual label obtained during the data collection to verify the classification decision. Using 250 test trials (5 subjects, 10 actions, 5 trials each), we achieved an overall classification accuracy 84.13% when only one sensor node was required on average to classify each action. This value was obtained when the augmented rejection threshold with $b = 0$ was used (see Algorithm 2). We also measured the classification accuracy and the average number of sensor nodes required for classification for the case of fixed threshold (see Algorithm 1). Fig. 4 shows classification accuracy for each class of movements. The results verify that adjusting the value of rejection threshold based on augmented approach provides the best results in terms of the average number of active nodes for classification.

In summary, we performed analysis for four categories according to the movement rejection criterion: 1) Fixed criterion, when the value of the rejection threshold was fixed based on training data. The classification accuracy was 91.33% and the average number of nodes required to converge was 5.5 nodes. In this case the same accuracy as the centralized algorithm was obtained. 2) Augmented approach with $b = 0$, where threshold was updated in real-time according to the number of nodes already visited. This reached an accuracy of 84.13% and 1 active node. 3) Augmented with $b = 1$; with this setup, we obtained 86% accuracy and 2.6 nodes in average. 4) Augmented with $b = 2$; the classification accuracy for this case was 91.2% and the average number of nodes was 5.3. These results are summarized in Table III.

Fig. 5 shows the average number of nodes for classification of each movement. The values are categorized based on the rejection criterion. On average, only one node was needed for

TABLE III
OVERALL CLASSIFICATION ACCURACY AND AVERAGE NUMBER OF NODES FOR DIFFERENT SETUPS

	Centralized	Fixed Threshold	Augmented (b=2)	Augmented (b=1)	Augmented (b=0)
Accuracy	91.33%	91.33%	91.20%	86.00%	84.13%
Average # of nodes	7.0	5.5	5.3	2.5	1.0

TABLE IV
VALUE OF EPSILON CALCULATED FOR EACH MOVEMENT CLASS

No.	Action	ϵ
1	Stand to sit	127
2	Sit to lie	90
3	Bend and grasp	115
4	Kneel	159
5	Turn	121
6	Look back	129
7	Move forward	99
8	Move to the side	94
9	Reach up	196
10	Jump	80

the case of augmented threshold adjustment with $b = 0$.

Table IV shows the value of ϵ for each action based on equation (12). We recall that for each particular movement, this value represents how well that movement is separated from the rest of classes on the training data. As an example, movement 9 (reach up to cabinet) has the largest value of ϵ. This observation can be interpreted as follows. Movement 9 can be uniquely identified by the node placed on the forearm (e.g. node 2) as this is the only node that experiences distinguishable patterns when person performs the action. During other actions either several body segments are expected to be involved or different motions are introduced by the forearm. As a consequence, sensor data obtained for this movement can provide different structural and relational information from those obtained for the other actions.

D. Robustness

1) Subject-independent Classification: Successful deployment of a classification model requires the results to be independent of the observations based on which the system has been developed. In this section, we demonstrate the robustness of this system to changes in target population. For this purpose, cross-subject classification accuracy is calculated where the data from a test subject is not used for training. This allows us to estimate the amount of misclassification for a new subject without prior training data from that subject.

Since five subjects participated in data collection, five different tests were conducted, each measuring classification accuracy for one subject not being used for training. The test subject was not used for building the training model including calculation of the GMM parameters and forming training data for k-NN classification. Classification accuracy ranged between 75% for subject 5 (i.e. S_2) to 88% for subject 2 (i.e. S_2). Subjects S_1, S_3 and S_4 obtained 80%, 83% and 83% accuracy respectively. As mentioned before, the S_5 had the lowest accuracy among all other subjects. Major source of misclassifications for this subject seemed to be 'Turning' action (40% accuracy for this movement when S_5 used for test, and S_1 to S_4 were used for training). This can be explained by the fact that S_5 was the oldest subject with an age of 55, and therefore, her movements have been less similar to other subjects whose age ranged between 25 and 35.

2) Calibration: An important aspect of sensor-based motion analysis is robustness with respect to sensor displacement and misorientation. The accelerometers used in our study, LIS3LV02DQ, measure acceleration relative to the gravity. Therefore, in static mode, the accelerometer detects the gravity. This information can be used to correct misplacement and misorientation of the nodes in order to maintain consistent mote positioning for throughout the experiments. While this approach helps in correcting initial node placements, more complex calibration models are needed to compensate for dynamic changes due to node misplacements during each movement trial. One approach is to find signals/features that are insensitive to node misplacement, and use them for classification [30].

E. Algorithm Complexity

The five statistical features described in Section III-B are calculated from each sampled data. As a result, the feature extraction linearly grows with the number of samples within each action and the number of features, turning the feature extraction into a linear algorithm in the number of features and length of actions.

Once GMM clustering is developed, it is used to generate transcripts for each action. Transcript generation for a test trial consists of finding proper label for each data point based on maximum posterior probability criterion described in Section IV-A. This process is linear in the number of training points.

The k-NN classification is performed based on inter-transcript edit distance calculation. The edit distance function is usually implemented using dynamic programming and is quadratic in the length of transcript.

VII. DISCUSSION AND FUTURE WORK

Currently, our sensing platform is used for data collection, and the signal processing modules are developed offline in MATLAB to facilitate design process. However, our preliminary results on algorithm development for real-time execution demonstrate the applicability of the processing tasks for implementation and execution on the mote [31].

The system presented in this paper does not accommodate an automatic segmentation technique. The segmentation process is currently done manually using video data of the experiments. However, we are working on developing automatic segmentation and annotation techniques that meet computation constraints of the system [17]. Moreover, the only use of the video data is segmentation. Therefore, once an automatic segmentation is employed, our system can work independent of the video-based training.

Our work in constructing movement transcripts is ongoing. We would like to explore the effectiveness of using transcripts

to extract numeric parameters from actions. Examples of this include grading swings in sports and determining pathological qualities of gait. Furthermore, we are planning to determine the performance bounds on our distributed algorithms.

VIII. Conclusion

We presented a dynamic distributed model of movement classification in body sensor networks. The system relies on motion transcripts which are built using mobile wearable inertial sensors. Motion transcripts were chosen because their use significantly decreases the cost of communication in the network. Additionally, we proposed a distributed approach, where individual nodes transmit their local results using a timer based on the likelihood of local results being eliminated by the pruning. When all but one action is eliminated, the algorithm stops. The dynamic nature of the algorithm decreases the number of active nodes required to make a classification decision, thus contributing to further reducing power consumption in the network. Our results show the effectiveness of this approach, both for recognition and reduction of communication.

References

[1] L. Hyman, *Phonology: Theory and Analysis*. Heinle & Heinle Publishers, 1975.

[2] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," *Computer Networks*, vol. 38, no. 4, pp. 393–422, 2002.

[3] R. Musaloiu and A. Terzis, "Minimising the effect of wifi interference in 802.15.4 wireless sensor networks," *Int. J. Sen. Netw.*, vol. 3, no. 1, pp. 43–54, 2007.

[4] P. Zappi, C. Lombriser, T. Stiefmeier, E. Farella, D. Roggen, L. Benini, and G. Troster, "Activity recognition from on-body sensors: accuracy-power trade-off by dynamic sensor selection," *Lecture Notes in Computer Science*, vol. 4913, p. 17, 2008.

[5] H. Ghasemzadeh, E. Guenterberg, and R. Jafari, "Energy-Efficient Information-Driven Coverage for Physical Movement Monitoring in Body Sensor Networks," *IEEE Journal on Selected Areas in Communications*, vol. 27, pp. 58–69, 2009.

[6] S. Kang, J. Lee, H. Jang, H. Lee, Y. Lee, S. Park, T. Park, and J. Song, "Seemon: scalable and energy-efficient context monitoring framework for sensor-rich mobile environments," in *MobiSys '08: Proceeding of the 6th international conference on Mobile systems, applications, and services*. New York, NY, USA: ACM, 2008, pp. 267–280.

[7] P. Veltink, L. Martens, and R. Van Lummel, "Detection of static and dynamic activities using uniaxial accelerometers," *IEEE Transactions on Rehabilitation Engineering*, vol. 4, no. 4, p. 375, 1996.

[8] G. Guerra-Filho, C. Fermller, and Y. Aloimonos, "Discovering a language for human activity," in *FS'05: Proc. of the AAAI 2005 Fall Symposium on Anticipatory Cognitive Embodied Systems*, 2005, pp. 70–77.

[9] G. Guimarães and L. Pereira, "Inferring Definite-Clause Grammars to Express Multivariate Time Series," in *Proceedings of the 18th international conference on Innovations in Applied Artificial Intelligence*. Springer, 2005, pp. 332–341.

[10] Z. Husz, A. Wallace, and P. Green, "Human activity recognition with action primitives," *Advanced Video and Signal Based Surveillance, 2007. AVSS 2007. IEEE Conference on*, pp. 330–335, Sept. 2007.

[11] O. C. Jenkins and M. J. Mataric, "Automated derivation of behavior vocabularies for autonomous humanoid motion," in *AAMAS '03: Proceedings of the second international joint conference on Autonomous agents and multiagent systems*. New York, NY, USA: ACM, 2003, pp. 225–232.

[12] N. Niwase, J. Yamagishi, and T. Kobayashi, "Human walking motion synthesis with desired pace and stride length based on hsmm," *IEICE - Trans. Inf. Syst.*, vol. E88-D, no. 11, pp. 2492–2499, 2005.

[13] P. Fihl, M. Holte, T. Moeslund, and L. Reng, "Action recognition using motion primitives and probabilistic edit distance," *Lecture Notes in Computer Science*, vol. 4069, p. 375, 2006.

[14] T. Stiefmeier, D. Roggen, and G. Tröster, "Gestures are strings: efficient online gesture spotting and classification using string matching," in *BodyNets '07: Proceedings of the ICST 2nd international conference on Body area networks*, ICST, Brussels, Belgium, Belgium, 2007, pp. 1–8.

[15] J. Polastre, R. Szewczyk, and D. Culler, "Telos: enabling ultra-low power wireless research," *Information Processing in Sensor Networks, 2005. IPSN 2005. Fourth International Symposium on*, pp. 364–369, April 2005.

[16] N. Stergiou, *Innovative Analyses of Human Movement: Analytical Tools for Human Movement Research*. Human Kinetics, 2003.

[17] E. Guenterberg, H. Ghasemzadeh, V. Loseu, and R. Jafari, "Distributed continuous action recognition using a hidden markov model in body sensor networks," in *DCOSS '09: Proceedings of the 5th IEEE International Conference on Distributed Computing in Sensor Systems*. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 145–158.

[18] H. Ghasemzadeh, E. Guenterberg, S. Ostadabbas, and R. Jafari, "A motion sequence fusion technique based on pca for activity analysis in body sensor networks," in *EMBC '09: Proceedings of the 31st Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, 2009.

[19] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*. Wiley-Interscience Publication, 2000.

[20] J. B. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability*, L. M. L. Cam and J. Neyman, Eds., vol. 1. University of California Press, 1967, pp. 281–297.

[21] S. C. Johnson, "Hierarchical clustering schemes," *Psychometrika*, vol. 32, no. 3, pp. 241–254, September 1967.

[22] M. Figueiredo and A. Jain, "Unsupervised learning of finite mixture models," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 24, no. 3, pp. 381–396, Mar 2002.

[23] C. Fraley and A. Raftery, "How Many Clusters? Which Clustering Method? Answers Via Model-Based Cluster Analysis," *The Computer Journal*, vol. 41, no. 8, pp. 578–588, 1998.

[24] G. J. McLachlan and T. Krishnan, *The EM Algorithm and Extensions, 2nd Edition*. John Wiley, 2008.

[25] T. Cover and P. Hart, "Nearest neighbor pattern classification," *Information Theory, IEEE Transactions on*, vol. 13, no. 1, pp. 21–27, 1967.

[26] V. Levenshtein, "Binary Codes Capable of Correcting Deletions, Insertions and Reversals," in *Soviet Physics Doklady*, vol. 10, 1966, p. 707.

[27] J. Afalg, H.-P. Kriegel, A. Pryakhin, and M. Schubert, "Multi-represented classification based on confidence estimation," in *Advances in Knowledge Discovery and Data Mining*, ser. Lecture Notes in Computer Science, vol. 4426. Springer Berlin / Heidelberg, 2007, pp. 23–34.

[28] L. J. Findley, M. A. Gresty, and G. M. Halmagyi, "Tremor, the cogwheel phenomenon and clonus in Parkinson's disease." *J Neurol Neurosurg Psychiatry*, vol. 44, no. 6, pp. 534–546, 1981.

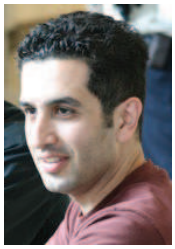
[29] P. Rousseeuw, "Silhouettes: a graphical aid to the interpretation and validation of cluster analysis," *J. Comput. Appl. Math.*, vol. 20, no. 1, pp. 53–65, 1987.

[30] H. Harms, O. Amft, and G. Tröster, "Modeling and simulation of sensor orientation errors in garments," in *Bodynets 2009: Proceedings of the 4th International Conference on Body Area Networks*. ACM press, 2009.

[31] H. Ghasemzadeh, N. Jain, M. Sgroi, and R. Jafari, "Communication Minimization for In-Network Processing in Body Sensor Networks: A Buffer Assignment Technique," in *IEEE/ACM Design Automation and Test in Europe (DATE)*, 2009.

Hassan Ghasemzadeh received the B.Sc. degree in Computer Engineering from Sharif University of Technology, Tehran, Iran, and the M.Sc. degree in Computer Engineering from University of Tehran, Tehran, Iran in 1998 and 2001 respectively. He joined the University of Texas at Dallas in 2007 where he is pursuing his Ph.D. in Computer Engineering. His research interests lie in different aspects of embedded systems. He is currently working on collaborative signal processing, reconfigurable computing and algorithm design for medical embedded systems. He is a student member of the IEEE.

Vitali Loseu received the B.S. degree in Computer Science, and the M.S. degree in Computer Science from the University of Texas at Dallas in 2007 and 2008 respectively. He then proceeded to pursue his Ph.D. in Computer Engineering in the Embedded Systems and Signal Processing Lab (ESSP). His research interests lie in system optimization for reconfigurable computing. He is a student member of the IEEE.



Dr. Roozbeh Jafari received his B.Sc. in Electrical Engineering from Sharif University of Technology in 2000. He received an M.S. in Electrical Engineering from SUNY at Buffalo, and an M.S. and a Ph.D. in Computer Science from UCLA in 2002, 2004 and 2006 respectively. He spent 2006-2007 in EECS department at UC Berkeley as a post-doctoral researcher. Dr. Jafari is currently an assistant professor in Electrical Engineering at the University of Texas at Dallas. His research is primarily in the area of networked embedded system design and reconfigurable computing with emphasis on medical/biological applications, their signal processing and algorithm design. He is the director of ESSP Lab.