Collaborative Signal Processing for Action Recognition in Body Sensor Networks: A Distributed Classification Algorithm Using Motion Transcripts

Hassan Ghasemzadeh, Vitali Loseu, Roozbeh Jafari
Embedded Systems and Signal Processing Lab, Department of Electrical Engineering
University of Texas at Dallas, Richardson, TX 75080-3021
{h.ghasemzadeh, vitali.loseu, rjafari}@utdallas.edu

ABSTRACT

Body sensor networks are emerging as a promising platform for remote human monitoring. With the aim of extracting bio-kinematic parameters from distributed body-worn sensors, these systems require collaboration of sensor nodes to obtain relevant information from an overwhelmingly large volume of data. Clearly, efficient data reduction techniques and distributed signal processing algorithms are needed. In this paper, we present a data processing technique that constructs motion transcripts from inertial sensors and identifies human movements by taking collaboration between the nodes into consideration. Transcripts of basic motions, called primitives, are built to reduce the complexity of the sensor data. This model leads to a distributed algorithm for segmentation and action recognition. We demonstrate the effectiveness of our framework using data collected from five normal subjects performing ten 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.

Categories and Subject Descriptors

C.3 [**Computer Systems Organization**]: Special Purpose and Application-Based Systems—*Real-time and embedded systems*; J.3 [**Computer Applications**]: Life and Medical Science—*Health*; H.1.2 [**Information Systems**]: Models and Principles—*User/Machine Systems Human information processing; Human factors.*

General Terms

Design, Algorithms, Experimentation.

Keywords

Body Sensor Networks, Collaborative Signal Processing, Distributed Classification, Motion Transcripts.

IPSN'10, April 12–16, 2010, Stockholm, Sweden.

1. 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 healthcare monitoring. They can be used in rehabilitation, sports medicine, geriatric care, gait analysis, and many other biomedical applications.

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. 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. Actions present the most interest for recognition systems since they add a temporal characteristic to the sensor observations. While actions provide more information than motions they lack realization of intelligent intent in human behavior. This role if filled with the highest level of motion abstraction called activity. An activity is a goal-oriented group of actions. 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 sound is 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 recognition accuracy while respecting the inherent limitations of our sensing platform. In BSNs, the sensor placement area is limited to a human body. These systems are usually arranged in a star topology with a base station in the middle [2]. It is commonly assumed that the base station has a larger power reserve and significantly more memory. Centralized algorithms employ the base station as the coordinator to reduce computational

stress on individual sensor nodes. This can result in nodes forwarding a significant amount data to the base station for signal processing. However, communication generally consumes more energy than local computation [3]. From the energy preservation point it is more beneficial to perform signal processing on individual nodes. This warrants 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.

In this paper, we make the following contributions: 1) we introduce a new representation of human movements, called motion transcripts, which reduces complexity of original data by transforming multidimensional signals into a sequence of symbols. 2) we propose a distributed algorithm for segmentation and classification of movements using motion transcripts. Since each movement is represented as a sequence of symbols, it enables our system to lower the amount of information stored at individual nodes, and to minimize the amount of data passed in the network. With the dynamic selection of the nodes needed for classification the overall number of active nodes is reduced.

2. RELATED WORK

Reducing the amount of active nodes is a common approach for power optimization and wearability enhancement in BSNs. One way to reduce the number of active nodes is to keep track of the performed movements and pay attention only to a subset of sensors that can observe transition out of the current motion [4]. Zappi et al. [5] 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. Authors in [6] 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. While it is easy to analyze a 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. A distributed classification scheme can be employed to potentially be able to classify a large number of actions and keep the number of active nodes low.

The concept of primitives has provided an efficient representation of human movements in computer vision domain. Using motion primitives as building blocks, Guerra-Filho et al. [7] study decomposing angles of body segments, calculated from cameras, into a well-representative language called HAL (Human Activity Language). As another example, authors in [8] 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 [9], where authors try to identify action primitives from motion capture data. Finally, authors in [10] 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 [11] where authors use edit distance to distinguish between motion primitive in 3D movement classification task. While the reviewed approaches successfully detect human actions, it is important to note that all of them rely on the information collected from all of the nodes in the network.

Table 1: 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.
Choreography (CR_j)	A concatenation of transcripts of different nodes assigned to action A_j.
Template (TPL_{ij})	A transcript which best represents action A_j as viewed by node s_i.
Class (C_{ij})	Set of observations of the same action A_j made by nodes s_i.

Several authors have developed techniques for automatic segmentation of motion sensor data. Authors in [12] present a clustering-based approach to detect and annotate daily activities (e.g. sleep) that recur regularly with similar times and durations during every given time frame (e.g. every day). Segmentation technique in [13] is based on HMM and aims to annotate a set of predefined events (e.g. initial stance when walking) from body-worn sensor nodes. These techniques focus only on segmentation and do not provide knowledge about the movement that occurs. Furthermore, they use a fixed set of sensor nodes for data processing and communication. Our approach is different in the sense that it performs simultaneous segmentation and classification of motion data and dynamically selects a subset of sensor nodes for data fusion and communication.

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

3. SYSTEM OVERVIEW

In this section we briefly describe the architecture of our system and signal processing flow. Table 1 defines some of the terms in the context of this study which are used throughout this paper.

3.1 Sensing Platform

Our system consists of several XBow® TelosB sensor nodes with custom-designed sensor boards. Each sensor board has a tri-axial accelerometer and a bi-axial gyroscope. 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

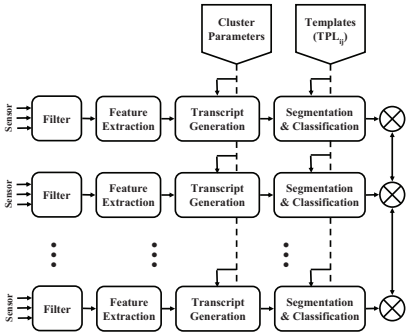


Figure 1: Signal processing for transcript generation, segmentation and distributed action recognition

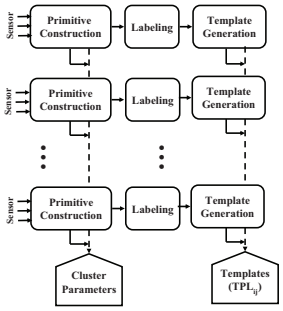


Figure 2: Training for segmentation and distributed classification

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 during training as a gold standard to mark the start and stop times associated with movements. For the prototype that is developed in this paper, the sensor readings and video are collected and synchronized in MATLAB while data from each body-worn sensor is obtained at 50 Hz. The choice of sampling frequency is important because it should provide sufficient resolution of human movements. In our system, this number is high enough to maintain this requirement. Furthermore, it satisfies the Nyquist criterion [14].

3.2 Signal Processing

A block diagram of our signal processing, transcript generation and movement classification is shown in Figure 1. The processing model requires several parameters that are measured during training as shown in Figure 2. In the following, each processing task is described briefly.

Filtering: The data collected at each node is locally filtered using a five-point moving average with the cutoff frequency of 2.4 Hz to reduce the noise. The number of points used to average the signal is chosen by examining the power spectral density of the signals. The filter is required to remove unnecessary artifacts (e.g. tremors in patients with Parkinson's disease) while maintains significant data.

Feature extraction: Features are extracted from a small moving window centered about each point of the signal stream. The features include *mean*, *standard deviation*, *root mean square*, *first* and *second derivatives*. Intuition behind choosing this set of features is that they are computationally inexpensive that can be executed on our light-weight sensor nodes. Furthermore, their effectiveness in capturing structural patterns of motion data is established through our experimental results.

Transcript generation: Each point is clustered based on the features calculated for the window surrounding it, while each cluster represents a movement primitive. A transcript is then 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. Transcript generation functions based on the clustering parameters obtained from training.

Per-node segmentation and classification: String matching technique is applied on the transcripts to detect parts of the signal that represent a specific action. Templates that are generated per movement class during training are located on the continuous data stream of characters to classify actions locally.

Distributed action recognition: An in-network processing algorithm is used to make a final decision on the current movement by combining data from most informative nodes and converging to a final decision. The node with most reliable classification decision starts propagating its local results to other nodes. On receiving data, other nodes combine the data with their local statistics and another node may decide to broadcast the accumulated results. This process continues until a target action is detected.

4. MOTION TRANSCRIPTS

A physical movement can be divided into a sequence of several smaller motions. A transcript of this movement would record order and timing of the basic 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.

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 3.2 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 challenging. A well-studied approach for grouping similar observations is clustering [15]. We use clustering analysis to group data points with consistent features to form a primitive.

Our model employs two steps for generating motion 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.

4.1 Primitive Construction

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 [16], hierarchical [17] and probabilistic model based clustering [18] have been developed. Gaussian Mixture Models (GMM) is a model based approach that 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 extensively used in speech recognition due to its ability to tolerate cluster overlap or cluster size and shape variations [19].

We use GMM as a clustering technique to define the primitives from a set of training movements. 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

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

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

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

and $\mathrm{P}(O_i)$ represents the prior probability that can be calculated by marginalization of joint probabilities as given by

$$\mathrm{P}(O_i) = \sum_k \mathrm{P}(O_i, k) \quad (3)$$

and $\mathrm{P}(k)$ 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. Therefore, the responsibility probability can be written as

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

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) [20] to find the parameters of the mixture model. For a GMM with K components, parameters of the mixing model include the mean vector and covariance matrix. As the number of components is unknown *a priori*, we perform multiple runs of the EM algorithm with varying values of K. The optimal number of clusters and the problem of choosing the best model are evaluated based on the Bayesian Information Criterion [19]. 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.

4.2 Labeling

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. Figure 3 shows transcript of a synthetic one-dimensional signal which illustrates correspondence between the primitives and signal patterns. In this figure, corresponding 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.

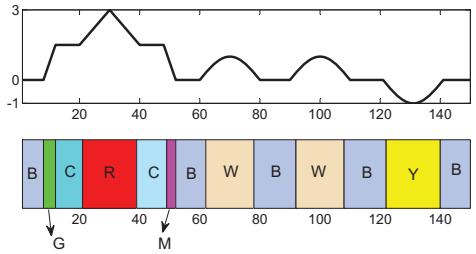


Figure 3: An example of motion transcripts generated for a one-dimensional synthetic signal.

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 technique 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. That is, each node s_i ($i \in \{1, \ldots, n\}$) requires a separate alphabet, Σ_i.

4.3 Template Generation

Our distributed action recognition requires each node to perform a local segmentation and classification before communicating with other nodes. This process is accomplished by comparing the continuous stream of characters with a set of predefined templates, which are obtained during training. Each node s_i creates a template TPL_{ij} for movement A_j, which represents all training trials of the movement observed by s_i. The template is a transcript which best represents a movement. For this purpose, we measure similarity between every pair of transcripts from a class A_j.

To compare two transcripts, a measure of similarity is required. Euclidean distance is widely used as the similarity measure when the training set is constructed based on statistical features. In our system, however, each movement is represented by a set of transcripts. Therefore, a similarity metric is required to find the difference between two strings. The *Levenshtein distance* [21], also called *edit distance*, 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.

Edit distance is used to compare every pair of transcripts within a class A_j. The transcript that has the smallest sum

of distances from all other transcripts is chosen as template:

$$TPL_{ij} = \arg\min_{s} \sum_{s \neq t} \delta(T_{ij}^{s}, T_{ij}^{t}) \quad (5)$$

where T_{ij}^{s} and T_{ij}^{t} are associated with any two training trials of movement A_j by node s_i, and δ denotes the edit distance function.

5. ACTION RECOGNITION

Action recognition aims at classifying human movements as 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. In general, a new observation of human movements can be classified in two ways. In the first method, a central classifier is designed at the base station where a new action is recognized 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. Deployment of a central classifier is not efficient in terms of communication power and bandwidth. Distributed nodes can produce redundant or overlapping information which can potentially induce extra communication. To overcome this drawback, a distributed algorithm, which combines knowledge form different nodes and operates in a real-time manner, is required. Development of such algorithm would become challenging as different sensor nodes can contribute to recognition of movements to different levels. 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.

5.1 Centralized Architecture

A centralized classifier receives data from all sensor nodes and makes a decision by combining the data using a fusion scheme. In our framework, 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) [15], a fusion technique is required to represent each trial of a movement by integrating spatially distributed transcripts. For this reason, we make a *choreography* for each trial, by concatenating corresponding transcripts from all sensor nodes and producing a new transcript.

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}.*

$$S = Concat(S_1, S_2, \ldots, S_n) \quad (6)$$

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 choreography CR_j=Concat(T_{1j}, T_{2j}, $\ldots$, 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. It is easy to see that the length function is additive with respect to the string concatenation.

In a centralized architecture, all sensor nodes transmit their local transcripts to a base station. For each observation of action A_j, a choreography CR_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 movements based on which the classifier is previously developed.

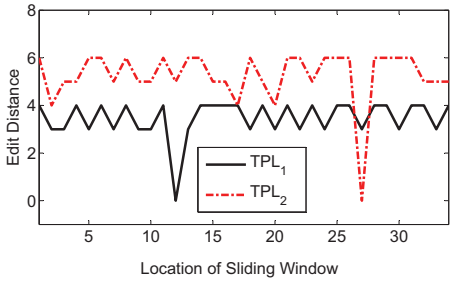
Let CR_{iq} be a choreography generated for an unknown action A_q. For each class C_{ij}, let CR_{ij} be the closest to CR_{iq} choreography generated during training. A 1-NN classifier assigns A_q to the class $A_{\hat{j}}$ such that:

$$\hat{j} = \arg\min_{j} \quad \delta(CR_{iq}, CR_{ij}) \quad (7)$$

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

... AHABBCECDDF DCBA AEFGHABCDHG EFBGCH DEFGH ...

(a) An example of motion transcript generated by a sensor node. There are two movements of interest including Mvt_1 and Mvt_2 associated with templates TPL_1="DCBA" and TPL_2="EFBGCH".



(b) String matching to detect Mvt_1 and Mvt_2. Edit distance value becomes zero due to exact matching of the transcript and corresponding templates.

Figure 4: Per-node segmentation and classification.

5.2 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 contrast, in a distributed scenario, each node makes a local decision on the target movement and may decide to propagate its local results to a next node. 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 first describe the process of segmentation and local classification that provides information that needs to be transmitted during distributed action recognition. We then explore an additive property of the edit distance which enables the distributed algorithm.

5.2.1 Segmentation and Local Classification

As the transcript generation transforms signal readings into a continuous sequence of symbols, a segmentation algorithm is required to detect portions of the transcript that correspond to a complete action. For that, the transcript

is compared with the previously generated templates. The comparison is made within each sensor node and with respect to the edit distance over a sliding window on the stream symbols. Each template has its own window which is sized according to the length of that template. Within each window, the edit distance between the transcript and each template is calculated. The distance value changes as the window moves over the stream. The transcript contains both actions of interest (target) and unknown movements. The edit distance value decreases as the moving window includes a larger part of a target action and a smaller part of actions that are not of interest. Therefore, the edit distance value decreases as the window moves closer to the action portion of the signal, and starts increases once the action is passed. When the distance function reaches a local minimum, the corresponding spot is recognized as an action. However, this information alone is not sufficient to recognize which particular action is performed. If the distance from the observed template to a template TPL_{ij} is below a certain threshold r_{ij}, then the corresponding spot can be an action A_j. The threshold is obtained during training by calculating mean and standard deviation of edit distances for each action A_j. A threshold value r_{ij} for action A_j is defined as $r_{ij} = \mu_{ij} + \sigma_{ij}$ where μ_{ij} and σ_{ij} are the mean value and standard deviation of edit distance between pairs of training transcripts. Since for multiple actions the distance may stay below the threshold, a 1-NN classifier is employed to assign an unknown spot to one of pre-specified actions. Assume an unknown spot is associated with a transcript T_{iq} representing a new action A_q which we need to classify. The node s_i measures distance between the new movement T_{iq} and nearest template. Assume TPL_{ij} denotes the nearest template to T_{iq}. The classifier assigns the new movement A_q to class $C_{i\hat{j}}$ according to (8).

$$\hat{j} = \arg\min_{j} \sum_{i=1}^{n} \delta(T_{iq}, TPL_{ij}) \tag{8}$$

A simple example of local segmentation and classification for a system with two movements of interest (Mvt_1 and Mvt_2) is shown in Figure 4. Figure 4(a) shows the stream of symbols generated by a sensor node. The two actions are represented by templates TPL_1="DCBA" and TPL_2="EFBGCH" respectively, and start at times 12 and 27. At each point in time, there are two sliding windows which are sized according to the length of TPL_1 and TPL_2 (i.e., there are two windows with sizes of 4 and 6). Figure 4(b) shows how the value of the edit distance function changes for each window (associated with TPL_1 and TPL_2). For this specific example, we assume that an exact matching (i.e. $\delta = 0$) would correspond to detection and classification of each movement. In reality, however, an approximate matching ($\delta \leq r_{ij}$) is used to specify the spot assiciated with a movement.

As stated previously, we compare each template with the motion transcript over a sliding window. We note that all the templates TPL_{ij} produced by different nodes for a particular action A_j have the same length because they are manually segmented and sized during training. That is,

$$\ell(TPL_{ij}) = \ell(TPL_{kj}) \quad \forall s_i, s_k \tag{9}$$

This property allows us to align segments across different nodes prior to running the distributed classification algorithm. Segments of the same action that are spotted by different nodes have equal lengths. If a segment detected by a node s_i is slightly delayed, we correct the time alignment by moving that segment to match with the segment that appears earliest in another node s_k.

5.2.2 Additive Property

To develop a distributed algorithm based on motion transcripts, we take advantage of additivity of edit distance with respect to concatenation. This property implies that the summation of edit distances computed locally is equal to the edit distance of the overall choreography. 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 operations described above is also additive.

A direct consequence of the additive property of the edit distance is that a global decision, on the current action occurring in the system, can be made by calculating edit distances locally (as described in Section 5.2.1) and adding them in the network to find the most similar movement. That is, a 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 not producing significant information 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.

5.2.3 Algorithm

Motivated by the idea described earlier, in this section, we derive a distributed algorithm for action recognition. 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 classification. 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 to initiate the 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 Vec-*

tor (DV) to evaluate confidence level of classification. This vector stores the distance between the unknown 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 movements 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 (10) through (13).

$$DV_i = \{\delta(T_{iq}, TPL_{i1}), ..., \delta(T_{iq}, TPL_{im})\} \tag{10}$$

$$\hat{j} = \arg\min_{j} \quad \delta(T_{iq}, TPL_{ij}) \tag{11}$$

$$\Delta_i = \frac{1}{m-1} \sum_{j \neq \hat{j}} \delta(T_{iq}, TPL_{ij}) \tag{12}$$

$$\tau_i \propto \frac{1}{\Delta_i} \tag{13}$$

where m denotes the number of actions. 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 [22, 23]. 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 correspond to a set of large distances between T_{iq} and existing classes. A large distance between T_{iq} and a class TPL_{ij} suggests that it is less likely that TPL_{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 Target Movement Vector (TMV_k) by node s_k

```
if δ(T_kq, TPL_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 (13)
end if
```

5.2.4 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. For our experiments, we calculate ϵ_j as in (14).

$$\epsilon_j = \sum_{i=1}^{n} \left[\frac{1}{M_j} \sum_{s} \delta(T_{ij}^s, TPL_{ij}) \right] \tag{14}$$

where M_j represents the number of samples in class C_{ij} and trial 's' refers to any training transcript. The idea behind choosing such value for ϵ_j is motivated by the classification decision in (8). For each movement, we calculate expected edit distance between a given trial and the movement template. This is done by calculating edit distance between every training trial 's' and the template ($\delta(T_{ij}^s, TPL_{ij})$), and taking an average over all such pairs ($\sum_s \delta(T_{ij}^s, TPL_{ij})$). By adding values from all the nodes, we compute the maximum edit distance we expect to get when a test movement is classified as A_j. 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 is needed. Although the choice of epsilon would determine the performance of the classifier, our proposed distributed classification technique can be applied independent of choice of epsilon.

Algorithm 2 Updating rejection criteria for faster convergence

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

5.2.5 Augmenting Classification

The criterion $\delta(T_{kq}, TPL_{kj}) \geq \epsilon_j$ in Algorithm 1 for rejecting movements from further processing is determined conservatively. This method may require the algorithm to go after more sensor nodes than optimally required for 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 is one of the approaches which updates the value of ϵ_j dynamically based on the number of nodes already vis-

Table 2: Experimental movements

No.	Movement
1	Stand to sit
2	Sit to lie
3	Bend and grasp
4	Kneel
5	Turning 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

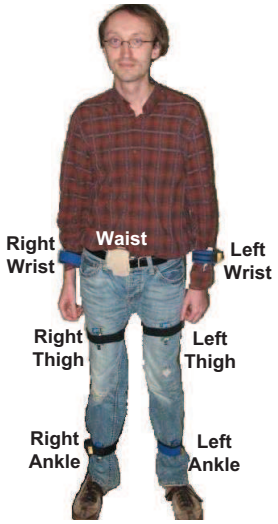


Figure 5: Experimental subject wearing seven sensor nodes.

ited. In this algorithm, n_v represents the number of nodes already considered for classification, n is the total number of sensor nodes, 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.

6. 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.

6.1 Data Collection

We developed our trial product for identifying 10 transitional movements listed in Table 2. 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 Figure 5. Subjects were asked to repeatedly perform each specific action 10 times.

The motes were programmed to sample sensors 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. This confirms previous findings in [24, 25] that use a sampling rate between 40 Hz and 50 Hz for acceleration readings.

Table 3: Speed of movements

Mvt	Range (sec.)	Mean (sec.)	Std.
1	0.9–3.1	1.98	0.51
2	1.4–3.0	1.92	0.34
3	2.0–3.8	2.71	0.50
4	2.0–2.8	2.50	0.18
5	1.8–2.9	2.37	0.23
6	2.8–4.2	3.41	0.34
7	2.1–2.9	2.46	0.22
8	1.8–3.3	2.49	0.47
9	1.6–3.0	2.16	0.36
10	2.3–3.5	2.71	0.26

Although we carried out our experiments in a controlled environment where subjects were asked to repeatedly perform each action, we did not constraint our subjects to perform actions with a specific speed. Table 3 shows range, average and standard deviation of the speed for each movement, taken over all trials. The speed was calculated after performing manual segmentation of the signals with the help of video and counting the number of samples within each trial. As the table shows, movements have a relatively wide range of speed. For example, movement "Stand to sit" has a speed of 0.9 seconds for the fastest trial while in the slowest trail it has a speed of 3.1 seconds.

6.2 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 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. 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. Filters that had cutoff frequency within the range of undesirable motions were pruned out (e.g. tremors in patients with Parkinson's disease occur at frequencies 4-5.3 Hz [26]). Among the remaining filters, the one that generates highest quality clusters (given by Silhouette measure [27]) during transcript generation was chosen.

The filtered data went through subsequent signal processing tasks including feature extraction, transcript generation and segmentation as described in Section 3.2. Motion transcripts were generated by individual nodes using separate alphabets. Figure 6 illustrates transcript of 'Reach up to cabinet' generated by the node placed on the right-wrist. For visualization, only accelerometer readings are shown in this figure. 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, A^{22} in Figure 6 accounts for the same mapping of the first 22 points.

Table 4: 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

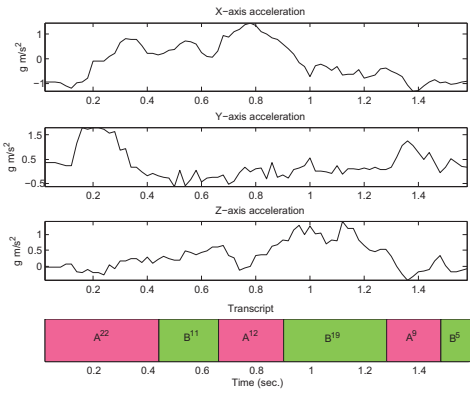


Figure 6: Transcript for a trial of 'Reach up to cabinet' generated by the right-wrist node

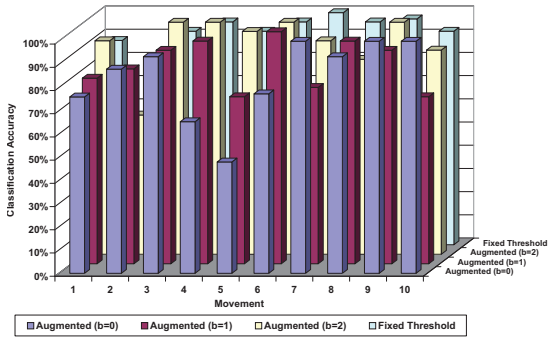


Figure 7: Classification accuracy per movement for different experimental categories.

6.3 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. Based on the resulting distance vector, a sensor node might decide to transmit its local statistics as describes in Section 5.2. 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 classification decision.

The analysis was performed 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 (see Algorithm 1). The classification accuracy was 91.33% and the average number of nodes required to converge was 5.5. In this case the same accuracy as the centralized algorithm was obtained. 2) Augmented approach with $b = 0$ (see Algorithm 2), where threshold was updated in real-time according to the number of nodes already visited. This reached an accuracy of 84.13% and 1 node in average. 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% in average and the average number of nodes was 5.3. These results are summarized in Table 4. Figure 7 shows classification accuracy for each class of movements, where movement numbers are defined in the Table 2. 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.

Table 5: Average number of active nodes

	Augmented			Fixed Threshold
Mvt	b = 0	b = 1	b = 2	
1	1.00	2.88	6.36	6.68
2	1.00	2.68	5.00	5.58
3	1.28	1.82	3.70	4.98
3	1.00	2.24	5.44	5.02
4	1.00	2.32	3.56	5.62
5	1.00	4.06	6.88	5.82
6	1.00	2.18	4.86	5.18
7	1.00	2.66	4.82	6.32
8	1.28	2.62	6.56	3.90
9	1.00	3.36	5.80	5.80
Overall:	1.05	2.68	5.29	5.51

For each test trial, our distributed algorithm accumulated results provided by several senor nodes until it converged according to the distance threshold ϵ. The value of ϵ was obtained according to the training model as discussed earlier. We calculated the number of nodes required for each trial to be classified as one of the 10 movements. This number was 1 on average for the case of augmented threshold adjustment. Furthermore, we computed the number of nodes for each group of trials associated with an action. In Table 5, we show the average number of nodes for classification of each movement. The values are categorized based on the rejection criterion.

Figure 8 shows the value of ϵ for each action based on equation (14). We recall that for each particular movement, this value represents how well that movement is separated from the rest of classes on 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.

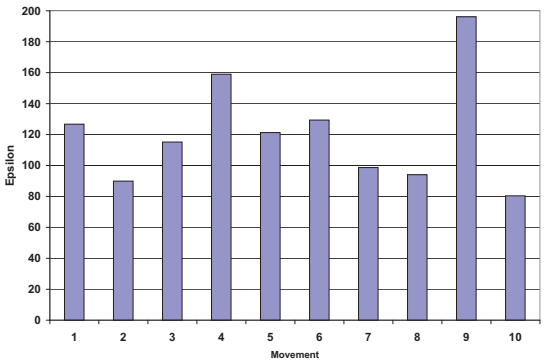


Figure 8: Value of epsilon calculated for each movement based on training data

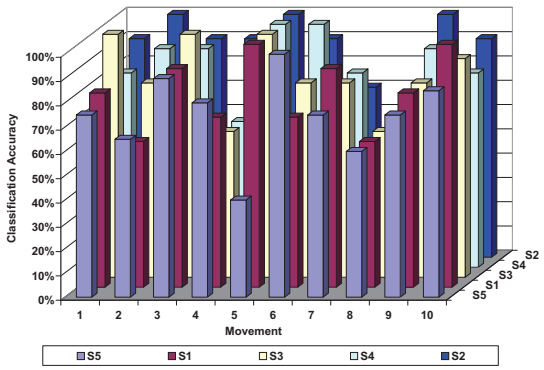


Figure 9: Classification accuracy per subject without prior training data from test subject.

Table 6: Communication cost

Mvt	Centralized Raw (Kbps)	Centralized Transcript (bps)	Distributed Distance Vector (bps)
1	20.5	245	60–415
2	20.5	252	62–350
3	20.5	182	44–220
4	20.5	196	48–240
5	20.5	210	50–285
6	20.5	147	35–205
7	20.5	196	49–252
8	20.5	196	48–305
9	20.5	231	55–218
10	20.5	182	44–257
Overall	20.5	204	49–275

6.4 Communication Cost

Table 6 displays the communication cost of different approaches we discussed in this paper. The second column shows the required bandwidth when raw sensor readings are sent directly to the base station for processing. It is a function of sampling frequency, number of sensors, and number of nodes. We assumed that each sensor reading is stored as a 12 bits value. The third column depicts the centralized approach that employs motion transcripts. Since each transcript is made of continuous chunks of the same label, we transmit only symbols and their corresponding lengths, which collectively require no more than 12 bits. For the last column, we report a range of values, since the number of communications in the distributed approach depends on the number of nodes that are involved in classification. When the most conservative configuration is used, the distributed approach has a 35% communication requirement increase compared to the centralized approach with motion transcripts. However, with the least conservative approach a 75% gain is reported.

6.5 Robustness

In this section, we demonstrate the robustness of this system to changes in target population and movement set. First, the 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 previous training data from that subject. Second, the system is used to identify unknown movements that have not been used to training.

Since five subjects participated in data collection, five different tests were conducted, each measuring classification accuracy with one subject being used for validation and others for building the training model. Figure 9 shows per subject accuracy of the classification. The five subjects are labeled as S_1, S_2, S_3, S_4 and S_5. The figure shows the classification rate for each one of the ten movements. On average, S_2 gives the highest accuracy (88%) while the lowest accuracy (75%) is due to using S_5 as the test subject. Subjects S_1, S_3 and S_4 obtain 80%, 83% and 83% classification accuracy respectively. As mentioned before, the fifth subject (S_5) has the lowest accuracy among all the subjects. Major source of misclassifications for this subject seems to be movement 5 (Turning) as shown in Figure 9. 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.

Table 7: Confusion matrix for detecting unknown movements

Test Mvt	% Classified as 1	2	3	4	5	6	unknown
7	0	0	0	26	0	0	74
8	0	0	0	14	0	0	86
9	0	0	0	6	14	0	80
10	14	0	0	6	0	0	80
overall	3.5	0	0	13	3.5	0	80

When an unknown movement occurs for which no training data exists, the system needs to report it as 'unknown'. To show the robustness of the system to new movements, the first six movements were used to train the system and the rest of the movements were used for testing. Table 7 shows how test trials are mapped to different training classes using the distributed algorithm. As mentioned earlier, when all entries are removed from the Target Movement Vector, the system declares the current movement as 'unknown'.

6.6 Algorithm Complexity

Major computational intensive blocks in our system include transcript generation and local classification. In order to estimate complexity of our transcript generation for real-time execution on the motes, we consider basic operations that are required to transform raw sensor readings into transcripts. In particular, we estimate the number of 'Addition', 'Multiplication', 'Shift', and 'Load/Store' operations that are needed for 'filtering', 'feature extraction', and 'cluster

Table 8: Number of basic instructions and total number of cycles to transform a five-dimensional vector of sampled data into a symbol.

Processing Block	#Add	#Mul	#Shift	#LD/ST
Filtering	20	5	0	5
Feature Extraction	60	40	10	0
Cluster Assignment	260	250	10	1
Total	340	295	20	6
#Cycles	340	885	20	6

assignment'. We note that once GMM clustering is developed during training, it is used to generate transcripts for each action. Transcript generation for a test trial consists of finding proper label for each data point (cluster assignment) based on maximum posterior probability criterion described in Section 4.1. Each summation, shift, and read/write can be executed in 1 cycle on MSP430 [28]; however, a multiplication requires 3 cycles in presence of a hardware multiplier. The number of cycles required to transform one sample of all existing sensors (x,y,z accelerometer and x,y gyroscope form a five-dimensional vector) into a character is listed in Table 8. Given a sampling frequency of 50 Hz, the total number of cycles for these operations is $62,550$ per seconds.

The local segmentation and classification is done by calculating inter-transcript edit distances over a sliding window (see Section 5.2.1). The edit distance function is usually implemented using dynamic programming and is quadratic in the length of transcript. As a result, complexity of local classification is $O(l^2)$ where l is the size of the sliding window. This approximately requires $25,000$ comparisons, load and store operations, which results in $125,000$ cycles per seconds. Adding the total number of cycles for transcript generation to this value, the MSP430 needs $187,550$ cycles of computation. Given an 8 MHz clock frequency of the microcontroller on our TelosB motes, this results in 2.24% CPU utilization.

7. DISCUSSION AND FUTURE WORK

The focus of our work is the distributed action recognition algorithm which dynamically selects prominent sensor nodes for movement classification. To the best of our knowledge, this is the first study on dynamic node selection by means of inertial sensors. However, our work can be compared with several previous studies on classifying daily activities based on centralized architectures. In particular, authors in [5] obtain 84% accuracy using five body-mounted accelerometers. A multi-modal system, composed of seven different sensors presented in [29], provides 90% accuracy in detecting twelve movements. The node selection approach in [30], that address the problem of node selection relies on manual selection of the best combination of nodes based on experimentation.

Our classification scheme uses motion transcripts along with a distributed algorithm to reduce the amount of data that needs to be stored and transmitted across the network. Major factors that affect resource consumption in terms of memory and communication include the number of actions and the sampling frequency. Each sensor node stores a Target Movement Vector and a Distance Vector, both of size m (number of actions). The same vectors are transmitted during classification (Section 5.2). Therefore, the amount of stored as well as communicated data increase with the number of actions. The length of each template is proportional to the sampling rate. By increasing sampling frequency, the template size grows accordingly, increasing the amount of data that is stored in each node.

Performance of our classification algorithm is independent of the sequence of movements that occur. For data collection, we asked each subject to perform each movement 10 times; however, our system can achieve the same accuracy when movements of different types occur in a sequence.

Currently, our sensing platform is used for data collection, and the signal processing modules for distributed action recognition 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].

In this paper, we did not perform an analytical study on the effectiveness of motion transcripts in detecting movements with varying speed. However, as shown in Table 3, our experiments were conducted without limitations on the speed of movements. Yet, our system achieves reasonably high classification accuracy (Table 4).

Our immediate plan for future is to build other classifiers than k-NN that use motion transcripts and operate on other similarity measures than edit distance. In particular, we plan on investigating the effectiveness of feature extraction from transcripts based on properties of N-grams [32] and constructing a distributed classifier that operates on these features in euclidean space. Our main goals of using N-grams are 1) to reduce computing complexity of edit distance calculation; 2) to detect variability in movements (e.g. variation in speed of movements).

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.

8. 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. Using transcripts of movements, 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. Our results demonstrate the effectiveness of this approach, both for reliable classification and communication reduction.

9. REFERENCES

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

[2] J. Yoo, N. Cho, and H.-J. Yoo, "Analysis of body sensor network using human body as the channel," in *BodyNets '08: Proceedings of the ICST 3rd international conference on Body area networks*. ICST, Brussels, Belgium, Belgium: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008, pp. 1–4.

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

[4] 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.

[5] 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.

[6] 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.

[7] G. Guerra-Filho, C. Fermüller, 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] D. Lymberopoulos, A. Bamis, and A. Savvides, "A methodology for extracting temporal properties from sensor network data streams," in *MobiSys '09: Proceedings of the 7th international conference on Mobile systems, applications, and services*. New York, NY, USA: ACM, 2009, pp. 193–206.

[13] 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.

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

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

[16] 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.

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

[18] 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.

[19] 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.

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

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

[22] J. Aβfalg, 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.

[23] M. Neuhaus and H. Bunke, "Graph-based multiple classifier systems a data level fusion approach," in *Image Analysis and Processing âĂŞ ICIAP 2005*, ser. Lecture Notes in Computer Science, vol. 3617. Springer Berlin / Heidelberg, 2005, pp. 479–486.

[24] G. Lyons, K. Culhane, D. Hilton, P. Grace, and D. Lyons, "A description of an accelerometer-based mobility monitoring technique," *Medical Engineering and Physics*, vol. 27, no. 6, pp. 497 – 504, 2005.

[25] W.-Y. Chung, S. Bhardwaj, A. Purwar, D.-S. Lee, and R. Myllylae, "A fusion health monitoring using ecg and accelerometer sensors for elderly persons at home," in *Engineering in Medicine and Biology Society, 2007. EMBS 2007. 29th Annual International Conference of the IEEE*, Aug. 2007, pp. 3818–3821.

[26] 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.

[27] 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.

[28] J. H. Davies, *MSP430 Microcontroller Basics*. Newton, MA, USA: Newnes, 2008.

[29] J. Lester, T. Choudhury, and G. Borriello, *A Practical Approach to Recognizing Physical Activities*. Springer-Verlag Berlin Heidelberg, 2006.

[30] L. Bao and S. S. Intille, "Activity recognition from user-annotated acceleration data," in *Pervasive*, 2004, pp. 1–17.

[31] R. Gravina et al., "Demo abstract: Spine (signal processing in node environment) framework for healthcare monitoring applications in body sensor networks," *Proc. of the 5th European conference on Wireless Sensor Networks*, 2008.

[32] A. M. Robertson, "Applications of n-grams in textual information systems," *Journal of Documentation*, vol. 54, pp. 48–67(20), 1 January 1998.