Physical Movement Monitoring using Body Sensor Networks: A Phonological Approach to Construct Spatial Decision Trees

Hassan Ghasemzadeh, *Member, IEEE,* Roozbeh Jafari, *Member, IEEE*

***Abstract*—Monitoring human activities using wearable sensor nodes has the potential to enable many useful applications for everyday situations. Limited computation, battery lifetime and communication bandwidth make efficient use of these platforms crucial. In this paper, we introduce a novel classification model that identifies physical movements from body-worn inertial sensors while taking collaborative nature and limited resources of the system into consideration. Our action recognition model uses a decision tree structure to minimize the number of nodes involved in classification of each action. The decision tree is constructed based on the quality of action recognition in individual nodes. A clustering technique is employed to group similar actions and measure quality of per-node identifications. We pose an optimization problem for finding a minimal set of sensor nodes contributing to the action recognition. We then prove that this problem is NP-hard and provide fast greedy algorithms to approximate the solution. Finally, we demonstrate the effectiveness of our distributed algorithm on data collected from five healthy subjects. In particular, our system achieves a 72.4% reduction in the number of active nodes while maintaining 93.3% classification accuracy.**

***Index Terms*—Body Sensor Networks, Action Recognition, Collaborative Signal Processing, Phonology, Primitive, Decision Tree Model.**

I. Introduction

Wireless sensor networks have caught tremendous attention recently due to their potential for a large number of application domains. Applications range from monitoring systems such as environmental and medical monitoring to detection and supervision systems for military surveillance. A special class of these systems, called body sensor networks (BSNs), uses a network of light-weight embedded sensory devices to acquire and process physiological data about the subject wearing the system. BSNs can foster medical services by providing real-time and remote healthcare monitoring. They can be effective for rehabilitation, sports medicine, geriatric care, gait analysis, and detection of neuro-degenerative disorders such as Alzheimer's, Parkinson's, and Huntington's diseases [1].

Action recognition aims to detect transitional movements such as 'sit to stand', 'sit to lie', and 'walking'. Action recognition is usually required for development of many other applications. In gait analysis, certain information about the quality of gait is extracted when the person is walking. Thus, the current action needs to be reported prior to execution of other processing tasks. In monitoring Parkinson's Disease (PD) patients, several symptoms such as resting tremor, muscular rigidity, bradykinesia or delayed initiation of movements, and postural instability need to be detected. However, these symptoms are not equally pronounced during all of the human actions. For example, bradykinesia cannot be identified when a subject is perfectly still. In order to properly recognize PD symptoms, action recognition needs to be performed first.

Limited processing power and finite battery energy are two major obstacles in realizing real-time applications of the BSNs. The limited computing capability warrants the need for development of computationally inexpensive algorithms that run on the light-weight sensor nodes, and reduce complexity of the large amount of data collected by the sensor nodes. Battery lifetime, however, can be maximized by optimizing function of individual components such as processing unit and communication system. Studies have shown that communication consumes significantly more energy than data processing. Hence, significant power saving can be obtained by enhancing the communication system.

This study focuses on developing a computationally simple and distributed algorithm for action recognition. The task involves introducing a novel representation of human actions in terms of their basic building blocks, called primitives. With this approach, each action is represented as a set of symbols associated with the primitives. A distributed algorithm is then developed which detects human actions according to a decision tree model. The decision tree is derived directly from interpretation of action primitives and the level of contribution of each sensor node for action identification. The distributed algorithm produces a global classification decision based on a subset of results generated by individual sensor nodes. The compact representation of actions along with distributed nature of the algorithm enables our system to lower the amount of information stored at individual nodes, and to minimize the amount of data passed in the network. Therefore, the amount of energy required by individual nodes for data transmission is reduced. Furthermore, with the dynamic selection of the nodes needed for classification the overall number of active nodes is reduced. This would lead to reducing the overall power consumption of the system and can potentially increase system lifetimes.

Our contributions in the paper can be summarized as follows. 1) An efficient representation of individual nodes' knowledge about each action is presented using the concept

H. Ghasemzadeh is with the West Wireless Health Institute, 10350 North Torrey Pines Road, La Jolla, CA 92037, USA; e-mail: hghasemzadeh@gmwhi.org; R. Jafari is with the Department of Electrical Engineering, University of Texas at Dallas, Richardson, Texas 75080-3021, USA; e-mail: rjafari@utdallas.edu.

of action primitives. This provides a compact and flexible representation for atomic movements which can enhance communication by reducing the size of messages required to transmit across the network. We provide phonological rules for constructing primitives associated with each action. 2) we investigate the detection of human actions based on a decision tree model. In a decision tree for action recognition, each internal decision vertex is associated with a sensor node, and terminal leaves correspond to actions of interest. Depending on the observation made by a sensor node, an internal vertex can branch to one node or another. When a new action occurs, the system will classify that action by visiting a sequence of nodes on a path from the root to a leaf. The tree is optimal if a minimum number of nodes classifies all the actions. Such model reduces the number of active nodes by minimizing height of the tree.

II. Related Work

A number of researchers have integrated on-body sensors in a wireless network for the purpose of activity recognition and lifestyle monitoring. In [2], authors report the results of a study on activity recognition using different types of sensory devices, including built-in wired sensors, RFID tags, and wireless inertial sensors. The analysis performed on 104 hours of data collected from more than 900 sensor inputs shows that motion sensors outperform the other sensors on many of the movements studied. A wireless body sensor system for monitoring human activities and location in indoor environments is introduced in [3] where each sensor node is equipped with accelerometer, gyroscope and magnetometer. Authors in [4] use a network of five accelerometers to classify a sequence of daily activities. They report a classification accuracy of 84% for detecting twenty actions. The system in [5] uses seven different sensors embedded in a single node to classify twelve movements. The accuracy obtained by this system is 90%. All the aforementioned systems use a centralized approach for action recognition, where a central classifier combines data of all sensor nodes and make a global decision. In contrast, our study aims to build a distributed action recognition model that uses only a subset of nodes for classification. In a previous work [6], we introduced a heuristic approach for distributed action recognition based on the concept of motion transcripts. There are two major differences between the current study and our previous work: 1) In this paper, we introduce the concept of spatial primitives that represent each action as a set of primitives spatially distributed across the network. Our previous work uses the notion of temporal primitives where each action is represented as a sequence of primitives by each individual node. 2) While the distributed classification algorithm in [6] is heuristic and cannot essentially provide the optimal ordering of the nodes for classification, our decision tree based classifier in this paper has strong theoretical foundation.

The idea of primitive-based activity recognition originated from research work in computer vision, where both static and dynamic vision-based approaches have been developed. For static methods, individual time frames of a video sequence are used as the basic components for analysis. Recognition involves the combination of discrete information extracted from individual frames. In dynamic methods, a fixed interval of a video stream is the major unit of analysis. The Hidden Markov Model (HMM) [7], which takes into account the correlation between adjacent time instances by formulating a Markov process [8], is often used for the dynamic representation of motion due to its ability to handle uncertainty with its stochastic framework. Examples include the work presented in [9], that introduces a statistical technique for synthesizing walking patterns. The motion is expressed by a sequence of primitives extracted using an HMM-based model. The limitation of HMM in efficiently handling several interdependent processes has led to the creation of grammar-based representations of human actions. In [10], the authors propose a platform for a visuo-motor language. They provide the basic kernel for the symbolic manipulation of visual and motor information in a sensory-motor system. Their phonological rules are modeled as a finite automaton. Authors in [11] present an algorithm that finds primitives for human gestures using a motion capture system measures the 3D position of body parts. The trajectory of motion of these parts is considered as a gesture, and primitives are constructed based on the density of the training data set.

The primary objective of our work is to develop a fast classification algorithm which minimizes the number of active nodes involved in distributed action recognition. We use the notion of decision tree for this purpose. In order to construct a tree, we map each movement onto a set of predefined primitives. Our spatially distributed primitives simplify the representation of movements and enable construction of a decision trees. Our model further simplifies communication as each node in the tree needs to convey information on its final result to the next node in the tree.

III. System Architecture and Signal Processing

Our action recognition platform is a BSN consisting of several sensor units in a wireless network. The sensor nodes aim to collectively detect transitional movements according to a training model. Each node, which is also called a mote, has a triaxial accelerometer, a biaxial gyroscope, a microcontroller, and a radio. Nodes sample sensor readings at a certain frequency and can transmit the data wirelessly to each other. The motes that are used in this study are TelosB, which are commercially available from XBow®. The core processing unit of a Telos mote is TI MSP430 with active power of 3 mW. The power consumption of the radio is 38 mW and 35 mw in receive and transmit modes respectively. Each mote is interfaced with custom-designed sensor board and is powered by a Li-Ion battery. For the purpose of data collection, we use an extra mote as a base station. The base station is connected to a laptop by USB and communicates with on-body nodes using a TDMA protocol. Furthermore, two Logitech webcams are used to record video of all experiments. The video is used only as a guide for manual segmentation of the actions. The sensor readings and video are collected and synchronized in MATLAB where we develop our distributed algorithms.

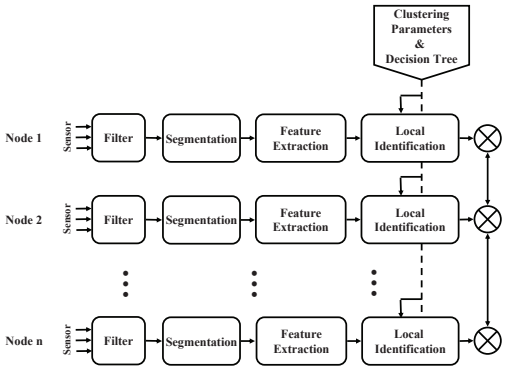

Fig. 1. Signal processing flow for distributed action recognition based on decision tree model.

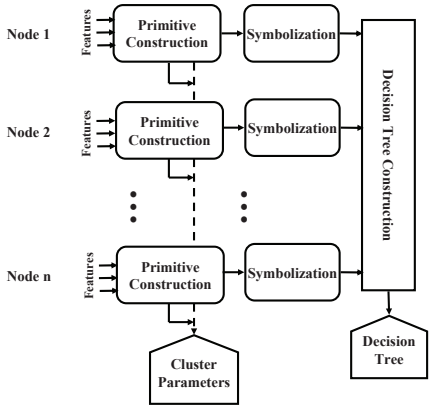

Fig. 2. Training model to construct action primitives and build a decision tree.

Currently, we use this sensing platform for data collection and perform all processing offline in MATLAB.

Our distributed algorithm uses the signal processing model shown in Fig. 1 to classify an unknown action by integrating information from sensors across the network. This processing model requires several parameters as well as a decision tree which are pre-calculated during training. The training is conducted as shown in Fig. 2. Data are collected from each sensor (x, y, z accelerometer, and x, y gyroscope) at 50 Hz and are filtered with a five-point moving average filter to remove high frequency noise. Next, signals are manually segmented to determine the temporal regions that represent actions. Other processing tasks are described briefly in the following.

Feature Extraction: Single-value features are extracted from the signal segment. Statistical features include mean, start-to-end amplitude, standard deviation, peak-to-peak amplitude, and RMS power. Moreover, 10 morphological features are obtained from 10 uniformly distributed points on the signal where each feature is the value of the signal at one of the points. Intuition behind using the above feature set is that 1) these time-domain features are computationally inexpensive that can be executed on our light-weight sensor nodes 2) they capture both statistical and structural properties of the signal 3) significance of statistical and morphological features for human movement analysis has been previously suggested by other researchers [12], and their effectiveness is established by our experimental results in this paper.

Per-Node Identification: Each node uses a phonetic expression of movements to map a given action onto the corresponding primitive. Each primitive is represented by a cluster, which groups similar actions together based on the features that are extracted from each signal segment. During identification, an unknown action is mapped onto a cluster based on the cluster configuration obtained during training.

Distributed Action Recognition: The global state of the system is recognized by combining local knowledge from different nodes using a decision tree model. The decision tree allows for incremental classification of actions and is specifically constructed to minimize the number of nodes involved in the classification. Each node can distinguish certain actions based on the cluster assignment of the actions. The node which is most informative among all existing nodes is used as the root of the tree. The remaining nodes are organized in the tree according to their capabilities in distinguishing the rest of the actions. This decision tree based model classifies an unknown action by extracting information from a subset of nodes in a certain order and typically determining the action before all nodes are considered. The decision tree, which is constructed during training, determines the node that initiates communication. The algorithm proceeds by transmitting local results of one node to the next node in the tree. On receiving data, the node combines the data with its local statistics and may decide to branch to another node in the tree. This process continues until all the actions are distinguished, detecting the target action.

Primitive Construction and Symbolization: A phonological approach is used to construct spatially distributed primitives of actions at each sensor node. Primitives are created using unsupervised classification techniques that map signal readings with similar patterns onto the same clusters. We use a k-means clustering algorithm to group similar actions based on the value of the statistical and morphological features extracted from each signal segments. On each individual sensor node, several actions might fall into the same cluster. Each cluster represents a primitive. When primitives are generated, a symbol is assigned to each primitive. By symbolization, each action is represented in terms of a set of meaningful phonemes that are labeled by alphabets.

IV. Preliminaries

Accurate detection of actions requires a global view of the whole system, but each individual node in a BSN has only local knowledge of the event taking place. The ability of a node to recognize actions varies based on the type of action. For example, consider the two actions ‘stand to sit’ and ‘bend’. A node mounted on the ‘arm’ might distinguish these two actions, but a sensor on the ‘ankle’ might not provide useful information. The main advantage of the clustering model in our system is that by grouping training trials at each node, nodes not contributing to a certain action will be identified. That is, we will be able to determine which nodes are useful for recognizing which actions.

A. Clustering Implications

We employ clustering to find primitives for each action. In this section, we explain how we benefit from clustering algorithms and strategies of finding the most effective clustering configuration.

1) Clustering Techniques: Clustering is grouping together data points in a data set that are most similar to each other. Two major clustering techniques are hierarchical clustering [13] and K-means clustering [14]. In the hierarchical method, each data item is initially considered a single cluster. At each stage of the algorithm, similar clusters are grouped together to form new clusters. In the K-means algorithm, K centroids are chosen; one for each cluster. In this way, training data are grouped into a predefined number of clusters. Unlike hierarchical clustering, in which clusters are not improved after being created, the K-means algorithm iteratively improves the initial clusters. The continuously improving nature of this algorithm leads to high-quality clusters when provided appropriate data. We use this algorithm for our analysis because it is simple and operates based on the firm foundation of analysis of variances [15].

2) Cluster Validation: Although K-means is a popular clustering technique, the partition attained by this algorithm is dependent on both the initial value of centroids and the number of clusters. To increase the likelihood of arriving at a good partitioning of the data, many improvements to *K*-means have been proposed in the literature. The sum of square error (SSE) is a reasonable metric used to find the global optimal solution. To cope with the effects of initialization, we use uniformly distributed initial centers and repeatedly search for the configuration that gives the minimum error. We calculate the SSE error function as in (1), where x_i denotes the ith data item, μ_k denotes the centroid vector associated with cluster C_k, and K is the total number of clusters.

$$SSE = \sum_{k=1}^{K} \sum_{i \in C_k} (x_i - \mu_k)^2 \tag{1}$$

Another problem with K-means is that it requires prediction of the correct number of clusters. Usually, a cluster-validity framework provides insight into this problem. We employ the silhouette quality measure [16], which is robust and takes into account both intra-cluster and inter-cluster similarities to determine the quality of a cluster. Using non-normalized features, we calculate this metric in Euclidean space. Let C_k be a cluster constructed by the *K*-means algorithm. The silhouette measure assigns a quality metric S_i to the ith data item of C_k. This value signifies the confidence of the membership of the ith item to cluster C_k. S_i is defined by (2), where a_i is the average distance between the ith data item and all of the items inside cluster C_k, and b_i is the minimum of the average distances between the ith item and all of the items in each cluster besides C_k. That is, the silhouette measure compares the distance between an item and the other items in its assigned cluster to the distance between that item and the items in the nearest neighboring cluster. The larger the S_i, the higher the level of confidence about the membership of the ith sample in the training set to cluster C_k.

$$S_i = \frac{b_i - a_i}{\max\{a_i, b_i\}} \tag{2}$$

While S_i, also called *silhouette width*, describes the quality of the membership of a single data item, the quality measure of a partition, called the *silhouette index*, for a given number of clusters K is calculated using (3), where N is the number of data items in the training set.

$$Sil(K) = \frac{1}{N} \sum_{i=1}^{N} S_i \tag{3}$$

To obtain the most effective configuration in terms of the number of clusters, one can choose the *K* that has the largest silhouette index, as shown in (4).

$$\hat{K} = \arg\max_{K} \{Sil(K)\} \tag{4}$$

B. Phonology

Phonology refers to identifying basic primitives of actions. In our networked framework, primitives are distributed among sensor nodes with varying ability to recognize actions. Our phonetic description is able to characterize each action in terms of primitives using a two-stage process that are described in the following sections.

1) Primitive Construction: Primitives are created by individual sensor nodes during training. We use K-means to perform local clustering at each node, transforming the feature space into groups of dense data items. Each cluster is associated with a primitive in our model. This technique is effective since it provides insights into the usefulness of nodes for detecting each action. Actions with similar patterns tend to be assigned to the same cluster at each node, while they might be represented by different clusters at another node. Let A=$\{a_1, a_2, \ldots, a_m\}$ be a set of m actions to be classified. The clustering algorithm at node s_i will transform the actions into a series of clusters $\{P_{i1}, P_{i2}, \ldots, P_{ic}\}$, where the number of clusters c is limited to be at most m, the number of actions. The intuition behind this constraint is that similar actions will be grouped, and therefore, the total number of clusters will be less than the number of actions. We employ the validation techniques explained in Section IV-A2 to find the most effective clustering configuration.

2) Symbolization: The second step in constructing our phonetic description is to select a final group of primitives and assign symbols to them. Some of the clusters defining our initial primitives are of low quality, meaning the primitives they define will not be good representations of our actions. We refine our clusters by calculating the silhouette quality measure for each cluster and eliminating clusters that do not meet a certain threshold. The threshold is chosen to guarantee that each action falls into at least one cluster. In this way, the set of primitives at node i might be reduced to $\{P_{i1}, P_{i2}, \ldots, P_{ip}\}$, where $p \leq c$ is the number of final primitives after applying the quality measure and c is the number of original primitives. After cluster refinement, each cluster P_{ir} ($r \in \{1,2, \ldots, p\}$) is assigned a unique symbol ρ_{ir} from an alphabet Σ.

TABLE I
NOTATIONS

Symbol	Description
S=$\{s_1, s_2, \ldots, s_n\}$	set of n sensor nodes that form a BSN
i & j	indices used for sensor nodes (e.g. s_i, s_j)
n	total number of sensor nodes
A=$\{a_1, a_2, \ldots, a_m\}$	set of m actions/movements to be detected
k & l	indices used for actions (e.g. a_k, a_l)
m	total number of actions
P=$\{\rho_{ir}\}$	set of primitives created by sensor node s_i
r & t	indices used for primitives (e.g. ρ_{ir}, ρ_{it})

V. MOVEMENT IDENTIFICATION PROBLEM

Physical movement monitoring by sensor networks requires the combination of local knowledge from each node to achieve a global view of human behaviors. In this section, we study the problem of constructing a decision tree for action recognition based on the semantic subspace generated by the primitives. We use the notations in Table I throughout this section.

A. Decision Path for Action Recognition

The problem of recognizing actions using primitives can be viewed as a decision tree problem in which each internal decision node represents a sensor node and its branches the primitives identified within that node. The terminal leaves correspond to the actions to be identified. It is required that the tree identifies each action correctly. The aim of action recognition is to assign an unknown action to one of m mutually exclusive actions. The ordering of nodes in the tree changes its height and thus the time needed to converge to a solution. Given a decision tree T, path length for an action a_k ($a_k \in A$; A denotes set of all actions as shown in Table I) is defined to be the number of internal nodes in the path from root to a_k. The path length for action a_k is denoted by $\ell(a_k)$. The cost of the tree is the sum of all path lengths.

$$Cost(T) = \sum_{a_k \in A} \ell(a_k) \tag{5}$$

We first seek a linear ordering of the nodes that minimizes convergence time. Using this model, the recognition policy will explore a predefined series of sensor nodes regardless of observations made by individual nodes in real-time. Therefore, such classification is static in terms of the ordering. An optimal ordering of the sensor nodes is then a minimum-cost decision path. In an effort to make this model more efficient, we will then look for methods of constructing a full decision tree in which different branches can be taken according to the accumulative knowledge attained by currently visited sensor nodes.

To better illustrate the identification problem, we provide a simple example, shown in Fig. 3, which depicts the mapping of actions to primitives. The system consists of three sensor nodes denoted by s_1, s_2, and s_3, and four actions denoted by a_1, a_2, a_3, and a_4 which are shown using a 2-dimensional feature space (f_1 and f_2). The ellipsoids depict the distribution of different classes across the network. Dashed-line rectangles show the mapping of actions to primitives. The system has seven final primitives denoted by $\{\rho_{11}, \rho_{12}, \rho_{13}, \rho_{21}, \rho_{22}, \rho_{31}, \rho_{32}\}$. In node s_1, action a_1 is mapped to primitive ρ_{11},

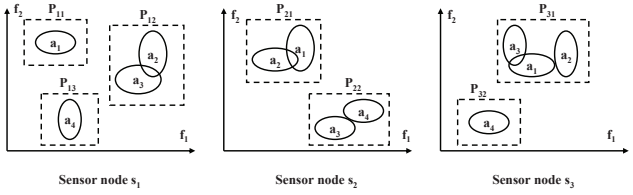

Fig. 3. An example of three nodes (s_1, s_2, s_3) and four actions (a_1, a_2, a_3, a_4). The actions are mapped to seven primitives (ρ_{11}, ρ_{12}, ρ_{13}, ρ_{21}, ρ_{22}, ρ_{31}, ρ_{32}). Each action is symbolized by corresponding primitives; a_1=$\{\rho_{11}$, ρ_{21}, $\rho_{31}\}$; a_2=$\{\rho_{12}$, ρ_{21}, $\rho_{31}\}$; a_3=$\{\rho_{12}$, ρ_{22}, $\rho_{31}\}$; a_4=$\{\rho_{13}$, ρ_{22}, $\rho_{32}\}$

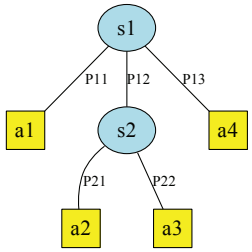

Fig. 4. A sample decision tree for the example illustrated in Fig. 3.

actions a_2 and a_3 are mapped to primitive ρ_{12}, and action a_4 is mapped to primitive ρ_{13}. In other nodes, the actions are mapped to primitives as shown. This phonetic expression can effectively describe the ability of individual nodes to identify the actions. For instance, node s_1 can distinguish action a_1 from the rest of the actions, as it finds no ambiguity when mapping an action to ρ_{11}, but it cannot distinguish between actions a_2 and a_3, as they are mapped to the same primitive. While each node has limited knowledge of the system, we require a global view in which every action is distinguished from the rest. Furthermore, we require an ordering of sensor nodes that minimizes the total time of convergence.

Given an instance of a decision problem, one can construct different decision trees. Fig. 4 illustrates a sample decision tree for the example represented in Fig. 3. The problem of finding a minimal decision tree is shown to be hard to approximate [17]. Therefore, in this section, we investigate construction of a decision path for action recognition. The method of linearly ordering the nodes restricts the shape of the decision tree so that all nodes are placed on a single path from the root and the tree has a height equal to the total number of nodes required for recognition. We will investigate the construction of a full decision tree in Section V-E.

B. Problem Formulation

In this section, we present a formal definition of our action identification problem using a decision path.

Definition 1: **(Local Discrimination Set)** Let A=$\{a_1, a_2, \ldots, a_m\}$ be a finite set of actions mapped to a set of primitives P=$\{\rho_{ir}\}$. The local discrimination set LDS_i for a node s_i is defined by:

$$LDS_i = \{(a_k, a_l) |\ a_k \in \rho_{ir}\ ,\ a_l \in \rho_{it}\ ,\ r \neq t,\quad k < l\} \tag{6}$$

The LDS_i expresses the pairs of actions that can be distinguished by s_i. In the example shown in Fig. 3, the local discrimination set for node s_1 is LDS_1=$\{(a_1$,$a_2)$,

(a_1,a_3), (a_1,a_4), (a_2,a_4), $(a_3,a_4)\}$. For the other nodes, we have LDS_2=$\{(a_1,a_3)$, (a_1,a_4), (a_2,a_3), $(a_2,a_4)\}$ and LDS_3=$\{(a_1,a_4)$, (a_2,a_4), $(a_3,a_4)\}$.

***Definition** 2:* **(Global Discrimination Set)** Let A=$\{a_1, a_2, \ldots, a_m\}$ be a finite set of actions and P=$\{\rho_{ir}\}$ a collection of primitives. The global discrimination set GDS is defined by:

$$GDS = \{(a_k, a_l) |\ a_k \in A\ ,\ a_l \in A\ ,\ k < l\} \tag{7}$$

The global discrimination set contains all the pairs of actions that are required to be distinguished from one another. For example, the global discrimination set for the system shown in Fig. 3 is GDS=$\{(a_1,a_2)$, (a_1,a_3), (a_1,a_4), (a_2,a_3), (a_2,a_4), $(a_3,a_4)\}$. In this example, the objective is to distinguish between every pair of actions.

***Definition** 3:* **(Complete Ordering)** Let A=$\{a_1, a_2, \ldots, a_m\}$ be a finite set of actions and P=$\{\rho_{ir}\}$ a collection of primitives. An ordering O=$\{s_1, s_2, \ldots, s_n\}$ is complete if the following condition holds.

$$\bigcup_{i=1}^{n} LDS_i = GDS \tag{8}$$

This indicates that the ordering is capable of distinguishing between all required pairs of actions. In the previous example, the ordering O=$\{s_1,\ s_2\}$ is complete since $\mathrm{LDS}_1 \bigcup \mathrm{LDS}_2 = \mathrm{GDS}$, but the ordering O=$\{s_2,\ s_3\}$ is not complete because this ordering cannot discriminate between actions a_1 and a_2.

***Definition** 4:* **(Ordering Cost)** Let O=$\{s_1, s_2, \ldots, s_n\}$ be a complete ordering of sensor nodes and $f(a_k)$ a function that gives the index of the first node in which the following condition holds:

$$\{(a_k, a_l) \mid k < l\ ,\ a_k \in A\} \quad \subset \quad \bigcup_{i=1}^{f(a_k)} LDS_i \tag{9}$$

That is, $f(a_k)$ is the number of nodes required to distinguish a_k from all other actions. Then the total cost of the ordering is given by the following equation:

$$Z = \sum_{a_k \in A} f(a_k) \tag{10}$$

This formulation weights the cost of an ordering so that an ordering in which more actions require fewer nodes has a lower cost. For instance, let O=$\{s_3,\ s_2,\ s_1\}$ be a complete ordering for the example shown in Fig. 3. Then $f(a_4) = 1$ because action a_4 can be completely identified by the first visited node (s_3). At the next node (s_2), action a_3 can be distinguished from the remaining actions (a_1 and a_2). Thus, $f(a_3) = 2$ because action a_3 is identified at the second node. Finally, actions a_1 and a_2 will be detected by visiting the third node (s_1) meaning that $f(a_1)$=$f(a_2)$=3. Therefore, the total cost for this ordering is 9.

***Definition** 5:* **(Min Cost Identification Problem)** Given a finite set GDS and LDS, where LDS=$\{LDS_1, LDS_2, \ldots, LDS_n\}$ is a collection of subsets of GDS such that the union of all LDS_i forms GDS, Min Cost Identification (MCI) is the problem of finding a complete linear ordering such that the cost of the ordering is minimized.

In the above example, it would be easy to find the optimal solution by a brute-force technique. We can see that the cost for the optimal ordering (s_1, s_2) is 6.

C. Problem Complexity

In this section we address the complexity of Min Cost Identification. We show that this problem is NP-hard by reduction from Min Sum Set Cover.

***Definition** 6:* **(Min Sum Set Cover)** Let U be a finite set of elements and S=$\{S_1, S_2, \ldots, S_m\}$ a collection of subsets of U such that their union forms U. A linear ordering of S is a bijection f from S to $\{1, 2, \ldots, m\}$. For each element $e \in U$ and linear ordering f, we define $f(e)$ as the minimum of $f(S)$ over all $\{S_i : e \in S_i\}$. The goal is to find a linear ordering that minimizes $\sum_e f(e)$.

***Theorem** 1:* The Min Cost Identification problem is NP-hard.

Proof: We will prove that the Min Cost Identification problem is NP-hard by reduction from Min Sum Set Cover (MSSC). Consider an MSSC instance (U,S) consisting of a finite set of elements U and a collection S of subsets of U. The objective is to find a minimum-cost linear ordering of subsets such that the union of the chosen subsets of U contains all elements in U. We now define a set $\tilde{U}$ by replacing elements of U with all elements (a_k, a_l) from the GDS. We also define $\tilde{S}$ by replacing its subsets S_i with LDS_i. $(\tilde{U}, \tilde{S})$ is an instance of the MCI problem. Therefore, MCI is NP-hard. Since solutions for the decision problem of MCI are verifiable in polynomial time, it is in NP, and consequently, the MCI decision problem is also NP-Complete. ■

***Theorem** 2:* There exists no polynomial-time approximation algorithm for MCI with an approximation ratio less than 4.

Proof: The reduction from MSSC to MCI in the proof of Theorem 1 is approximation preserving; that is, it implies that any lower bound for MSSC also holds for MCI. In [18], it is shown that for every $\varepsilon > 0$, it is NP-hard to approximate MSSC within a ratio of $4 - \varepsilon$. Therefore, 4 is also a lower bound for the approximation ratio of MCI. ■

D. Greedy Solution

The greedy algorithm for MCI is adapted from the greedy algorithm for MSSC and is shown in Algorithm 1. At each step, it searches for the node that can distinguish between the maximum number of remaining actions. It then adds such a node to the solution space and removes the actions it distinguishes from further consideration. The algorithm terminates when all pairs of actions are distinguished from each other. The approximation ratio is 4 as previously discussed.

Algorithm 1 can be used to find the minimum number and preferred locations of sensor nodes required to recognize certain actions. This can be used for power optimization because at any time, only a subset of sensor nodes will be required to be active, based on the actions of interest at that

Algorithm 1 Greedy solution for MCI

Require: Set of actions A, set of primitives P, and set of nodes S
Ensure: Linear ordering O
  calculate set LDS_i for every node s_i
  calculate set GDS
  $O = \phi$
  while $O \neq GDS$ **do**
    take node s_i such that LDS_i is maximum cardinality
    $O = O \bigcup s_i$
    for all $e \in LDS_i$ **do**
      remove e from all LDS_j (j={1,...,n})
    end for
  end while

time. Furthermore, reducing the number of required nodes helps in enhancing wearability of the BSN platform.

E. Full Decision Tree for Action Recognition

The linear ordering of the sensor nodes provides a decision tree that explores a pre-defined series of the nodes. When looking for the next node to visit, the algorithm which constructs the decision tree does not take into account the primitives to which an action might be mapped. The convergence time can further be reduced if information on primitives is taken into consideration. Each primitive within a sensor node is a mapping of several actions. An unknown action is detected by following a path in the tree starting from root and ending at a leaf node. A action mapped onto a certain primitive within may require a different path than a action mapped to another primitive. Therefore, constructing a minimum-cost decision tree is required to guarantee fastest possible action recognition.

Obtaining an optimal decision tree has been shown to be NP-hard. In the following, we present results of hardness of approximation for this problem introduced in [19] and [20]. Authors in [20] have studied decision tree problem for entity identification where an input table represents m attributes (columns) of N entities (rows). To identify an unknown entity, a decision tree is required in which each internal node is labeled with an attribute and its branches are labeled with the values that the attribute can take. The entities are placed in the leaves of the tree. The cost of a decision tree is the expected distance of an entity from the root. The goal is to construct a minimum-cost decision tree. The case of the problem where entities are equally likely is called UDT [20].

***Theorem** 3:* For any $\varepsilon > 0$, it is NP-hard to approximate the UDT problem within a ratio of $(4 - \varepsilon)$ [20].

The problem of constructing a minimum-cost decision tree for action recognition is similar in spirit to the UDT problem stated above. Sensor nodes, primitive, and actions in the primitive identification problem correspond to attributes, values, and entities in the UDT problem. Therefore, the same approximation ratio holds for our full decision tree construction problem.

***Corollary** 4:* For any $\varepsilon > 0$, it is NP-hard to approximate the problem of construction a full decision tree for action recognition, within a ratio of $(4 - \varepsilon)$.

We present a greedy algorithm to construct minimum-cost decision tree for our recognition problem. The algorithm is adapted from [20] for entity identification and shown in Algorithm 2.

Algorithm 2 Greedy approximation for full decision tree

Require: $\alpha \subset A$, a subset of actions A={a_1, a_2, ..., a_m} to be identified
Ensure: Decision tree T
  if $|\alpha| = 1$ **then**
    T is a single node $a_k \in A$
  else
    let s_i be the sensor node whose LDS_i is maximum cardinality
    create root node s_i
    for all $e \in LDS_i$ **do**
      remove e from all LDS_j (j={1,...,n})
    end for
    for all $r \in \{1, ..., B\}$ **do**
      let $\alpha_{ir} = \{a_k | a_k \in \rho_{ir}\}$
      T_j = Greedy(α_{ir})
      let s_j be the root of T_j
      add T_j to T by adding a branch from s_i to s_j
    end for
  end if

***Definition** 7:* **(Branching Factor)** Let A={a_1, a_2, ..., a_m} be a set of actions and Π_i ={ρ_{i1} ,ρ_{i2}, ..., ρ_{ip}} denotes set of primitives associated with sensor node s_i. Then the branching is the maximum number of distinct primitives among all sensor nodes, and is given by (11).

$$\mathrm{B} = \arg\max_i |\Pi_i| \tag{11}$$

The intuition behind the greedy approximation is that a good decision tree should distinguish pairs of actions at higher levels of the tree. Therefore, a natural idea is to make the sensor node that distinguishes the maximum number of pairs as the root of the tree. Assigning a sensor node s_i as the root node will partition the set of actions A into disjoint sets $\alpha_{i1}, \alpha_{i2}, ..., \alpha_{iB}$ where α_{ir}=$\{a_k | a_k \in \rho_{ir}\}$. The same greedy is applied to each of these sets to obtain B decision trees and make them the sub-trees of the root node.

***Theorem** 5:* The greedy algorithm has an approximation ratio of $O(rB \log m)$ (see [20] for proof).

In the approximation ratio for Algorithm 2, B is the branching factor, m denotes the number of input actions and rB is a Ramsey number [21] which can be suitably defined and has the value of at most $\log B$.

F. Distributed Classificaiton Algorithm

In this section, we present a distributed algorithm which uses the decision tree model for action recognition. The algorithm assumes that each node processes data locally and maps an unknown action to the clusters generated during training. We further assume that all the nodes are perfectly synchronized prior to execution of the distributed algorithm. It can be done during segmentation to ensure that all the nodes are collaborating for classification of the same action. Communication is initiated by the most informative node which is located at the root of the tree. The root node generates a token which is transmitted across the network as the classification algorithm proceeds. The computation is executed by a series of the nodes until the solution converges. Each sensor node maintains a data structure, including the decision tree structure, its local computation, and statistics received from other nodes. In particular, each node s_i keeps track of recognition convergence by a variable *Target Movement Vector* (TMV) which initially contains all actions as possible target

TABLE II
STATISTICS

Node#	Position	#Primitives	Ordering	#Actions	$\sum_{a_k \in A} f(a_k)$
1	Left-wrist	8	-	-	-
2	Left-arm	6	3	5	26
3	Right-wrist	6	2	4	11
4	Right-arm	8	-	-	-
5	Right-ankle	7	4	7	54
6	Right-thigh	5	-	-	-
7	Left-ankle	8	1	3	3
8	Left-thigh	5	5	4	74
9	Waist	7	6	2	**86**

movements. As the algorithm proceeds, each node may decide to discard some actions from the *TMV* according to the cluster membership of the action. The algorithm takes three steps as follows.

Initialization: Each sensor node s_i assigns an unknown action a_k to a cluster indicated by primitive ρ_{ir}. It further updates TMV by rejecting all actions a_l that do not belong to the cluster ρ_{ir}. Moreover, the root node generates a token which enables inter-node communication.

Transmission: A sensor node s_i transmits its local statistics only if it is the current owner of the token t. The data including updated TMV_i and the token are then transmitted to the next node in the decision tree. The next node is determined according to the assignment of the current assignment of the unknown action (ρ_{ir}).

Update: On receiving data, each node s_j updates its local Target Movement Vector TMV_j by combining the results provided by the sender node s_i. That is, s_j might reject further actions from consideration in subsequent steps of the algorithm. The receiver also checks conditions for termination. Specifically, it checks the convergence vector TMV_k which contains possible actions left. If only one action is left in the vector, the node declares a convergence and reports that action as the target action. However, if more than one action is left in TMV_k, the node would find the next node in the tree for data transmission.

VI. Experimental Analysis

This section describes experimental procedure and results that demonstrate the effectiveness of the developed action recognition algorithms.

A. Data Acquisition

To validate the proposed framework for movement classification, experiments were conducted using a BSN composed on the sensor nodes described in Section III. Five subjects aged between 22 and 55 wore nine motion sensor nodes. The nodes were placed at the positions shown in Table II. The subjects performed 25 transitional actions for ten times each. Examples of experimental actions include 'stand to sit', 'sit to lie', 'kneel', and 'jump'. A complete list of experimental actions can be found in [22]. The experiments consisted of a relatively wide range of actions that required motions from different segments of the body.

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. The tool was developed in MATLAB to split collected data into different files according to the type of movement and the sensor node that has transmitted data.

B. Data Processing

We used 50% of the data collated from all subjects as a training set and 50% as a test set. The training set was used for constructing primitives and finding the minimum-cost ordering of the nodes as well as full decision tree, while the test set was used to verify the accuracy of our recognition technique.

For each trial, the raw sensor readings were passed through a five-point moving average filter to reduce high frequency noise. 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 three-point filter was pruned out because it had a cutoff frequency of 4.3 Hz which is within the range of undesirable motions. Among the remaining filters, the filter that generates highest quality clusters during primitive construction was chosen. The Silhouette measure [16] reported the highest value for the five-point moving average filter.

For the purpose of segmentation, the video data recorded during data collection was used to capture parts of the signal that correspond to a complete action. Using video, we found the start and the end of each trial and ignored non-activity parts in subsequent processing. For each of the five data streams received from each sensor node (x, y, z acceleration and x, y angular velocity), the five features described in Section III were extracted.

C. Constructed Primitives

As previously stated, we use K-means clustering at each sensor node to create action primitives. The clustering refinement approach would map each action to one of the generated clusters. The number of extracted primitives ranges from 5

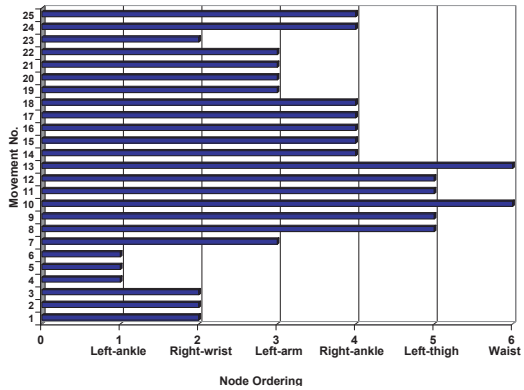

Fig. 6. Identification order of actions using decision path. The path has a total cost of 86 and an average length of 3.44 per classification

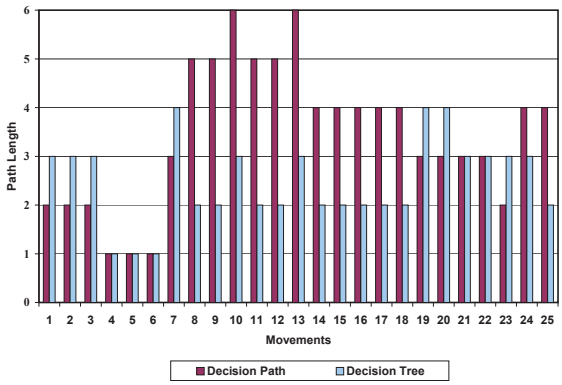

Fig. 7. Path length for each action using linear ordering vs. full decision tree.

for 'Left-thigh' and 'Right-thigh' to 8 for 'Left-wrist', 'Right-arm' and 'Left-ankle'. The number of primitives per node and the actions that are mapped to each cluster depend on the ability of the node in distinguishing between different actions. We realized that actions a_{24} and a_{25} (going upstairs and downstairs) are mapped to the same cluster on every individual node except 'Right-ankle' node, which means that the 'Right-ankle' is the only node that can distinguish these two actions, and therefore, is needed for detection of a_{24} and a_{25}. Also, the four actions a_{15} to a_{18} (moving forward, backward, or to the side) are associated with the same cluster on all the nodes except 'Right-ankle' and 'Left-ankle' nodes, meaning that 'ankle' nodes contribute most to detection of these actions.

D. Constructed Decision Trees

The greedy solution described in Algorithm 1 was used to solve the the Min Cost Identification (MCI) problem. Solution to this problem is a linear ordering of the nodes that can be used for action recognition. Given that the system consisted of 9 sensor nodes to recognize 25 actions, the size of the GDS set was 300, which is the total number of action pairs (a_k, a_l) to be distinguished. The ordering obtained by the greedy algorithm is given in Table II. It shows that in the worst case, 6 sensor nodes were sufficient to achieve a global knowledge of the current event in the system. The most informative node was the node placed on the 'Left-ankle' with ordering 1, meaning that it needs to be be the first node to visit. The value of the cost function associated with this node was 3 (last column in Table II) because it could alone distinguish three actions (actions 4, 5, and 6 as shown in Fig. 6) from all others actions. The second node on the decision path was the 'Right-wrist' node which can make a classification decision about four other actions (actions 1, 2, 3, and 23). The value of the cost function for this node was 8 which together with the first node give a collective cost of 11 as shown in the last column of Table II. The remaining actions can be classified by integrating information from 'Left-arm', 'Right-ankle', 'Left-thigh', and 'Waist' nodes as shown in Fig. 6. Each one of these nodes can detect 5, 7, 4, and 2 actions respectively.

Fig. 6 shows the nodes required to identify each action using the constructed decision path. Visited nodes are listed along the x-axis, and actions are listed along the y-axis. The actions 4, 5, and 6 which are 'Sit to lie', 'Lie to sit', and 'Sit to lie to sit' can be detected using only the 'Left-ankle' node. This is in fact due to the unique patterns of the legs during transition between 'Sit' and 'Lie' postures. The actions that can be distinguished by the second node ('Right-wrist') include actions 1, 2, 3, and 23. This implies that the actions that involve transitions between 'Stand' and 'Sit' can be identified by visiting one lower body node (e.g. 'Left-ankle') and one upper body node ('Right-wrist'). We note that the actions that involve 'Reach-up' and 'Grasp' (actions 7, 19, 20, 21, and 22) could not be classified using observations made by only one hand ('Right-wrist'). However, by visiting the third node ('Left-arm') these actions can be distinguished from the rest. This can be interpreted by the fact that the set of 'Reach-up' and 'Grasp' actions includes both 'one-hand' and 'two-hand' actions which require information from both hands to be distinguished from each other. Other actions that can be classified by visiting the three remaining nodes ('Right-ankle', 'Left-thigh', and 'Waist') can be interpreted accordingly. From Fig. 6 the average number of nodes required to detect an action is 3.44.

We used the greedy approximation described in Algorithm 2 to construct a full decision tree for the 25-action experiment. The resulting tree is shown in Fig. 5. The order for visiting nodes in the full decision tree may change depending on the action because node ordering changes based on the target action. In the full decision tree illustrated in Fig. 5, internal nodes, which correspond to the sensor nodes, are specified by light ellipsoids while the leaves of the tree that represent actions are specified by dark squares. Each link in the tree is labeled by a number corresponding to the primitive that defines branch condition to the next node. A label ρ_{ir} on edge e_{ij} indicates rth primitive within sensor node s_i that makes a branch to node s_j. The root node is the node whose local discrimination set (LDS) had maximum cardinality. The sensor node s_7 was considered to be the root node because LDS_7 had maximum cardinality (241) among all the nodes. When taking a branch, the new node was the sensor node capable of distinguishing maximum number of remaining action pairs (a_k,a_l).

For each action, the path length was defined to be the number of internal nodes in the path from root to that action.

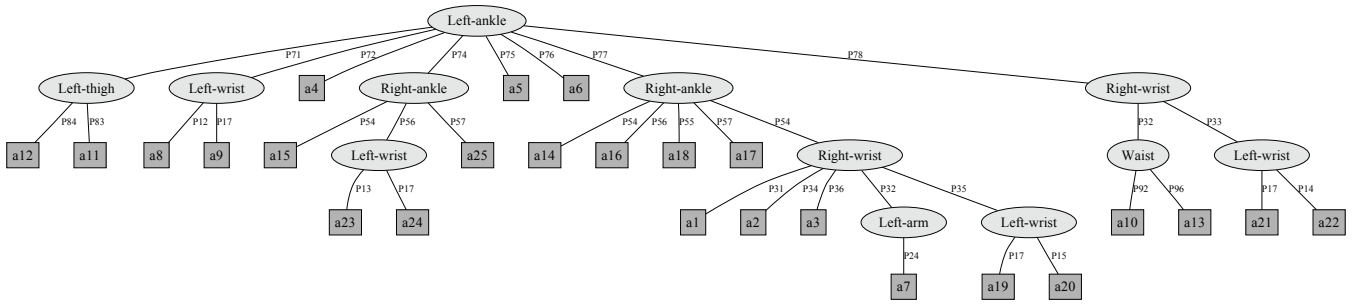

Fig. 5. **Full decision tree constructed to classify** 25 **actions. The tree has a total cost of** 62 **and an overall expected length of** 2.48

TABLE III
COMPARING SYSTEM PERFORMANCE BEFORE AND AFTER EACH OPTIMIZATION

Optimization	#Nodes	Reduction(%)	Classification	Accuracy(%)
No Optimization	9	0.0%	k-NN (at base station)	98.2%
Linear Ordering	3.44	61.8%	Decition Path)	95.7%
Branching Tree	2.46	72.4%	Full Decision Tree)	93.3%

Fig. 7 shows path length for each action using decision path and full decision tree. This value represents the number of sensor nodes required to detect each action. While the expected path length for decision path was 3.44, decision tree had an average value of 2.48 which indicates faster recognition compared with the linear ordering. Therefore, compared with the original system with 9 sensor nodes, the decision path and decision tree models achieve 61.8% and 72.4% improvements in terms of node reduction. The cost of each decision tree was measured by summation of path lengths over all actions. While the decision path had a cost of 86, the full tree had a total cost of 62.

E. Recognition Accuracy

To show the effectiveness of our action recognition algorithm using only the active nodes reported by the decision trees, we used the linear ordering of nodes as well as full decision tree to classify our test set (50% of the collected data). After filtering, segmentation, and feature extraction, each test trial was mapped to its corresponding primitives on each active node (one of ρ_{ir} clusters). The distributed algorithm described in Section V-F was used to classify each unknown action as one of the 25 predefined actions. Using the primitive representation and the decision path defined by the ordering of active nodes, we achieved an accuracy of 95.7%. Using full decision tree, we obtained an accuracy of 93.3%. In Table III, we compare the effectiveness of the two decision tree models against the original system with 9 nodes. For the case of 9 sensor nodes, we assumed that features from all the nodes are used to build a k-NN classifier at a base station. With our distributed classifier, the accuracy reduces by 2.5% and 4.9% using linear ordering and branching decision tree respectively.

F. Communication Saving

As mentioned previously, our decision tree classifier reduces the communication cost by lowering the amount of data that is needed to be transmitted across the network. Fig. 8 shows the amount of bandwidth required for classification of each

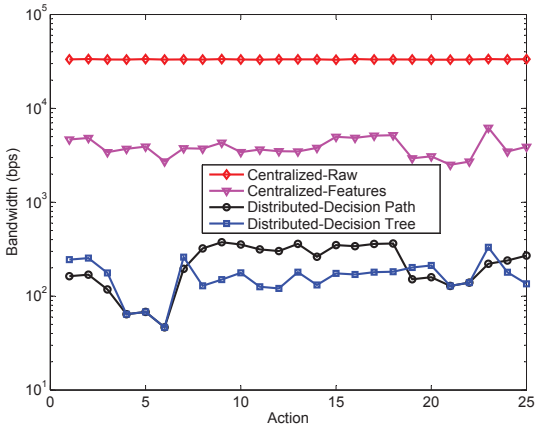

Fig. 8. Comparing communication saving of distributed classification with centralized approaches.

action using centralized and distributed algorithms. The number associated with each action represents an instantaneous bandwidth which is calculated as summation of bandwidths from all active nodes during occurrence of that action. In each classification scenario, the required bandwidth is a function of sampling frequency, number of sensors, number of nodes, and the size of data units that are transmitted. With a centralized classifier, each sensor node can transmit either raw sensor readings or a vector of statistical features to the base station. For transmitting raw data, we assume that each sensor reading is stored as a 12 bits value, which is sufficient for the readings acquired from our motion sensors. The 12 bit data unit is also enough to represent each feature.

To estimate the bandwidth required for detection of each action, the 802.15.4 frame format is used with the ratio of control to payload data being 0.2202. The control data include headers from both PHY and MAC layers. Through the following example, we explain how bandwidth is calculated for a particular action. For 'Stand to Sit', the average length of the action over all test trials is 123.9 samples which translates into 2.48 seconds. For the case of transmitting raw data, each node needs to transmit $5 \times 123.9 \times 12 = 7434$ bits over the 2.48 sec period. We note that there is a total of 5 sensors in our system, which are associated with x,y,z accelerometer and x,y gyroscope. The 7434 bits of payload data can be sent through 9 packets each accommodating a maximum of 109 bytes data. Thus, the total amount of data sent to the base station by each node is $7434 + 9 \times 192 = 9162$ bits (assuming 24 bytes of control data per packet). Since all sensor nodes are involved in

TABLE IV
NUMBER OF BASIC INSTRUCTIONS AND TOTAL NUMBER OF CYCLES FOR FEATURE EXTRACTION (FE) AND LOCAL IDENTIFICATION (CA) AS SHOWN IN FIG. 1 (ACTION LENGTH = l SAMPLES)

Task	#Add	#Mul	#Comp	#Ld/St
FE ($Mean$)	$5l$	5	0	$5l$
FE ($S2EAmp.$)	5	0	0	15
FE ($P2PAmp.$)	5	0	$10l$	$5l$
FE (RMS^2)	$5l$	$5l$	0	$5l$
FE (Std^2)	$5l$	$5l$	0	$5l$
CA	$2KF$	KF	K^2	$2KF$
Avg. #Cycles	2737	4969	45.5	3318

communication during a centralized approach, the total amount of data sent to the base station over the period of 'Stand to Sit' (2.48 sec) is $9162 \times 9 = 82458$ bits. This translates into 32.5 Kbps bandwidth.

A similar approach can be used to estimate communication cost for other classification scenarios. A decision tree based classifier uses only a subset of the nodes to detect actions. On receiving the *Target Movement Vector* (TMV), each node updates the vector according to its local results and transmits the vector to the appropriate node in the tree. We note that the TMV is a binary vector which is sized according to the number of actions. Therefore, distributed classifiers require transmission of a 25-bit vector between the nodes. Obviously, number of such transmissions varies from one action to another and depends on the number of active nodes involved in the classification. For example, only 'Left-ankle' and 'Right-wrist' nodes are involved in detection of 'Stand to Sit' using a decision path classifier (see Fig. 6). The total amount of payload in this case is 25 bits which needs to be transmitted in 2.48 seconds. A single packet can accommodate the 25 bits of payload. Adding the control data yields a total of 25 + 192 = 217 bits data. Given that two nodes are involved in the classification and each node broadcasts the TMV, this results in a bandwidth of 175 bps.

On average, we obtain bandwidths of 33.21 Kbps, 3.91 Kbps, 234 bps, and 167 bps for centralized algorithm with raw data transmission, centralized algorithm with features transmission, decision path, and full decision tree respectively.

G. Algorithm Complexity

Major computational intensive blocks in our system include feature extraction and local identification. In the following, we estimate complexity of each computing block for real-time execution on our TelosB motes. In particular, we calculate the number of 'Addition', 'Multiplication', 'Comparison' and 'Load/Store' operations for 'feature extraction', and 'cluster assignment'.

Table IV shows approximate number of operations for different processing tasks to perform local identification of an action of length l from a feature space of size F on a sensor node with K primitives (clusters). We note that morphological features do not require significant computing as they correspond to the value of sensor readings at certain times. Furthermore, features are extracted from all the five sensors including x,y,z accelerometer and x,y gyroscope. Cluster assignment is done locally to find mapping of the action to a cluster. This requires calculation of euclidean distance between an unknown action and all cluster centers on an F-dimensional feature space, and finding the closest cluster. Each summation, comparison, and read/write can be executed in 1 cycle on MSP430; however, a multiplication requires 3 cycles in presence of a hardware multiplier. In Table IV, average number of cycles are calculated based on the average length of actions ($l = 115.12$ samples), average number of clusters ($K = 6.67$), and the total number of features ($F = 75$). Given an 8 MHz clock frequency of the microcontroller on our TelosB motes, this results in 0.06% CPU utilization.

VII. DISCUSSION AND FUTURE WORK

Major contribution of our work is construction of a decision tree model for distributed action recognition in BSNs. To the best of our knowledge, this is the first study on dynamic node selection and communication enhancement based on the properties of decision trees. However, our work can be compared with several previous studies on classifying daily activities using centralized architectures. In particular, authors in [23] obtain 84% accuracy using five body-mounted accelerometers. A multi-modal system, composed of seven different sensors presented in [5] provides 90% accuracy in detecting twelve actions. Furthermore, the accuracy reported by the centralized k-NN and Naive Bayes classifiers in [22, 24, 25] is more than 90% for classification of different human actions.

Our decision tree classifier can significantly reduce energy consumption by activating a small subset of the nodes for every classification decision. This improvement is supported by MAC protocols that provide sleep/awake transmission cycles. S-MAC [26] is one of such protocols that avoid power consumption due to listening to an idle channel.

One important aspect of BSN platforms is mobility. Previous findings in [27] show that packet delivery rate varies in time as a subject moves to new physical environments. This is mainly due to the fact that inter-node transmissions occur by the radio waves that reflect off of the surfaces in the environment, making open environments less reliable in terms of communication. Therefore, a direct link might become unreliable for communication due to changes in physical location of the subject. Possible solutions to compensate with this problem include 1) increasing power level of the radio on our TelosB motes 2) replacing a direct link transmission on our decision tree with a multi-hop path where high packet delivery rates can be obtained. As suggested in [28], increase in power level can significantly reduce packet loss (4%) in close room environments such as hospitals, while packet lost ratio is less that 20% when lowest power level is used. Throughout our data collection process, the packet lost never exceeded 20% for collecting data of different nodes, subjects, actions and trials. In case of open environment, an unreliable communication link can be replaced with a path in the decision tree based on the quality of communication.

The system presented in this paper does not accommodate an automatic segmentation technique. The segmentation process is currently performed 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 [29, 30].

VIII. Conclusion

In this paper, we proposed a framework for distributed action recognition using body sensor networks. Our in-network classification operates based on a decision tree model. We presented a phoneme representation of human behavior which enables construction of optimal decision trees. We showed that the problem of determining the optimal ordering of sensor nodes for movement monitoring at the primitive level leads to constructing minimum-cost decision tree. Furthermore, we studied the problem of constructing a full decision tree and presented a greedy approximation for that. This full decision tree may further reduce the average number of sensor nodes required to detect each movement. We verified that our technique achieves 95.7% and 93.3% accuracies in classifying human actions using decision path and full decision trees respectively with a 61.8% and 72.4% average reduction in node usage.

References

[1] H. Ghasemzadeh, R. Jafari, and B. Prabhakaran, "A body sensor network with electromyogram and inertial sensors: Multimodal interpretation of muscular activities," *Information Technology in Biomedicine, IEEE Transactions on*, vol. 14, no. 2, pp. 198 –206, mar. 2010.

[2] B. Logan, J. Healey, M. Philipose, E. Tapia, and S. Intille, "A Long-Term Evaluation of Sensing Modalities for Activity Recognition," *Lecture Notes in Computer Science*, vol. 4717, p. 483, 2007.

[3] L. Klingbeil and T. Wark, "A wireless sensor network for real-time indoor localisation and motion monitoring," in *IPSN '08: Proceedings of the 7th international conference on Information processing in sensor networks*. Washington, DC, USA: IEEE Computer Society, 2008, pp. 39–50.

[4] L. Bao and S. S. Intille, "Activity recognition from user-annotated acceleration data," *Pervasive 2004*, pp. 1–17, April 2004. [Online]. Available: http://dx.doi.org/10.1007/b96922

[5] J. Lester, T. Choudhury, and G. Borriello, *A Practical Approach to Recognizing Physical Activities*. Springer-Verlag Berlin Heidelberg, 2006. [Online]. Available: http://dx.doi.org/10.1007/11748625_1

[6] H. Ghasemzadeh, V. Loseu, and R. Jafari, "Collaborative signal processing for action recognition in body sensor networks: a distributed classification algorithm using motion transcripts," in *IPSN '10: Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks*. New York, NY, USA: ACM, 2010, pp. 244–255.

[7] L. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, Feb 1989.

[8] J. Aggarwal and S. Park, "Human motion: modeling and recognition of actions and interactions," *3D Data Processing, Visualization and Transmission, 2004. 3DPVT 2004. Proceedings. 2nd International Symposium on*, pp. 640–647, Sept. 2004.

[9] 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.

[10] 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.

[11] L. Reng, T. B. Moeslund, and E. Granum, "Finding motion primitives in human body gestures," *Gesture in Human-Computer Interaction and Simulation*, vol. 3881, pp. 133–144, 2006.

[12] R. Jafari, R. Bajcsy, S. Glaser, B. Gnade, M. Sgroi, and S. Sastry, "Platform design for health-care monitoring applications," *High Confidence Medical Devices, Software, and Systems and Medical Device Plug-and-Play Interoperability, 2007. HCMDSS-MDPnP. Joint Workshop on*, pp. 88–94, June 2007.

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

[14] 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.

[15] P. Berkhin, "A survey of clustering data mining techniques," in *Grouping Multidimensional Data*. Springer Berlin Heidelberg, 2006, pp. 25–71.

[16] 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.

[17] D. Sieling, "Minimization of decision trees is hard to approximate," *J. Comput. Syst. Sci.*, vol. 74, no. 3, pp. 394–403, 2008.

[18] U. Feige and P. Tetali, "Approximating min sum set cover," *Algorithmica*, vol. 40, no. 4, pp. 219–234, 2004.

[19] B. Heeringa, "Improving access to organized information," January 2006.

[20] V. T. Chakaravarthy, V. Pandit, S. Roy, P. Awasthi, and M. Mohania, "Decision trees for entity identification: approximation algorithms and hardness results," in *PODS '07: Proceedings of the twenty-sixth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*. New York, NY, USA: ACM, 2007, pp. 53–62.

[21] R. L. Graham, B. L. Rothschild, and J. H. Spencer, *Ramsey theory*. Wiley-Interscience, 1980.

[22] 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.

[23] 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.

[24] W. H. Wu, A. A. T. Bui, M. A. Batalin, L. K. Au, J. D. Binney, and W. J. Kaiser, "Medic: Medical embedded device for individualized care," *Artif. Intell. Med.*, vol. 42, no. 2, pp. 137–152, 2008.

[25] H. Ghasemzadeh, J. Barnes, E. Guenterberg, and R. Jafari, "A Phonological Expression for Physical Movement Monitoring in Body Sensor Networks," in *Mobile Ad Hoc and Sensor Systems, 2008. MASS 2008. 5th IEEE International Conference on*, 2008, pp. 58–68.

[26] W. Ye, J. Heidemann, and D. Estrin, "An energy-efficient mac protocol for wireless sensor networks," vol. 3, 2002, pp. 1567–1576 vol.3.

[27] A. Natarajan, M. Motani, B. de Silva, K.-K. Yap, and K. C. Chua, "Investigating network architectures for body sensor networks," in *HealthNet '07: Proceedings of the 1st ACM SIGMOBILE international workshop on Systems and networking support for healthcare and assisted living environments*. New York, NY, USA: ACM, 2007, pp. 19–24.

[28] A. Natarajan, B. Silva, K. Yap, and M. Motani, "To hop or not to hop: Network architecture for body sensor networks," in *Sensor and Ad Hoc Communications and Networks, 2009. SECON '09. 6th Annual IEEE Communications Society on*, June 2009.

[29] E. Guenterberg, H. Ghasemzadeh, V. Loseu, and R. Jafari, "A distributed continuous action recognition using a hidden markov model on body sensor networks," in *DCOSS '09: International Conference on Distributed Computing in Sensor Systems*, 2009.

[30] E. Guenterberg, H. Ghasemzadeh, and R. Jafari, "A distributed hidden markov model for fine-grained annotation in body sensor networks," in *BSN '09: Proceedings of the Sixth International Workshop on Body Sensor Networks*, 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 received a Ph.D. in Computer Engineering from University of Texast at Dallas in 2010, and joined the West Wireless Health Instititue as a Postdoctoral Fellow. 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 member of the IEEE.

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.