Mobile Phone Assisted Cooperative On-Node Processing for Physical Activity Monitoring*

Robert Diemer and Samarjit Chakraborty

Institute for Real-time Computer Systems, TU Munich, Germany
diemer@tum.de, samarjit@tum.de

Abstract. One of the main challenges in the body-area sensor network domain is to suitably break complex signal processing tasks into manageable parts in order to reduce their algorithmic complexity while retaining their output quality. The goal is to map some of these tasks onto sensor nodes and the others onto computation platforms or gateways like mobile phones. In this paper we attempt to address this problem in the specific context of physical activity monitoring. To start with, physical activity *recognition* tasks are carried out on the mobile phone. But as soon as a steady-state (e.g., walking or running at constant speed) is detected, this information is transmitted to the sensor node. At this stage, the sensor node *monitors* the known physical activity, which entails relatively simpler algorithms. In the event of a change in activity pattern, it switches back to raw data transmission and hands over processing to the mobile phone. Such cooperative signal processing significantly improves the battery life of the mobile phone as well as that of the sensor node. We present the main principles behind such distributed physical activity monitoring algorithms and compare their output quality with those from standard processing done entirely on the mobile phone.

1 Introduction

Lately, there has been a tremendous amount of interest in augmenting personal portable devices like mobile phones with lifestyle or health-monitoring applications. Sensor nodes attached to the user's body or clothing continously transmit sensed information to the mobile phone, which is then processed to record various physical activity patterns or the health condition of the user (e.g., body temperature and pulse rate). In this paper we consider a setup with a mobile phone and a small sensor board with a triaxial accelerometer, which are connected via Bluetooth. Typically, because of the limited processing capability of the microcontroller on the sensor board, all data is transmitted to the mobile phone. Several signal processing algorithms are then run on the phone in order to carry out the necessary physical activity recognition tasks. However, such continuous data reception and processing is against the standard usage pattern of a mobile phone and hence stresses its processor excessively. As a result, it is

* This work was supported by German Federal Ministry of Education and Research (01FC08069).

S.L. Min et al. (Eds.): SEUS 2010, LNCS 6399, pp. 239–251, 2010.

unable to go into deep power-down modes, thereby draining its battery quickly. In particular, our experiments show that the battery life of the mobile phone reduces significantly if it is constantly subjected to data reception from the sensor node and the associated processing tasks. Further, the continuous wireless data transmission severely shortens the battery life of the sensor node too.

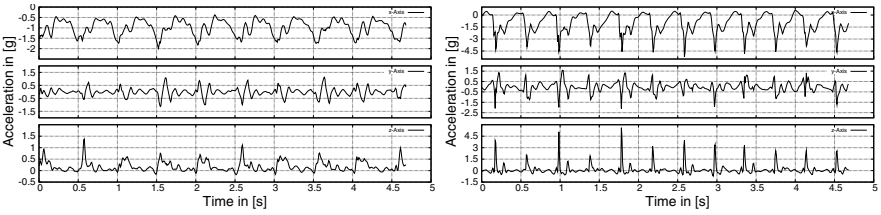
Our contributions: This is a standard problem, whose solution is to partition complex signal processing tasks and suitably map some of them onto the senor node and others onto the mobile phone with the aim of retaining activity recognition/monitoring quality and at the same time save power. In this paper we propose such a cooperative and distributed signal processing solution for physical activity recognition and monitoring. In particular, our algorithms process triaxial acceleration signals in order to classify and monitor the following activity types: level walking, walking up stairs, walking down stairs, stand-to-sit transition, and sit-to-stand transition. The first three activity types can be generally considered as *locomotive* in nature while the remaining two activity types are *postural transitions*. Besides these gait and physical activities, the general motionless resting state can also be identified using our algorithms. Since the computational capability of our sensor node [4] is limited, the physical activity *recognition* tasks are carried out on the mobile phone. But as soon as a steady-state (e.g., walking or running at constant speed) is detected, this information is transmitted to the sensor node. At this stage, the sensor node *monitors* the known physical activity, which entails relatively simpler algorithms. In the event of a change in activity pattern, it switches back to raw data transmission and hands over processing to the mobile phone. The information gathered during the monitoring phase can either be used to compute metrics like the amount of energy expended or verify that the health conditions of the user/patient are within acceptable bounds.

The main technical contribution of the paper is to suitably modify and partition standard signal processing algorithms in order to retain the original recognition and monitoring quality and at the same time save power consumption. We show that we can achieve power savings of upto 20.6% for the full system (and 26.7% for the mobile phone alone) while retaining a comparable quality of recognition and monitoring.

The rest of the paper is organized as follows. In the next section we give an overview of the standard signal processing techniques used for activity recognition and how we partition them. Our results are presented in Section 3. Finally, we conclude in Section 4 by outlining some directions for future work.

2 Physical Activity Recognition and Monitoring

Physical activity recognition using accelerometer data fits into the larger framework of context awareness and can therefore be considered as a classification problem [6]. Research efforts in the area of activity recognition have previously focused on activity classification using feature vectors in conjunction with classifiers like Naive Bayes (NB), Decision Tree (DTr), Decision Table (DT), Gaussian



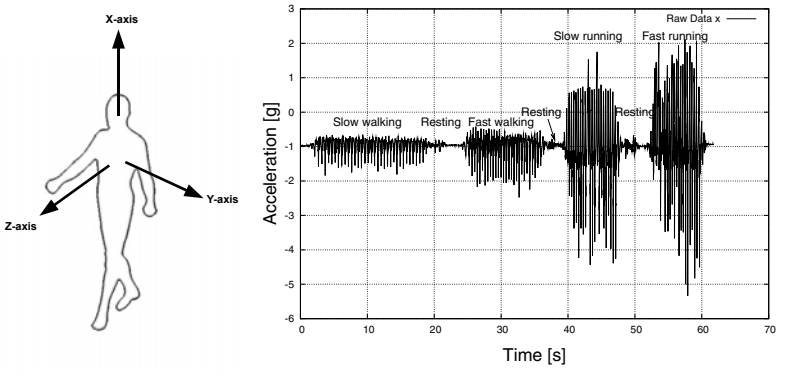
(a) Walking **(b)** Running

Fig. 1. Accelerometer data pattern for different activities

Mixture Model (GMM) and k-Nearest Neighbors (KNN) [5,6,3,2]. They have produced results with varying accuracies. In this paper we do not employ the above pattern recognition tools. Instead, our proposed algorithm makes use of pre-identified parameters in the acceleration signals in order to estimate the activity *type*. The basic concept is to design a flexible algorithm that uses a set of parameter constraints for the recognition task. The accuracy and robustness of the algorithm may be increased through additional constraints if such a need arises. This means that the accuracy of the algorithm can be adapted accordingly when the environment in which the algorithm operates, changes.

2.1 Nature of Acceleration Signals for Running and Walking

Figure 1 shows typical measurement values captured by a triaxial accelerometer during *Walking* and *Running* with the direction of the acceleration axis given in Figure 2a. In Figure 2b the x-acceleration data for several activity types is shown as an overview in order to give an impression of the obvious differences.



(a) Acceleration axis **(b)** X-acceleration data for several activity types

Fig. 2. Acceleration directions and activity types

Algorithm 1. Activity Classification (Mobile Phone)

```
   Input: x-acc[64], y-acc[64], z-acc[64]
   Output: state, steps[]
   /* Resting?                                                              */
 1 if variance (x-acc[64]) < x-threshold
 2 and variance (y-acc[64]) < y-threshold
 3 and variance (z-acc[64]) < z-threshold then
 4 |   print ("REST")
 5 |   steps[] ← 0
 6 |   state ← rest
 7 else
   |   /* Walking or Running?                                              */
 8 |   fft-coeff[] ← fourier-transform (x-acc[64])
 9 |   if max (fft-coeff[]) > threshold_{x-spec} then
   |   |   /* Perform step-detection with k-clustering algorithm           */
10 |   |   steps[] ← step-detection (x-acc[64])
11 |   |   if steps[] ≠ 0 then
12 |   |   |   print ("RUNNING")
13 |   |   |   state ← running
14 |   |   end
15 |   else
   |   |   /* Check, if periodic pattern in z-acceleration and if variance of
   |   |      y-acceleration exceeds threshold                             */
16 |   |   fft-coeff[] ← fourier-transform (z-acc[64])
17 |   |   if fft-coeff[] ≠ 0 and variance (y-acc[64]) > y-threshold-walking then
   |   |   |   /* Perform step-detection with k-clustering algorithm       */
18 |   |   |   steps[] ← step-detection (z-acc[64])
19 |   |   |   if steps[] ≠ 0 then
   |   |   |   |   /* General Walking, distinguish different walking types here... */
20 |   |   |   |   print ("WALKING")
21 |   |   |   |   state ← walking
22 |   |   |   else
   |   |   |   |   /* Do further checks here (movments, but no steps...)  */
23 |   |   |   end
24 |   |   else
   |   |   |   /* Do further checks here (movements, but not in y-axis...) */
25 |   |   end
26 |   end
27 end
28 return state, steps[]
```

The principle behind our algorithm is to define relevant parameters that allow us to effectively distinguish between different activity types. A total of eight such parameters have been defined. Algorithm 1 gives a schematic overview of how different activity types are classified. First, the variances of the data corresponding to the three axes are calculated, and if they are below a certain threshold then the activity is classified as *resting*. The next, more important parameter in determining the activity class, is the $threshold_{x-spec}$ parameter in the spectral domain. This is because $threshold_{x-spec}$ enables us to differentiate between walking and running. With parameters from the spectral domain [5], we are able to determine whether predefined frequency components above certain amplitudes exist in the frequency spectrum (as it may be seen in Figure 3). As the frequency spectrum is computed for the x-acceleration data, we do not filter this data because the x-acceleration data contains distinctive periodic patterns that are used for a peak detection algorithm to work effectively. This is why the

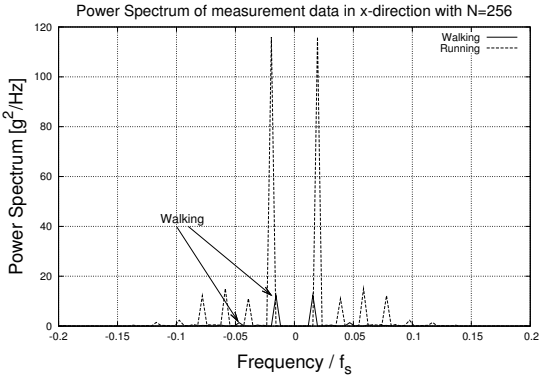


Fig. 3. Power spectrum of acceleration data in x-direction

x-acceleration data (and not the acceleration data from the remaining two axes) is crucial for detecting the activity type *running*.

We have observed that the value $threshold_{x-spec}$ has a relatively low variance across all test subjects we examined. Hence, $threshold_{x-spec}$ may be considered a relatively stable parameter. If frequency components above $threshold_{x-spec}$ are present in the spectrum, the data frame will subsequently be checked for running steps. This is possible because running motions result in a unique low frequency component in the x-acceleration spectrum that has a magnitude which is clearly above $threshold_{x-spec}$. With a k-clustering algorithm, the exact timestamp when a step occurs, is determined and the time between steps may also be calculated. If the frequency components are below $threshold_{x-spec}$, a different type of activity is ongoing, which needs to be analyzed further.

Similar to the first part of the analysis, the next part relies on the identification and definition of relevant threshold values in order to distinguish between different *walking* types. Here, spectral threshold values are used to identify frequency bins that are present above predefined threshold values. If the relevant frequency components are detected, then peaks and troughs detection is activated in order to detect the walking steps.

For walking, the important threshold values are $threshold_{z-spec}$ and th_{var-w}. The first threshold value $threshold_{z-spec}$ is similar to $threshold_{x-spec}$. It serves the same purpose as $threshold_{x-spec}$ with the difference that the spectrum being used for detecting walking belongs to the z-acceleration instead of the x-acceleration. The z-acceleration is used here because the x-acceleration spectrum for walking contains side lobes that are comparatively higher in magnitude than the z-acceleration spectrum. As a result, using the z-acceleration spectrum results in a better detection rate for walking steps. The second threshold value th_{var-w} is included in the algorithm to improve the detection results. It imposes the condition that the spread or variance of the y-acceleration must be greater than the threshold value in order for the data to be considered for steps detection. First the spectrum is checked for frequency components above the first

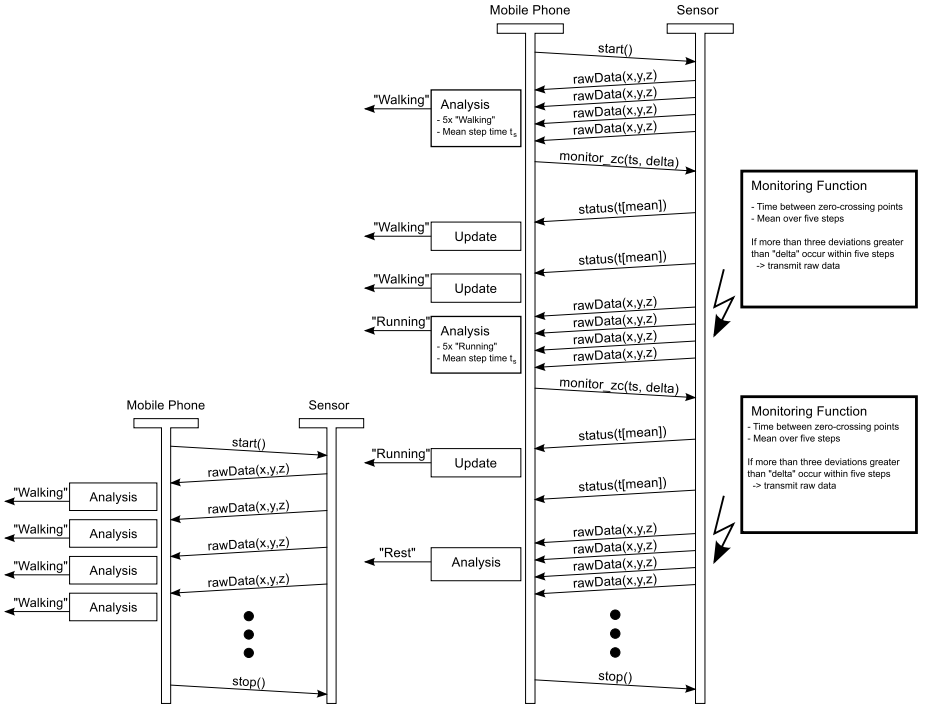


Fig. 4. Sequence charts for data transmission

threshold value. In addition to this, checks are performed so that the variance of the y-acceleration is greater than the second threshold value. If both conditions are met, the process for steps detection is activated. The threshold values for different population groups should be determined empirically as these parameters are quite stable but may depend on weight and height. The process of defining $threshold_{z-spec}$ can be seen as drawing a horizontal line across the spectrum and then checking for the existence of frequency bins above this line. This is a simple calibration technique that may be performed once before the user is equipped with this device for the first time.

2.2 Data Transmission

As already mentioned, our setup consists of a small sensor board measuring acceleration data at a sampling rate of 128 Hz, followed by transmitting it via Bluetooth. The analysis described in Section 2.1 is performed on the mobile phone each time new data packets arrive. This means that every 500 ms new data (64 values in each direction) is available and the activity classification has to be updated as shown in Figure 4a. This update rate has to be fulfilled in soft real-time for several reasons.

1. the sensor has a small amount of memory, and hence a limited amount of sensor data may be buffered,
2. the sensor has limited computational capability and energy budget and hence it is not able to handle all the computation involved,
3. often it is necessary to pass the results on to higher-level applications or to transmit them over the mobile network,
4. the user wants to get instantaneous feedback on the results of the monitoring.

The abovementioned procedure has the disadvantage of constantly engaging the mobile phone and thereby severely depleting its battery life. On the other hand, the sensor node is under-utilized, implying that it may also be used to share some of the computational tasks.

2.3 Cooperative, Distributed Analysis for Energy Optimization

The primary objective of the physical activity monitoring application is to automatically recognize different physical activities and record the relevant data for subsequent reprocessing. To achieve this a few parameters are necessary and have to be initially stored. These parameters are:

- type of physical activity
- intensity of execution
- duration

In [1] different types of physical activities are listed together with their corresponding MET[1] values. These MET values depend heavily on the specific physical activity type, e.g., walking, running, or cycling, and their performed intensity. If these MET values are known, it is then possible to determine the energy expenditure by multiplying them with their appropriate durations. An important observation is that during walking or running it is not necessary to precisely determine the duration of each step. Rather, it is sufficient to estimate the mean step duration as a base value and use it for calculating the walking or running speed and intensity. However, the main problem is the volume of data that needs to be transmitted between the sensor node and the mobile phone since the sensor node alone cannot support the computation necessary for estimating the necessary parameters.

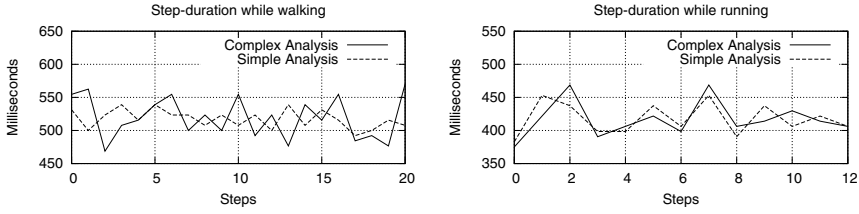
Our main observation is that the recorded sensor data is periodic in nature. If one is walking, the *shape* of the acceleration data recorded from one step is similar to the next one, provided the walking speed does not change significantly. The duration of each step can easily be computed by searching the x-acceleration data for the *zero-crossing* points with an offset of -1 g. This is shown in Algorithm 2. This algorithm can be used both for *walking* and *running* data. However, it may not be suitable to *distinguish* between these activities in a reliable fashion and nor can it *recognize* the activity itself. The solution to this problem is to perform the *recognition* task on the mobile phone. Once a periodic pattern like walking or running with a nearly constant speed is recognized, this information is

[1] Metabolic Equivalent.

Algorithm 2. State tracking (Sensor)

```
   Input: x-acc_old, x-acc_new, threshold, time_SetPoint, time_delta
   Output: steps[]
 1 if mode == zcPoint then
       /* Search crossing-points                                    */
 2     if x-acc_old < threshold and x-acc_act > threshold then
           /* Determine time between last and new crossing-point    */
 3         time_new = GetTime()
 4         time = time_act − time_old
 5         time_old = time_new
           /* Check deviation                                       */
 6         if abs (time- time_SetPoint) > time_delta then
 7             mode ← rawData
 8         else
 9             steps[] ← time
10             if Number of steps == 5 then
11                 SendStatus (steps[])
12                 clear steps[]
13             end
14         end
15     end
16 else
17     SendRawData()
18 end
```

transmitted to the sensor node and from there on the sensor node can take over the *monitoring* function. As soon as the sensor node takes over the monitoring task (along with the set-point and the maximum allowed difference from this set-point), it is able to act completely on its own. Only a *status update* is sent to the mobile phone once every five steps, containing either the step durations themselves or their mean value. Figure 4b shows the corresponding sequence chart. In summary, after a *steady-state* in the activity *recognition* is reached, the sensor node gets all the necessary information to do the *monitoring*. If the difference between the measured value and the set-point is greater than a fixed variance, raw data is transmitted and the entire process starts all over once again. With such a cooperative and distributed analysis, only the computationally expensive activity recognition task is performed on the mobile phone and the less energy consuming monitoring task is performed on the sensor node, thereby avoiding the need to transmit all the measured data. Note that the time between two zero-crossing points with an offset of -1g is a necessary but not sufficient condition for activity recognition. However, in conjunction with the apriori knowledge generated by the mobile phone, it is sufficient to track a physical activity. Without this knowledge the sensor node can no longer rely on a single parameter and has to carry out a more detailed computation involving several parameters (thereby requiring more energy and computational capacity).



(a) Step duration (*Walking*) **(b)** Step duration (*Running*)

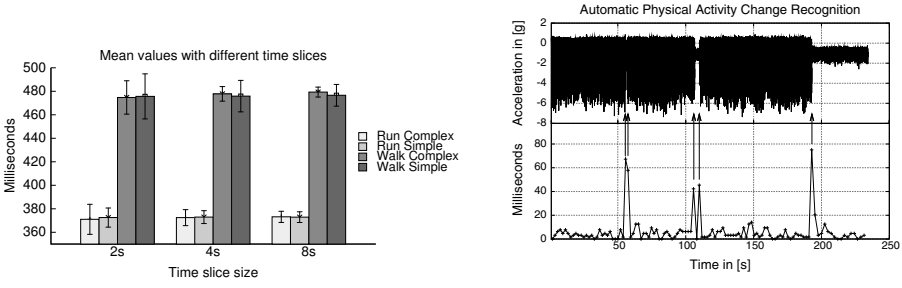
Fig. 5. Step time comparison

3 Experimental Results

In this section we compare the physical activity recognition and monitoring results obtained from the (i) purely mobile phone based implementation, and (ii) from the distributed, cooperative implementation. We also compare the energy consumptions of these two implementations. In the following discussion, the raw-data transmission and subsequent analysis on the mobile phone is referred to as the *complex* analysis/algorithm because of the multiple parameters that are used by the algorithm (as shown in Algorithm 1). On the other hand, the distributed zero-crossing Algorithm 2 is referred to as *simple* because of the comparatively fewer parameters its uses.

Figures 5a and 5b show the times between successive steps in milliseconds. Ideally, the measured step durations should be the same, but in reality small differences can be seen. During *Walking* the time between successive steps is not as similar as those during *Running*. This depends on the fact that acceleration patterns during running are more periodic and self similar compared to those during walking.

As mentioned before, not single step durations, but rather the *mean* value of these durations is important for physical activity recognition. Such mean values may either be computed over a number of steps (e.g. 5 steps) or over fixed time



(a) Mean step duration for different time slices **(b)** Deviation in mean values for interleaved *Walking* and *Running*

Fig. 6. Mean step durations for *Walking* and *Running*

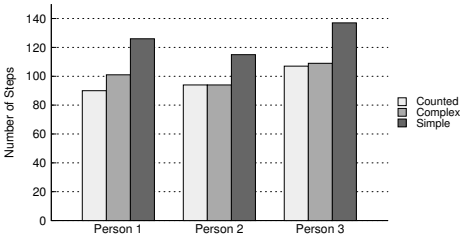


Fig. 7. Number of steps counted by visual inspection versus estimated using the proposed complex and simple algorithms

durations or slices (e.g. 3 sec). Figure 6a shows such mean values and illustrates a good conformity between the results obtained from the complex and the simple algorithms. It may be seen that the differences between the results from the complex and the simple algorithms decrease for longer time slices. In order to obtain Figure 6a, the mean value of step durations within intervals of two, four and eight seconds have been used. Here, all step durations within two, four or eight seconds have been recorded and their mean values have been computed. After this, the next time slice is used for computing the next mean value. In this manner a list of mean values is obtained. Finally, the mean of all these mean values is computed, together with the corresponding variance (as shown in Figure 6a). The entire data set consisted of four minutes of *Walking* and an additional four minutes of *Running*.

As soon as a change in physical activity occurs, the sensor switches its transmission into the raw mode. This change may be clearly seen in Figure 6b. In the upper half of the figure, the raw data from the x-acceleration axis has been shown (as in Figure 2b) and the interleaving between *Walking* and *Running* may be seen. The lower half of the figure shows the corresponding difference between two adjacent mean values. If this value is too high then it implies a change in physical activity (e.g., from *Walking* to *Running*).

Next we show the correctness of the number of estimated steps. As it can be seen in Figure 7, the complex physical activity recognition algorithm almost correctly estimates the number of steps, when compared with counting by visual inspection. Here, the data set was obtained from three different persons walking and running the same distance. All values from the same person were analyzed in order to see if the simple algorithm is able to distinguish between *Walking* and *Running* correctly. The simple algorithm, without any apriori knowledge, detects the number of steps using the deviation from mean step duration technique (with the person occassionally in the *Resting* mode). Here, the acceleration values oscillate around the offset/threshold value, resulting in the detection of "phantom" steps. This is the reason for the increased number of detected steps. The conclusion from Figure 7 is that the complex algorithm correctly detects phases without movements and during such phases does not detect steps, whereas the simple algorithm always tries to find one. However, when there is movement, the number of detected steps are within a comparable range.

The major benefit of the simple algorithm is the reduced energy consumption, especially in the mobile phone. If the simple algorithm completely executed on the mobile phone, then every measurement value would have to be transmitted and the mobile phone would not be able to get into the sleep mode.

$$E_{total}(t) = E_{data\_acq}(t) + E_{mcu}(t) + 2 \times E_{bt}(t) + E_{phone}(t) \quad (1)$$

Equation (1) shows the different components comprising the total energy consumed by the setup. First, is the energy consumption as a result of data acquisition (E_{data_acq}). The second term corresponds to the preprocessing on the microcontroller of the sensor node (E_{mcu}), the third term arises of the bluetooth communication (E_{bt}) and finally the last term arises from the analysis on the mobile phone (E_{phone}). The energy consumption of the data acquisition part (E_{data_acq}) depends on the circuit design and therefore cannot be optimized during runtime (it is static).

The energy consumed during data transmission (E_{bt}) is static too, but it may be reduced by transmitting less data. Such a reduction is achieved through the proposed cooperative distributed processing. We next give some numbers corresponding to the current consumed by the different subparts of our setup. The data acquisition part consumes about 0.65mA, the microcontroller about 2mA and the bluetooth module during transmission approx. 30mA and 2.2mA during connection in the *sniff* mode (both on the sensor node and on the mobile phone). All of these components work with the same supply voltage of 3.3V. Therefore, the energy consumption depends on the current consumption and the associated running time. The mobile phone consumes about 260mW in active mode, and only 30mW during idle mode, when the processor is sleep state. The proposed approach for energy saving relies on minimizing the amount of data to be transmitted and analyzed.

Energy consumption during data transmission is directly related to the amount of data and hence the necessary transmission time t_{bt_txrx}. This time is needed to transmit the necessary data between the sensor node and the mobile phone, where the running time for the analysis is crucial too. In the following, power consumption associated with the two cases – complex and simple – are analyzed.

Complex Analysis: In the complex case (i.e., without data analysis on the sensor node) all measurement values have to be transmitted via bluetooth to the mobile phone. Using a sample rate of $f_s = 128$Hz and two bytes per acceleration axis we get 768 bytes/sec. This has to be split into 400 byte packets, including additional informations like timestamp and packet number (16 bytes), because the sensor node does not have enough memory to buffer measurement values for one second. These packets are transmitted every 500ms. With a max. transfer rate of 721kbit s^{-1} the transmission lasts for

$$t_{bt\_txrx} = \frac{8 \times 800 bytes}{721 kbit\, s^{-1}} = 8.9ms$$

per two packets, during every second. Each time a new packet arrives on the mobile phone, the analysis algorithm is invoked. This requires approx. 16ms per run. With this information, the energy consumption over a one second period may be calculated as follows:

$$\begin{aligned}
&E_{complex}(1s) = 3.3V \times 0.65mA \times 1000ms(E_{data_acq}) + 3.3V \times 2mA \times 1000ms(E_{mcu}) \\
&+ 2 \times 3.3V \times (30mA \times 8.9ms + 2.2mA \times 991.1ms)(E_{bt}) \\
&+ 260mW \times (8.9ms + 2 \times 16ms)(E_{phone}, active) + 959.1ms \times 30mW(E_{phone}, idle) \\
&= \mathbf{64.3mWs}
\end{aligned}$$

Simple Analysis: With the optimized analysis on the sensor node, only a fraction of the data has to be transmitted to the mobile phone since the analysis is done on the sensor. With *Walking* or *Running* the time for five steps is about 1.5–2 seconds. With this, we have the same packet overhead of 16 bytes per packet and we have to transmit 2 bytes per step. For every five steps a packet consisting of $(5\times\ 2 + 16)$ bytes = 26 bytes is transmitted. This occurs every 1.5 seconds, so we get a transmission time of:

$$t_{bt\_txrx} = \frac{8 \cdot 26/1.5\text{bytes}}{721kbit\,s^{-1}} = 0.19ms$$

over every second. From this, the energy consumption may be calculated over a one second period in an analogous way:

$$\begin{aligned}
&E_{simple}(1s) = 3.3V \times 0.65mA \times 1000ms(E_{data_acq}) + 3.3V \times 2mA \times 1000ms(E_{mcu}) \\
&+ 2 \times 3.3V \times (30mA \times 0.19ms + 2.2mA \times 999.81ms)(E_{bt}) \\
&+ 260mW \times (0.19ms + 2 \times 0.05ms)(E_{phone}, active) + 999.71ms \times 30mW(E_{phone}, idle) \\
&= \mathbf{53.3mWs}
\end{aligned}$$

These numbers show that the complex analysis needs $\frac{64.3-53.3mWs}{53.3mWs} = 20.6\%$ more energy compared to the simple algorithm. As already mentioned, the bottleneck in the overall system is the energy consumption in the mobile phone, which is composed of the consumption at the processor and in the Bluetooth module. In the mobile phone, the complex algorithm needs $\frac{47.5-37.5mWs}{37.5mWs} = 26.7\%$ more energy compared to the proposed simple one.

4 Concluding Remarks

In this paper we have proposed distributed, cooperative signal processing algorithms for physical activity recognition and monitoring. Our proposed algorithms perform a considerable amount of processing on the sensor node itself, thereby reducing the volume of data to be communicated to the mobile phone. Processing all the data on the mobile phone – using standard signal processing algorithms – require nearly 26% higher energy consumption. As a part of future work, we will investigate individual-specific dynamically adapted thresholds (in contrast to static ones) for better recognition quality.

References

1. Ainsworth, B.: The compendium of physical activities tracking guide (January 2002), http://prevention.sph.sc.edu/tools/compendium.htm
2. Bao, L., Intille, S.S.: Activity recognition from user-annotated acceleration data. In: Ferscha, A., Mattern, F. (eds.) PERVASIVE 2004. LNCS, vol. 3001, pp. 1–17. Springer, Heidelberg (2004)
3. Bidargaddi, N., Sarela, A., Klingbeil, L., Karunnanithi, M.: Detecting walking activity in cardiac rehabilitation by using accelerometer. In: 3rd International Conference on Intelligent Sensors, Sensor Networks and Information (December 2007)
4. Diemer, R., Kreuzer, J.: Multi-Sensorplattform for Activity Measurements. In: International Conference on Ambulatory Monitoring of Physical Activity and Movements (2008)
5. Ibrahim, R.K., Ambikairajah, E., Cellar, B.G., Lovell, N.H.: Time-frequency based features for classification of walking patterns. In: 15th International Conference on Digital Signal Processing (July 2007)
6. Ravi, N., Dandekar, N., Mysore, P., Littman, M.L.: Activity recognition from accelerometer data. American Association for Artificial Intelligence (2005)