Activity Recognition from On-Body Sensors: Accuracy-Power Trade-Off by Dynamic Sensor Selection

Piero Zappi[1], Clemens Lombriser[2], Thomas Stiefmeier[2], Elisabetta Farella[1], Daniel Roggen[2], Luca Benini[1], and Gerhard Tröster[2]

[1] Department of Electronic Informatic and System, University of Bologna, Italy
{pzappi,efarella,lbenini}@deis.unibo.it
www.micrel.deis.unibo.it

[2] Wearable Computing Lab., ETH Zürich, Switzerland
{lombriser,stiefmeier,droggen,troster}@ife.ee.ethz.ch
www.wearable.ethz.ch

Abstract. Activity recognition from an on-body sensor network enables context-aware applications in wearable computing. A guaranteed classification accuracy is desirable while optimizing power consumption to ensure the system's wearability. In this paper, we investigate the benefits of dynamic sensor selection in order to use efficiently available energy while achieving a desired activity recognition accuracy. For this purpose we introduce and characterize an activity recognition method with an underlying run-time sensor selection scheme. The system relies on a meta-classifier that fuses the information of classifiers operating on individual sensors. Sensors are selected according to their contribution to classification accuracy as assessed during system training. We test this system by recognizing manipulative activities of assembly-line workers in a car production environment. Results show that the system's lifetime can be significantly extended while keeping high recognition accuracies. We discuss how this approach can be implemented in a dynamic sensor network by using the context-recognition framework *Titan* that we are developing for dynamic and heterogeneous sensor networks.

1 Introduction

Wearable computing aims at supporting people by delivering context-aware services [1]. Gestures and activities are an important aspect of the user's context. Ideally they are detected from unobtrusive wearable sensors. Gesture recognition has applications in human computer interfaces [2], or in the support of impaired people [3]. Developments in microelectronics and wireless communication enable the design of small and low-power wireless sensors nodes [4]. Although these nodes have limited memory and computational power, and may have robustness or accuracy limitations [5,6], unobtrusive context sensing can be achieved by integrating them in garments [7,8] or accessories [9].

In an activity recognition system, high classification accuracy is usually desired. This implies the use of a large number of sensors distributed over the body, depending on the activities to detect. At the same time a wearable system must be unobtrusive and operate during long periods of time. This implies minimizing sensor size, and especially energy consumption since battery technology tends to be a limiting factor in miniaturization [10].

Energy use may be reduced by improved wireless protocols [11,12], careful hardware selection [13], or duty cycling to keep the hardware in a low-power state most of the time [14]. Energy harvesting techniques may also complement battery power [15], although the unpredictability of energy supply typical of harvesting makes it difficult to manage duty cycling schedules [16].

Activity recognition requires fixed sensor sampling rate and continuous sensor node operation, since user gestures can occur at any time and maximum classification accuracy is desired. Therefore adaptive sampling rate and unpredictable duty cycling can not be used to minimize energy use. Current approaches typically rely on a small, fixed number of sensors with characteristics known and constant over time [17]. Once one sensor runs out of energy the system is not able to achieve its objective and maintenance is needed.

Here we investigate how to extend network life in an activity recognition system, while maintaing a desired accuracy, by capitalizing on an redundant number of small (possibly unreilable) sensors placed randomly over the user arms. We introduce an activity recognition system with a metaclassifier-based sensor fusion method that exploits the redundancy intrinsic in the sensor network. We modulate the number of sensors that contribute to activity recognition at runtime. Most sensor nodes are kept in low power state. They are activated when their contribution is needed to keep the desired classification accuracy, such as when active nodes fail or turn off due to lack of energy. This approach copes with dynamically changing networks without the need for retraining and allows activity recognition even in the presence of unexpected faults, thus reducing the frequency of user maintenance. The algorithm can be easily parallelized to best use the computational power of a sensor network. We show how this approach fits the *Titan* framework that we are developing for the execution of distributed context recognition algorithms in dynamic and heterogeneous wireless sensor networks.

The paper is organized as follows. In sec. 2 we describe the activity recognition algorithm with dynamic sensor selection. In sec. 3 we analyze the performance of the system in terms of classification accuracy and system life time. In sec. 4 we describe the *Titan* framework and how the activity recognition algorithms fit in it. We discuss results in sec. 5 and conclude in sec. 6.

2 Activity Recognition with Dynamic Sensor Selection

We introduce a method to recognize activities (gestures) from on-body sensors. This method relies on classifier fusion to combine multiple sensor data and comprises a dynamic sensor selection scheme. It exploits the intrinsic redundancy in a network of small and inexpensive acceleration sensors distributed on the body to achieve a desired recognition accuracy while minimizing the number of used sensors. Gesture classification is performed on individual nodes using Hidden Markov Models (HMM) [18]. A Naive Bayes classifier fuses these individual classification results to improve classification accuracy and robustness. This method is tested by recognizing the activities of assembly-line workers in a car production environment. Activity recognition enables the delivery of context-aware support to workers [19,17].

2.1 Metaclassifier for Activity Recognition

The activity recognition algorithm is based on a metaclassifier fusing the contributions from several sensor nodes [20]. The sensor nodes comprise a three-axis accelerometer to capture user motion (Analog Device ADXL330). Each axis of the accelerometer is considered as an independent sensor. Fig. 1 illustrates the activity recognition principle.

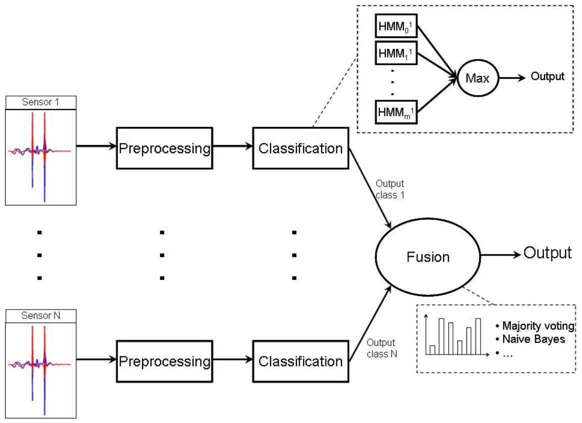

Fig. 1. Activity recognition architecture. Features extracted from the sensor data are classified by competing Hidden Markov Models (HMM), each one trained to model one activity class. The most likely model yields the class label. The labels are fused to obtain an overall classification result. Two fusing scheme have been compared: naive Bayesian and majority voting.

First on isolated instances, features are extracted from the raw acceleration data. The features are the sign of the acceleration magnitude (positive, negative or null). This is obtained by comparing the acceleration value with corresponding thresholds (-400mg and +400mg)[1]. Each sample is thus converted in one out of three possible symbols.

The features are then classified using discrete HMMs which model the gesture dynamics in the feature space. HMMs, together with Dynamic Time Warping

[1] Use of alternative features will be investigate in future works.

(DTW) [21] and neural networks [22], are a common approach to handle temporal dynamics of gestures. Our choice is motivated by previous work which showed HMMs to be a good approach [17,23]. We use ergodic HMMs with 4 states. For each accelerometer axis we train one HMM per class using the Baum-Welch algorithm starting with 15 random initial models and selecting the one that shows best classification accuracy on the training set. During activity recognition, the HMMs compete on each input sequence. The HMM best modelling the input sequence indicates the gesture class label. Training and evaluation of sequences is done using the Kevin Murphy's HMM Toolbox.

Finally, in order to end up with a single classification result we fuse the class label output from each accelerometer using a naive Bayes technique. The naive Bayes classifier is a simple probabilistic classifier based on the Bayes' theorem and the (strong) hypothesis that the input features are independent. The classifier combines the Bayes probabilistic model with a decision rule. A typical decision rule is to classify an instance as belonging to the class that maximizes the *a posteriori* probability [24].

Given the conditional model $P(C|A_0, A_1, ..., A_n)$, where C denotes the class and A_i n input attributes (in our case, the HMMs output from the sensors), we can use the Bayes theorem to define:

$$\begin{aligned} P(C|A_1, A_2, ..., A_n) &= \frac{P(A_1, A_2, ..., A_n|C)\, P(C)}{P(A_1, A_2, ..., A_n)} \\ \text{Posterior} &= \frac{\text{Likelihood} \times \text{Prior}}{\text{Marginal}} \end{aligned} \tag{1}$$

Posterior is the probability of a certain class given the input sequence. *Likelihood* is the conditional probability of a certain sequence given a certain class, *Prior* is the prior probability of the selected class, and *Marginal* is the probability of having the input sequence.

Applying the hypothesis of independence and the decision rule we obtain:

$$C_{out}(a_1, a_2, ..., a_n) = argmax_c \frac{P(C = c) \prod_{i=1}^{n} P(A_i = a_i|C = c)}{P(A_1 = a_1, A_2 = a_2, ..., A_n = a_n)} \tag{2}$$

As the denominator in equation 2 is identical for every class we only need to compute the numerator for each class and find argmax. Also, since all the classes in our experiments have the same probability, we do not need to compute $P(C = c)$. The *Likelihood* is thus the only parameter that has to be calculated. This step is achieved during training by building the confusion matrix[2] for each HMM and defining $P(A_i = a_i|C = c) = \frac{t_c}{t}$, where t_c is the number of training instances for which the class $C = c$ and the attribute $A_i = a_i$ and t is the number

[2] A confusion matrix is a visualization tool typically used in supervised learning. Each column of the matrix represents the classifier output (predicted class), while each row represents the actual class of the instances. One benefit of a confusion matrix is that it clearly shows whether the system is confusing two classes (i.e. commonly mislabeling one as another).

of training instances for class c. However, depending on the training data, for some classes c we may not have a sample for which $A_i = a_i$. In this situation, $\prod_{i=1}^{n} P(A_i = a_i|C = c)$ of that class is always zero, despite the value of the other input attribute. For this reason we used the *M-estimate* of the *Likelihood* presented in Eq. 3, where p is an *a priori* probability of a certain value for an attribute, while m is the number of virtual sample per class added to the training set. In our experiment $p = \frac{1}{10}$ and $m = 1$.

$$P(A_i = a_i|C = c) = \frac{t_c + m\,p}{t + m} \tag{3}$$

As we deal with dynamic networks where the number of active nodes varies during time, the *Posterior* probability is calculated including only the contribution of the active nodes in the network.

Feature extraction and classification can be computed in parallel on all the sensor nodes, thus allowing the exploitation of intrinsic parallelism within the sensor network, while sensor fusion is performed on a single node.

2.2 Evaluation of Activity Recognition Performance

In order to assess our approach, we consider the recognition of the activities of assembly-line workers in a car production environment. We consider the recognition of 10 activity classes (Table 1) performed in one of the quality assurance checkpoint of the production plant. These classes are a subset of 46 activities performed in this checkpoints [25].

Table 1. List of activity classes to recognize from body-worn sensors

Class	Description
0	write on notepad
1	open hood
2	close hood
3	check gaps on the front door
4	open left front door
5	close left front door
6	close both left door
7	check trunk gaps
8	open and close trunk
9	check steering wheel

We evaluate the performance of the approach in terms of correct classification ratio as a function of the number of nodes in the network. We perform a set of experiments using 19 nodes placed on the two arms of a tester (10 nodes on the right arm and due to a fault during the tests, 9 on the left arm) as illustrated in Fig. 2. Since we do not want to rely on particular positioninig and orientation of the nodes, the sensors were placed to cover the two arms without any particular

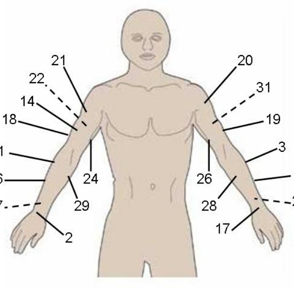

Fig. 2. Placements of the nodes on the right and left arm (dashed lines indicate nodes placed behind the arm, numbers represent the unique ID of each node)

constraints, as it is difficult to achieve such a placement for sensors unobtrusively integrated into people's garments. The subject executed 19 times each gesture listed in Table 1. Data from such trials has been recorded on a PC for subsequent analysis. Cross validation techniques have been used to extend the validation test up to all 19 instances. To perform cross validation, the input instances from the sensors have been divided into 4 folds (3 made up of 5 instances for each class and 1 of 4 instances for each class). We built 4 distinct sets of HMMs and confusion matrices. During the evaluation, for the classification of an instance we use a model obtained from a training set that did not include that specific instance.

To evaluate the correct classification ratio as a function of the number of nodes, we applied our algorithm to clusters of nodes with increasing size (one to 19 nodes). Although we consider each accelerometer axis as an independent sensor, the clusters are created in a nodewise manner. In other words a node is

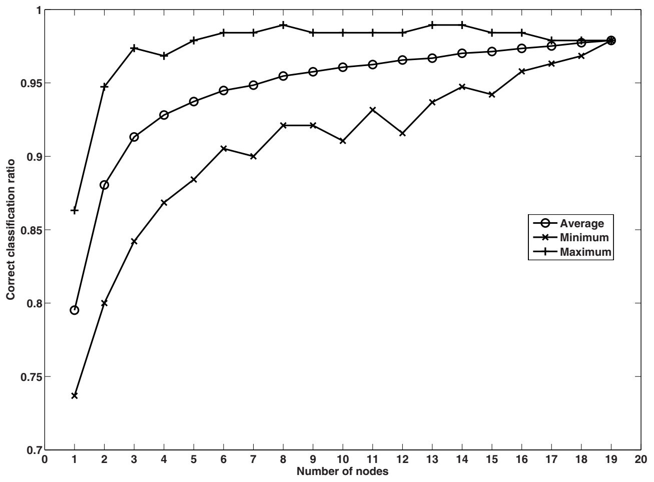

Fig. 3. Average, maximum and minimum correct classification ratio among random cluster as a function of cluster size

randomly selected and the contribution of its three axis is considered and fused. The reason is that when a node runs out of energy, the contributions of all its axes vanish. For each size we created 200 clusters from randomly selected sensor nodes. For each cluster size the average, maximum and minimum classification accuracy is recorded.

Figure 3 shows the correct classification ratio as a function of the cluster size. We achieve 98% correct classification rate using all 19 nodes and, on average, 80% using a single node. For smaller clusters the nodes composing the cluster influence the performance variance. For example, fusing the contributions from nodes 1, 3, and 24 results in 97% correct classification ratio, a value close to the accuracy that we can obtain using all the 19 nodes (Maximum curve in Fig. 3). On the other hand, fusing the outputs from nodes 20, 22 and 25 results in 84% accuracy (Minimum curve in Fig. 3) which is below what can be achieved using only one "good" node, e.g. node 16 (86%).

2.3 Dynamic Sensor Selection

We introduce a dynamic sensor selection scheme to select at run-time the sensors which are combined to perform gesture classification. This scheme seeks to achieve a desired classification accuracy while prolonging the system lifetime by minimizing the number of sensor used.

A minimum set of sensors to achieve the desired classification accuracy is first selected. Then the sensor set is updated at run-time when a sensor is removed from the network (e.g. due to failure or power loss). Since sensor nodes can fail while a gesture is performed, the algorithm ensures that the loss of a any single sensor still guarantees a performance above the desired minimum. In other words, a cluster of size D must satisfy the following condition: all subclusters of size $D-1$ must still achieve the desired minimum correct classification ratio. When a node fails, we first test wether the remaining nodes fulfill this condition. If not, all the clusters of size $D+1$ that can be built by adding one idle node to the given cluster are tested. The one that achieves the best performance is selected. If this new cluster fulfills the condition the system continues operation. If not, another idle node is added to the cluster and the process is repeated until a cluster that fulfills the condition is found or no idle nodes are left. In the latter case the system is not able to achieve the desired performance anymore.

The training instances are used to computed the expected performance of new clusters. This approach does not need system retraining, although it is valid only as long as the training set is a good representation of the user's gestures.

3 Characterization of Network Lifetime

Tests were done to assess the network lifetime (defined as the time until there are no more sensors available to achieve the desired classification accuracy) by simulating the evolution of the selected sensor set as nodes fail. For the sake of generality, we do not rely on a particular power consumption or fault model for network nodes, as it depends on the hardware and protocols chosen. In particular we are not interested in specifically identifying how long each sensor uses its radio or whether employs any kind of energy saving techniques. Instead we want to assess how our dynamic sensor selection algorithm extends network lifetime independently of these factors.

Since we assume that all nodes are identical and perform the same activity, we model node lifetime as a random variable following a Gaussian distribution with mean μ (arbitrary time units) and standard deviation as a percentage of the mean: $\alpha \times \mu$ ($\alpha < 1$)[3]. Network lifetime is then calculated as a multiple of μ. The lifetime of all the nodes is fixed at the beginning of the simulation according to this model.

The dynamic sensor selection algorithm then generates a subset of nodes able to achieve a desired accuracy even if any node of the subset fails. Then at each time step, we decrease the life of all the active nodes by one time unit. When the lifetime of a node is over, we assume that it takes a controlling unit one time unit to generate the next cluster. When no cluster matching the desired performance requirement is found the lifetime of the system is reached.

With this lifetime model only μ influences the overall system lifetime. The standard deviation has no effect on the overall system lifetime since augmenting this parameter augments the probability to see both nodes with shorter and longer lifetime thus compensating each other (see Table 2). In our tests we selected $\alpha = 0.3$.

Table 2. Network life as a function of the standard deviation chosen. Network life is calculated as a multiple of the mean node life (μ).

Standard deviation (% of mean)	10	20	30	40	50	60
Average life (time)	4.010μ	4.154μ	3.986μ	4.049μ	4.134μ	4.136μ

We compare the system lifetime when the dynamic sensor selection scheme is used to the system lifetime when all the sensors are used simultaneously (with the same node life model). In Fig. 4 the results of one trial are illustrated when the minimum accuracy required is 90%. The plot shows the performance of the network in two situations: (i) all the 19 nodes are active at the same time (dashed line); (ii) only a subset is used (continuous line). Since the objective of the network is to keep performances above 90% it can be considered expired when, due to node faults, is not possible to find a subset of nodes able to achieve such accuracy. Using all the 19 nodes together, the starting performance is higher, but quickly drops as the average node life (μ) is reached. With the dynamic sensor selection scheme, as the nodes fail (drops in the continuous line) they are replaced by inactive nodes, thus keeping the minimum required performance. Even when nodes fail, the performance never drops below the fixed threshold.

[3] The consequences of using other distribution to model the node lifetime will be explored in the future.

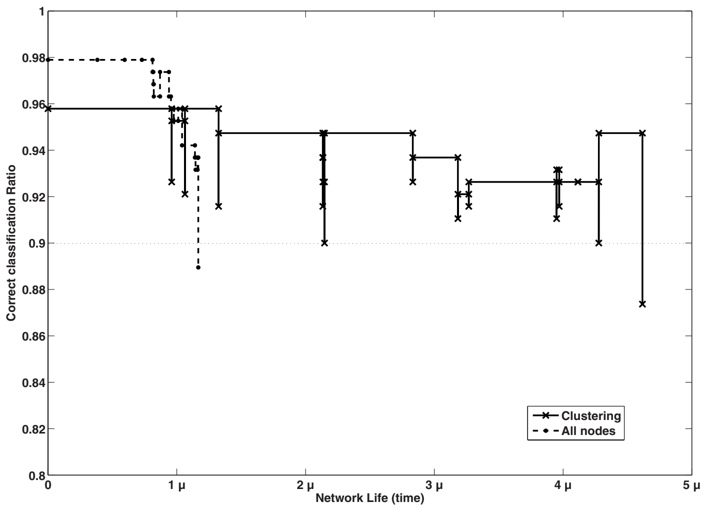

Fig. 4. Performance of the network versus time in the case of 90% minimum correct classification ratio. The network expires when its accuracy decreases under the fixed threshold (horizontal line at 0.9). Using all 19 sensors together results in a shorter life (dashed line), slightly above μ, while using dynamic sensor selection increases network life above 4μ. In the latter case, when a node fails, the performance never decreases below the fixed minimum.

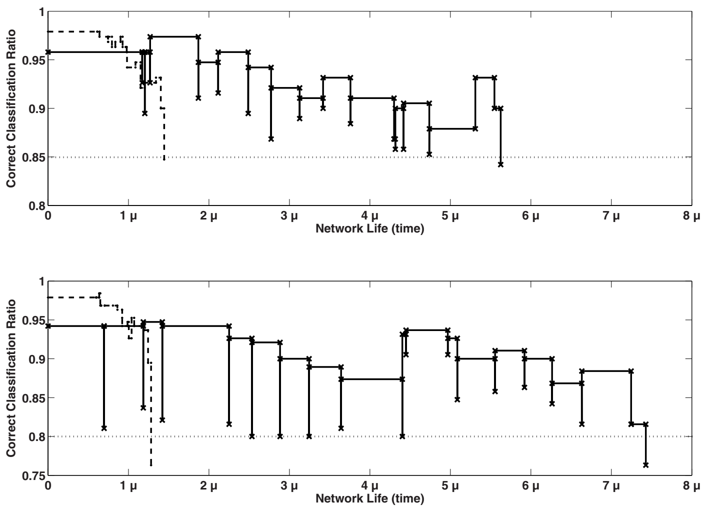

Fig. 5. Performance of the network versus time in the case that 85% (top) and 80% (below) minimum correct classification ratio

With a minimum classification accuracy of 90% the dynamic sensor selection scheme leads to a system lifetime about four times longer than when all the nodes are active. Network lifetime increases when the the minimum classification

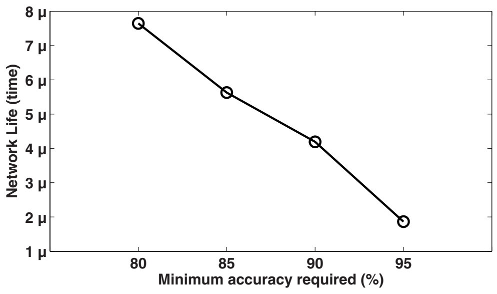

Fig. 6. Network life as a function of the minimum accuracy required

ratio is reduced. Fig. 5 shows the network lifetime for a minimum classification ratio of 80% and 85%. We performed 4 sets of 10 tests each one with increasing minimum accuracy required and calculated the average network life for each set. Fig. 6 shows that the average network life increases from around 2μ when minimum accuracy is 95% up to more than 7μ when minimum accuracy is 80%.

Without dynamic sensor selection all the sensors are used at the same time and the minimum classification accuracy does not play a role. As the nodes approach their average lifetime μ, they will fail within a short time window (related to the lifetime variance).

In Fig. 7 we illustrate (dark spots) how the network evolves over time. The size of clusters tends to increase over time. This evolution is explained by the fact that the algorithm always looks at the smallest cluster that satisfies the required accuracy. Once is not possible to find a cluster of the minimum size, the number of nodes is increased. Note also, according to our model, the life of the nodes varies according to a gaussian distribution with a standard deviation equal to 30% of the mean value.

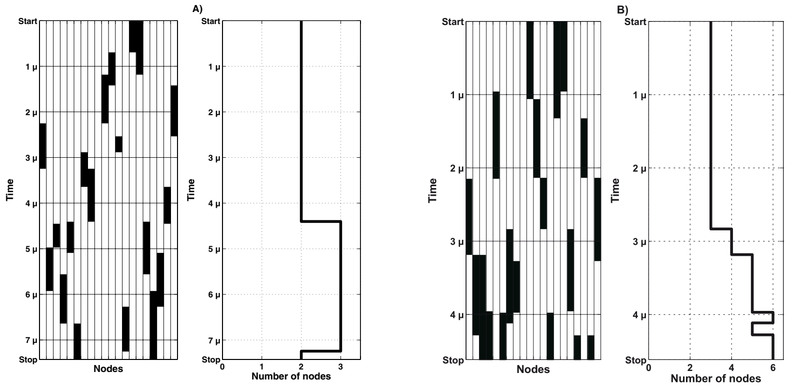

Fig. 7. Evolution of the network. On the left, in dark, are the active nodes at a certain time highlighted. On the right, the number of active nodes at a certain time is shown. A) 80% minimum accuracy. B) 90% minimum accuracy.

4 Implementation Using Tiny Task Networks (Titan)

The algorithm described above needs to be mapped on a wireless sensor network. The *Titan* framework that we are developing for context recognition in heterogeneous and dynamic wireless sensor networks can be used for this purpose [26]. We develop Titan as part of the ongoing e-SENSE project as a tool to enable and explore how context awareness can emerge in a dynamic sensor network. Titan simplifies the algorithm description, automates data exchange between selected sensor nodes, and adapts execution to dynamic network topologies. It thus qualifies for the implementation of the algorithm presented before.

Most context recognition algorithms can be described as a data flow from sensors, where data is collected, followed by feature extraction and a classification algorithm, which produces the context information. Within Titan, context recognition systems are represented as *Task Graphs*. It offers for each processing step (sampling, feature extraction, and classification) a set of predefined tasks. A task is usually a simple signal processing function, such as a filter, but may also be a more complex algorithm such as a classifier. A context recognition algorithm can be composed from those modular building blocks, which are provided by the nodes participating in the network.

A set of tasks are programmed into the sensor network nodes as a *Task Pool*. These tasks are instantiated when they are needed (i.e. they use RAM and CPU cycles only when they are used by a *Task Graph*). In a heterogeneous network, node processing power may vary, and nodes with higher processing power can provide more complex *Task Pools* than simpler nodes.

Figure 8 shows the Titan architecture and illustrates how a classification task graph is distributed on the sensor network; the *Task Graph Database* contains the classification algorithm description containing sensor tasks S_i, feature tasks F_i, a classification task C, and an actuator A_1 receiving the end result. Upon request to execute the algorithm, the *Network Manager* inspects the currently available nodes in the network, and decides on which node to instantiate what tasks,

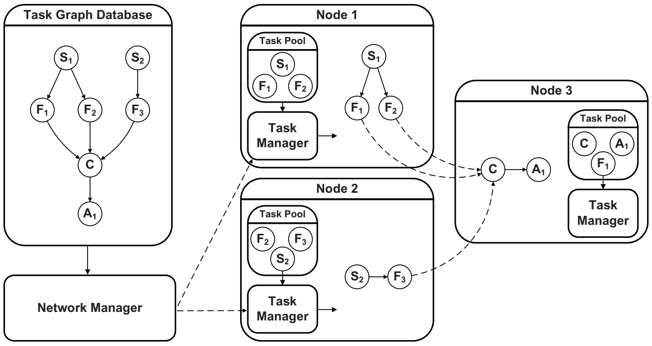

Fig. 8. Titan configures an application task graph by assigning parts of the graph to participating sensor nodes depending on their processing capabilities

such as to minimize processing load, overall power consumption, or maximise network lifetime. The Network Manager then sends a configuration message to the *Task Managers* on the sensor nodes, which instantiate the tasks on the local node. The Task Manager assigns a share of dynamic memory to the tasks for their state information and configures the connections between tasks, including transmitting data to other nodes.

During execution of the task graph, the Network Manager receives error messages from tasks or sensor nodes, and checks whether all participating sensor nodes are still alive. If changes to the current configuration are required, it adapts the distribution of the task graph on the network.

Titan provides several advantages. Ease of use, since a designer can describe his context recognition algorithm simply by interconnecting different tasks and selecting a few configuration parameters for those tasks. Portability, because it is based on TinyOS [27] which has been ported to a range of sensor network hardware and due to the abstraction of tasks, it is able to run on heterogeneous networks. Flexibility and speed, since it can reconfigure nodes in less than 1ms in order to quickly react to changes in dynamic sensor networks.

The meta classifier with dynamic sensor selection presented above can be incorporated into Titan by dividing it into a set of tasks that can be instantiated on different nodes. In particular, we define three new tasks: 1) a "gesture classification" task, which implements the HMM algorithm, 2) a "meta classification" task that performs Bayesian inference and decides the gesture class, 3) a "dynamic sensor selection" task that defines the set of sensors contributing to the meta classification task.

The initial cluster of nodes is created by the dynamic sensor selection task. The Network Manager instantiates on each of the nodes within this cluster the gesture classification task. The system runs as-is until a node fails (i.e. runs out of power). When the meta classification tasks senses that a node fails to send data it sends an error message to the Network Manager. The Network Manager instantiates the dynamic sensor selection task on a device with sufficient computational power (PDA, mobile phone), and then adapts the configuration of the nodes as needed. Since the cluster can tolerate the failure of any one of his nodes and guarantee the desired classification performance, the system can work continuously even when the dynamic sensor selection task is running. This relaxes the time constraint on this task and allows relatively complex clustering algorithms for the dynamic sensor selection task.

The task of the Network Manager for running the presented distributed gesture recognition algorithm is light-weight. To remember the current configuration of the participating nodes, it has to store just 1 byte for the node ID, 1 byte for their status (active,failed,not used,meta classifier), and a single byte for the current cluster size. This amounts to 39 bytes of storage for running the gesture recognition algorithm on our example of 19 nodes. The processing time is limited as well, as it just has to generate a small number of configuration messages at every update of the network. We are thus confident that the algorithm presented here is able to run on sensor network nodes, with the exception of the non-optimized dynamic sensor selection task which runs on a PDA or mobile phone.

5 Discussion

We have shown that by combining the fusion of classifier outputs operating on single sensors with a dynamic sensor selection scheme it is possible to extend the network lifetime while still achieving a minimum desired accuracy.

This technique may be easily used to adjust the number of sensors according to dynamically changing application constraints. Such change can be adopted as a consequence of changes in the user context (i.e. change in user location).

Active sensors may also be selected according to other criteria, such as the performance of a node as a function of the gesture. If we integrate information from the environment with the data of the sensor network, we may identify a subset of gestures that are most likely performed at a certain time. Thus active nodes may be selected among those which promise better classification performances only on that subset of gestures. However, since any change in configuration requires a set of messages to be sent among the nodes of the network, further investigation must validate this choice.

This metaclassifier is highly parallelizable and thus well suited for wireless sensor networks. Computation is shared among all active sensor nodes and none of the them is a single point of failure of the whole system. This is very important as we consider devices prone to fault or operating in environmental conditions that may severely alter the topology of the network.

Our activity recognition algorithm can find similar application in other fields of research. For example, sensor selection techniques try to extend network life by using a subset of nodes able to achieve the minimum desired performances. Such techniques are mainly used in environment monitoring [28] where dense networks cover the area of interest and sensors coverage area are overlapped.

Clustering is a fundamental research topic in sensor networks as it makes it possible to guarantee a basic level of system performance in presence of a large number of dynamically changing nodes [29]. Clustering algorithms vary depending on their application, such as guaranteeing certain latency, or balancing the activity among nodes and reducing power consumption. Energy aware clustering algorithms typically aim to reduce power consumption of the nodes either by reducing the messages sent over the wireless link by aggregating redundant data [30] or by keeping nodes in a low power state when there are other resources able to provide the same information [31].

Another research area closer to our work is feature selection. Feature selection includes a variety of techniques that aim to reduce the dimensionality of the input instances of a classifier. Some of its objectives are: reducing the measurement and storage requirements, reducing training and utilization times, defying the curse of dimensionality to improve prediction performance [32]. If we consider the HMM ouput as features, our approach may also be seen as a feature selection technique: since we dynamically select only a subset of the available ones.

Energy scavenging techniques can also take advantages from our approach. In fact now the nodes can rely on long periods when the application does not need their contribution. In such period they can collect energy and this relaxes the constraints on energy consumption due to the limited amount of energy that can be harvested from the environment. For example we showed an example of a network whose lifetime was extended by a factor 4 while still achieving 90% correct classification ratio. Since the average node life is one fourth of the total network life, each node may rely on three times its average life in order to harvest energy.

6 Conclusion

Wearable computing seeks to empower users by providing them context-aware support. Context is determined from miniature sensors integrated into garments or accessories. In a general setting the sensor network characteristics may change in unpredictable ways due to sensor degradation, interconnection failures, and jitter in the sensor placement. The use of a dense mesh of sensors distributed on the body may allow to overcome these challenges through sensor fusion techniques. Since such systems must remain unobtrusive, the reduction of node dimension and node interconnection is of high importance. Wireless sensor networks help achieving this unobtrusiveness since they do not require any wire connection. However, this implies that each sensor node must be selfpowered. In order to reduce obtrusiveness, the battery dimension must be kept at minimum, which results in low power availability.

Energy aware design aims to extend sensor nodes life by using low power devices and poweraware applications. Poweraware applications typically rely on duty cycling: they reduce the amount of time when the radio is active, and they increase the amount of time when the node can be placed in a low power state. In wearable computing, unpredictable duty cycles are proscribed. We described a different approach to extend network life while achieving desired accuracy. We capitalized on the availability of large number of nodes to implement a dynamic sensor selection scheme together with a metaclassifier that performs sensor fusion and activity recognition. This technique copes with dynamically changing number of sensor without need to retrain the system.

The method minimizes the number of nodes necessary to achieve a given classification ratio. Active nodes recognize locally gestures with hidden Markov models. The output of active nodes is fused by a naive Bayes metaclassifier. Inactive nodes are kept in a low power state. Once an active node fails the system activates one or more additional nodes to recover the initial performance. Compared to a system where all sensor nodes are continuously active, our approach can extend up to 4 times the network life while reaching 90% correct classification ratio, and up to 7 times while reaching 80% correct classification ratio. This method is highly parallelizable and well suited for wireless sensor networks.

We described how this method fits within the Titan framework that we develop to support context-aware applications in dynamic and heterogeneous sensor networks. Titan allows fast network configuration and is well suited for our technique as it allows to easily exploit network resources dynamically.

We now have demonstrated the advantage of a dynamic sensor selection scheme for accuracy-power trade-off in activity recognition. The implementation of this algorithm on wireless sensor nodes is still an open point. With qualitatively identical results, alternate classifiers and sensor selection methods that minimize computational power may be investigated. We also plan to extend the current method in order to be able to increase the inital number of nodes with on-line learning. Other future works can explore the use of an heterogeneous network that include different kind of sensors such as strain sensors or tilt sensors. Finally energy scavenging techniques benefit from our activity recognition algorithm: more time is available to harvest energy thanks to dynamic sensor selection. Evaluation of network performance with dynamically changing power availability needs to be carried out.

Acknowledgment

This work was supported by EU project WearIT@Work, contract number 004216, http://www.wearitatwork.com, and the 6th European Framework Programme Integrated Projects e-SENSE, contract number 027227, http://www.ist-e-SENSE.org.

References

1. Lukowicz, P., Junker, H., Staeger, M., von Bueren, T., Troester, G.: WearNET: A distributed multi-sensor system for context aware wearables. In: Borriello, G., Holmquist, L.E. (eds.) UbiComp 2002. LNCS, vol. 2498, pp. 361–370. Springer, Heidelberg (2002)
2. Kallio, S., Kela, J., Korpipää, P., Mäntyjärvi, J.: User independent gesture interaction for small handheld devices. International Journal of Pattern Recognition and Artificial Intelligence 20(4), 505–524 (2006)
3. Hernandez-Rebollar, J.L.: Gesture-driven american sign language phraselator. In: ICMI 2005. Proceedings of the 7th international conference on Multimodal interfaces, pp. 288–292. ACM Press, New York (2005)
4. Benini, L., Farella, E., Guiducci, C.: Wireless sensor networks: Enabling technology for ambient intelligence. Microelectron. J. 37(12), 1639–1649 (2006)
5. Watteyne, T., Augé-Blum, I., Ubéda, S.: Dual-mode real-time mac protocol for wireless sensor networks: a validation/simulation approach. In: Proceedings of the first international conference on Integrated internet ad hoc and sensor networks (2006)
6. Römer, K., Mattern, F.: The design space of wireless sensor networks. IEEE Wireless Communications 11(6), 54–61 (2004)
7. Van Laerhoven, K., Gellersen, H.W.: Spine versus porcupine: a study in distributed wearable activity recognition. In: McIlraith, S.A., Plexousakis, D., van Harmelen, F. (eds.) ISWC 2004. LNCS, vol. 3298, pp. 142–149. Springer, Heidelberg (2004)
8. Harms, H., Amft, O., Tröster, D.R.G.: Smash: A distributed sensing and processing garment for the classification of upper body postures. In: Third interational conference on body area networks (submitted, 2008)

9. Roggen, D., Bharatula, N.B., Stäger, M., Lukowicz, P., Tröster, G.: From sensors to miniature networked sensorbuttons. In: INSS 2006. Proc. of the 3rd Int. Conf. on Networked Sensing Systems, pp. 119–122 (2006)
10. Paradiso, J.A., Starner, T.: Energy scavenging for mobile and wireless electronics. IEEE Pervasive Computing 4(1), 18–27 (2005)
11. van Dam, T., Langendoen, K.: An adaptive energy-efficient mac protocol for wireless sensor networks. In: SenSys 2003: Proceedings of the 1st international conference on Embedded networked sensor systems, pp. 171–180. ACM Press, New York (2003)
12. Zigbee Alliance: Zigbee specification (2006), `http://www.zigbee.org`
13. Hill, J., Culler, D.: Mica: A Wireless Platform for Deeply Embedded Networks. IEEE Micro 22(6), 12–24 (2002)
14. Dai, L., Basu, P.: Energy and delivery capacity of wireless sensor networks with random duty-cycles. In: IEEE International Conference on Communications, pp. 3503–3510 (to appear)
15. Moser, C., Thiele, L., Benini, L., Brunelli, D.: Real-time scheduling with regenerative energy. In: ECRTS 2006. Proceedings of the 18th Euromicro Conference on Real-Time Systems, pp. 261–270. IEEE Computer Society Press, Washington (2006)
16. Vigorito, C.M., Ganesan, D., Barto, A.G.: Adaptive control of duty cycling in energy-harvesting wireless sensor networks. In: SECON 2007. 4th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks, June 18–21, 2007, pp. 21–30 (2007)
17. Stiefmeier, T., Ogris, G., Junker, H., Lukowicz, P., Tröster, G.: Combining motion sensors and ultrasonic hands tracking for continuous activity recognition in a maintenance scenario. In: 10th IEEE International Symposium on Wearable Computers (2006)
18. Rabiner, L.R.: A tutorial on hidden Markov models and selected applications in speech recognition. Proceedings of the IEEE 77(2), 257–285 (1989)
19. Maurtua, I., Kirisci, P.T., Stiefmeier, T., Sbodio, M.L., Witt, H.: A wearable computing prototype for supporting training activities in automative production. In: IFAWC. 4th International Forum on Applied Wearable Computing (2007)
20. Zappi, P., Stiefmeier, T., Farella, E., Roggen, D., Benini, L., Tröster, G.: Activity recognition from on-body sensors by classifier fusion: Sensor scalability and robustness. In: 3rd Int. Conf. on Intelligent Sensors, Sensor Networks, and Information Processing (2007)
21. Ming Hsiao, K., West, G., Vedatesh, S.M.K.: Online context recognition in multisensor system using dynamic time warping. In: Proc. of the 2005 International Conference on Intelligent Sensors, Sensor Networks and Information Processing, pp. 283–288 (2005)
22. Mitra, S., Acharya, T.: Gesture recognition: A survey. IEEE Transactions on Systems, Man and Cybernetics - Part C 37(3), 311–324 (2007)
23. Ganti, R.K., Jayachandran, P., Abdelzaher, T.F., Stankovic, J.A.: Satire: a software architecture for smart attire. In: MobiSys, pp. 110–123 (2006)
24. Rish, I., Hellerstein, J., Thathachar, J.: An analysis of data characteristics that affect naive bayes performance. In: ICML-01 (2001)
25. Stiefmeier, T., Roggen, D., Tröster, G.: Fusion of string-matched templates for continuous activity recognition. In: 11th IEEE International Symposium on Wearable Computers, pp. 41–44 (October 2007)

26. Lombriser, C., Stäger, M., Roggen, D., Tröster, G.: Titan: A tiny task network for dynamically reconfigurable heterogeneous sensor networks. In: KiVS. Fachtagung Kommunikation in Verteilten Systemen, pp. 127–138 (2007)
27. Hill, J., Szewczyk, R., Woo, A., Hollar, S., Culler, D., Pister, K.: System architecture directions for network sensors. In: Architectural Support for Programming Languages and Operating Systems (November 2000)
28. Chen, H., Wu, H., Tzeng, N.F.: Grid-based approach for working node selection in wireless sensor networks. In: IEEE International Conference on Communications, June 20–24, 2004, vol. 6, pp. 3673–3678 (2004)
29. Yu, J.Y., Chong, P.H.J.: A survey of clustering schemes for mobile ad hoc networks. IEEE Communications Surveys 7(1), 32–48 (2005)
30. Pham, T., Kim, E.J., Moh, M.: On data aggregation quality and energy efficiency of wireless sensor network protocols - extended summary. In: Proceedings of the First International Conference on Broadband Networks, pp. 730–732 (2004)
31. Guo, Y., McNair, J.: An adaptive sleep protocol for environment monitoring using wireless sensor networks. In: Communications and Computer Networks, pp. 1–6 (2005)
32. Guyon, I., Elisseeff, A.: An introduction to variable and feature selection. J. Mach. Learn. Res. 3, 1157–1182 (2003)