Scheduling for a Modular Activity Recognition System to Reduce Energy Consumption on SmartPhones

Martin Berchtold[1], Henning Günther[1], Matthias Budde[2] and Michael Beigl[2]
[1]Institut für Betriebssysteme und Rechnerverbund,Technische Universität Braunschweig
[2]Institut für Telematik, Chair for Pervasive Computing Systems, Karlsruhe Institute of Technology (KIT)

Abstract

During the last years, mobile phones more and more have come into the focus of interest of activity recognition research. The research community is faced with two system problems: First, how to separate and schedule activity recognition functionality, and second, how to minimize power consumption. In this paper we (a) propose the use of a modular concept with activity driven scheduling between modules to allow a fine grained and thus effective scheduling of activity recognition functionality and (b) present a sleep time scheduling mechanism that inserts sleeping phases to reduce power consumption. The paper presents the implementation and evaluation of the system on a mobile phone.

1 Introduction

Activity recognition has been a relevant research topic for many years now (e.g. [1], [2], [3], [4] and [5]), but with mobile phones providing sensors and sufficient calculation power to compute the recognition on the device itself, the topic becomes relevant for the common user. Many applications can benefit from activity and context information, in order to improve the handling of the device through the user. But when activity recognition is always running in the background to provide information to applications, the issue of shortened battery runtime becomes relevant.
There are two ways of dealing with that problem, one is to reduce the calculation effort for the activity recognition process, the other one is to use the activity information to schedule the recognition process itself. Even though we developed a schedulable modular recognition architecture that reduces the calculation effort significantly ($< 3.3\%$ processor load on average for the recognition of ten activities) without loss of accuracy or reducing the amount of recognized classes, we still experienced a significant reduction of battery runtime with the recognition process running.
Since usually activities do not change very frequently, but the sensors provide data for many classifications per second, a sleep time scheduling algorithm could reduce the power consumption caused by the activity recognition. Also, when the activity recognition is totally switched off for certain periods, the power saving mechanisms native to the phone can be reactivated.
In the field of activity recognition several articles on the topic of scheduling have been published. In [6] the trade-off between sampling rate, accuracy and power consumption was analyzed for the eWatch sensor platform, which is equipped with an accelerometer, a microphone, a light and temperature sensor. The analysis was based on activity recognition of seven different classes. Also, a Markov chain was used to predict the activities and based on that information an exponential backoff sampling strategy was used. The drawback here is, that due to user behavior the Markov chain has to be identified, while our method does not need any prior knowledge or model. In [7] a decision tree classifier is used to select the sensors necessary to determine the system state. This method is not applicable in our case, since we process only one sensor stream in each classification chain and we consider the chain to be a black box, for reasons of flexibility. A dynamic sensor selection to reduce power consumption in activity recognition is also done in [8]. Power and accuracy trade-offs for sound based context recognition are analyzed in [9], where no suggestions for scheduling the recognition are made. All of these published scheduling techniques have in common that they need a special architecture or algorithm for the scheduling to be effective.
The remainder of this paper is structured as follows: First, we introduce a scheduling technique for modular activity classification that can be used for any modular classifier structure. Second, a sleep time scheduling mechanism is explained, which also can be applied to any activity classification method, because it considers the activity classification as a black box and just uses the classification outcome to calculate the sleep time. In this paper, we apply the novel and yet simple scheduling techniques to our modular activity recognition system, which we find has several advantages over other classification mechanisms already.

2 Why Modular Activity Recognition?

For the recognition of activities we use a modular classifier architecture. The modularity has the following benefits over a monolithic architecture:

1. **Schedulable Units:** Modules allow us to schedule between activity recognition functionality and thus optimize and minimize the effort of the system spent for recognition.

2. **Calculation effort:** In [10] it is proved that with a infinite set of rules a FIS can reach infinite accuracy. With non-hypothetical machines, accuracy will always be finite. In [11] it was shown, that the relation between number of rules and accuracy of projection is not linear, but was assumed to be logarithmic. This restricts the implementable recognition functionality on a mobile device with limited processing power and battery capacity. But the users have various demands on what activities and contexts should be recognized, and therefore the amount of classes to be recognized can not be limited to a reasonable calculation effort. It was shown in [11] that a divide and conquer approach can reduce this problem.

3. **Flexibility:** Different users have different demands, behavior and environments. Therefore, personalization on the respective user is necessary to achieve reasonable recognition results. Also, different environments result in different patterns, a fact that also requires an adaption if these patterns change. Also, with a monolithic classifier structure, the search space for any heuristic is much bigger than with a modular structure, where only certain modules can separately be personalized or adapted onto changes [12].

4. **Expandability:** There are certain activities nearly every user performs, while others are specific to a certain group or individual. If new activities need to be recognized with a monolithic classifier structure, the whole classifier needs to be changed. A modular classifier structure offers extensibility [13], without the need for new identification of the whole set of modules.

5. **Meta semantic:** One of the biggest issues when performing activity recognition with mobile devices is, that the position of the device on the user (e.g. *in trousers pocket* or *holding in hand*) or in the environment (e.g. *lying on table* or *lying in car*) is unknown. Using a modular recognition structure, each module can recognize classes, which are specific to a certain context, e.g. a location. If then only one module is active at one point in time, this modules "meta semantic" indicates the location.

These points make a strong argument for using a modular classifier structure for activity recognition on mobile devices. In the following, we explain our modular classifier structure and show how it can be used in a scheduling algorithm to reduce energy consumption.

3 Activity Recognition Module

The process of deriving a class identifier from raw sensor values consists of several steps. The first step is the feature extraction, which calculates the relevant features for classification and is a first dimensionality reduction. Possible feature extraction methods for acceleration sensor measurements are mean and variance values, peak extraction, etc. For audio usually a fourier transformation is used with an additional dimensionality reduction such as mean and variance again, frequency centroid or a principal component projection. For activity recognition we solely use an accelerometer sensor and the features mean and variance, since they are expressive and efficient to calculate.
The next step of processing is a mapping algorithm, that reduces the dimensionality of the features to only one dimension which is classifiable. There are several methods for the mapping possible, such as Neural Networks (NN), Support Vector Machines (SVM), Bayesian Networks or Hidden Markov Models (HMM). For activity recognition in this paper we used a Recurrent Fuzzy Inference System for many reasons, of which the most significant ones are robustness [14] and the support for a reliability measure, that can be used to improve recognition rates.

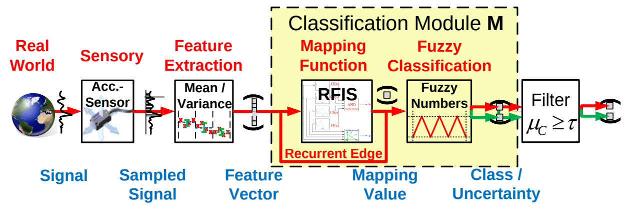

Figure 1: Activity recognition processing queue.

The last step is the classification, in which the one-dimensional outcome value of the mapping is assigned to a discrete class identifier. This assignment is also done fuzzily, so the outcome is not only a class identifier, but also a membership value identifying the reliability of the classification process. This membership value is used in a last step to separate the reliable from the unreliable classifications, thus increasing accuracy of the recognition process.
Step two and three are summarized in a classification module. Instead of just using one classification module, many are used in a dynamic queue.

4 Scheduling of Activity Recognition

We introduce two mechanisms of scheduling, one is the scheduling of the modules in the activity recognition, the other is a sleep time scheduling of the whole recognition process. With the modular structure of our activity classification, a dynamic queue of classification modules is possible, in which only one module is active at each point in time. This reduces the calculational effort significantly. The sleep time scheduling enables the activity recognition process to use the operating systems power savings mechanisms. When a sleep phase is scheduled, the operating system can take over and go into a power saving mode itself.

4.1 Dynamic Queue for Activity Recognition Module Scheduling

Each of the activity classification modules does not only recognize the respective classes, but also a so-called *complementary class*. If this *complementary class* is recognized, another module gets activated. In this manner the *complementary class* indicates whether a classification module can or cannot classify the feature vector input. More details on the *complementary class* can be found in [11].
All classification modules are organized in a dynamic queue. If the active module classifies onto its complementary class, the next module in the queue is activated, and so on. The first module in the queue that can classify the feature vector input (i.e. the first module that classifies on a class different from the complementary one) is then sorted in front of the queue. Only the first module in the queue is active and therefore only a fraction of the overall activity recognizers needs to be calculated. This reduces calculation effort and therefore power consumption. For better understanding of the dynamic queue of classification modules, an example is displayed in figure 2 and described in the following:

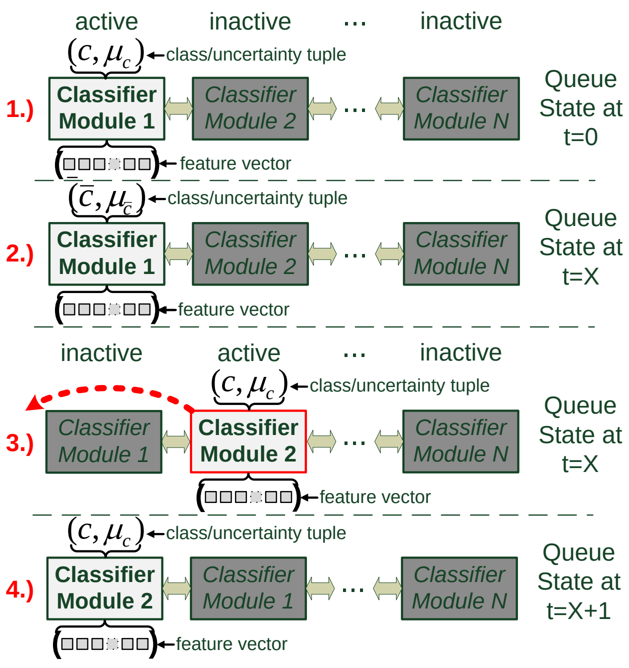

Figure 2: State of modular classifiers in queue at time $t = 0$ and $t = X$

1. The classification modules queue starts at time $t = 0$ (before classification of first feature vector) where the modules are sorted according to their module number. Only the first module in the queue is active and gets to classify the incoming feature vectors.

2. At a time $t = X$ the incoming feature vector gets classified by the first module onto the *complementary class*. This indicates, that this module can not classify this feature vector.

3. The next module in the queue gets now activated and tries to classify the same feature vector. In this example it succeeds and classifies the feature vector on a class different from the complementary one. This module number two now gets put first in queue.

4. In the next time step (new feature vector) the first classifier module is module number two. This module remains being first in queue until it classifies onto the *complementary class*.

Due to misclassification or feature vectors of classes that can not be recognized by any of the modules it is possible, that every module in the queue classifies onto the *complementary class*. In this case, the overall output of the queue is the *complementary class* and the order of the modules in the queue remains the same as before.

4.2 Sleep Time Scheduling Algorithm for Activation of Activity Recognition

With our modular activity recognition architecture, there are two entities possible for input of the sleep time scheduling algorithm. One is the frequency of class changes in successive classifications, the other one is the frequency of module alteration in the dynamic queue. Since our modular recognition architecture is different from the generally used monolithic methods, we use and evaluate the frequency of class changes. This method should therefore be applicable to nearly any activity classification method, since the whole classification process is considered a black box and only the sequence of classification outcomes is used for the sleep time scheduling.
As mentioned before, the sleep time scheduling algorithm is very simple, but effective, since it enables the recognition process to use the devices power savings mechanisms again. With an activity recognition that is always running without any pauses, the power savings mechanisms, such as a *suspend mode*, can never be active.
After each activity classification it is checked if the classification has the same result as the classification before. If the classification result is the same, a counter is increased and a method is called that initiates a sleep phase. This sleep phase is an integer value of the time which is needed for sampling a window size of sensor measurements. Since the calculation time for one classification and feature vector calculation is less than for sampling one window size

of sensor measurements, this method does not result in the missing of sensor values if no sleep phase is scheduled. If the old and the current detected activity class are not the same, the counter is set to zero and therefore no sleep phase is executed.

5 Evaluation

For evaluation we used a OpenMoko Freerunner phone, which is equipped with two 3-D accelerometer sensors, a 400MHz ARM processor (no floating-point unit), a WiFi 802.11 b/g transceiver and a 1200mAh Li-Ion battery. The evaluation includes measurements of the battery capacity and classification results of the activity recognition with and without scheduling. Also, the percentage of detected activity events is measured. An activity event in this manner is the period, where only one activity is performed. The activity event starts when the first classification should recognize a new activity and ends when the last classification of this event is done before a new activity starts. An example of activity events is given in figure 3. To have an upper limit, battery measurements were also taken without running the activity recognition.

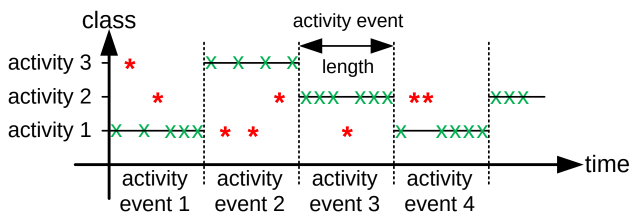

Figure 3: Example of activity events for three activity classes with correct (x) and incorrect ($*$) classifications.

5.1 Evaluation Setting

To have comparable results for an activity recognition with and without sleep time scheduling, a data replay method was implemented. Also, with the capability of relaying a data set, several intervals of activity class changes could be tested. This is necessary for two reasons: (1) the dependability of the sleep time scheduling on the frequency of class changes and (2) the interference of the sleep time scheduling with the recurrence of the classifier. Since the acceleration sensor and its sampling through the controller needs energy too, the replay mechanism intervenes after the sensor measurement and before the feature extraction. Here, instead of the current acceleration measurement the recorded measurement is inserted.

For evaluation we used a data set of $\sim$ 72 minutes for nine different activity classes of one person. The activity classes which were used are typical to mobile phones. The activity classes are grouped together in the different modules according to the location they occur and the amount of movement, where the semantic is called *conditional context*. All the activities, modules and *conditional* contexts are displayed in table 1.

Conditional Context	Context Class	Class No.	Classifier Module
Phone in users trouser pocket: *no movement*	user is sitting user is standing user is lying	1 2 3	$\mathbf{M}_1$
movement	user is walking user is climbing stairs user is cycling	4 5 6	$\mathbf{M}_2$
Phone in users hand:	just holding talking on phone typing text message	7 8 9	$\mathbf{M}_3$

Table 1: Conditional contexts, classes and classifier modules for the acceleration sensor.

Besides the replay method, there are some other issues in the evaluation method. One is, that the distribution of the operation system (Debian GNU/Linux neo 2.6.32v20) currently running on the evaluation device does not allow sensor measurements, calculations or active SSH sessions when the phone is in a power saving mode. This includes the inactive display, too. To make a remote evaluation possible, in which a normal keyboard could be used and the results could be displayed on a normal sized screen, the activity recognizer ran remotely over a SSH session.

Since the sleep time scheduling results in smaller to bigger leaps forward in time, one question to be answered in the evaluation is if activity events could be detected or not. Also, with the filtering on the reliability measure, which improves the overall recognition accuracy, the amount of classifications are reduced. Here, a correctly detected activity event could be filtered out and therefore the threshold determines also how many activity events could be detected. This is the case for both methods of activity recognition, the sleep time scheduled and the one where only the modules are scheduled. To especially evaluate this circumstance, the replayed test data is randomized according to slices of 80, 240 and 400 data pairs of sensor measurements, which leads to 10, 30 and 50 classifications (8 sample window size) of the same activity in a row. Here an integer multiple of the window size for feature extraction was chosen, since no window should include data of two different activities. In mean there are 5.75 classifications per second, which results in activity periods of $\sim$ 1.74, $\sim$ 5.22 and $\sim$ 8.7 seconds.

When measuring the power consumption we have to deal with imprecise information which is provided by the operation system of the device. The OS only provides integer measurements of the percentage of remaining battery power. Here a more precise percentage or remaining mAh would be desirable. Nevertheless, we can show, that the scheduled classification lowers the energy consumption of the activity recognition.

5.2 Activity Recognition without Sleep Time Scheduling

First, the accuracy and energy consumption of the activity recognition without sleep time scheduling and only scheduling of the modules is analyzed. For the percentage of correct classifications the classification without the sleep time scheduling is the upper limit. The sleep time scheduled classification can only be as good as this upper limit. For the measurement of power consumption the activity recognition with modular scheduling is the lower limit, where the recognition where also the sleep time is scheduled only can be better.

Since the sequence size of the activity events has no impact on the energy consumption of the activity classification with only module scheduling, we measured the same remaining battery capacity of 74% for all three trial runs. Compared to the measurements of power consumption (80% battery power remaining) when no activity recognition is done, the module scheduled activity recognition consumes 6*pp* (percentage points) more of the capacity.

The question is now, how many activity events could be detected and what is the accuracy of the recognition? Since the recognition results of the activity recognition for the event period length with 30 classifications in a row is with 78.2% not very high, a filtering of the classifications on the reliability measure is done. But this filter reduces the amount of classifications passed through to the next level of processing or the application. The trade-off here is how many activity events could be detected. For the three trials the percentage of detected events is plotted against the accuracy of classification in figure 4.

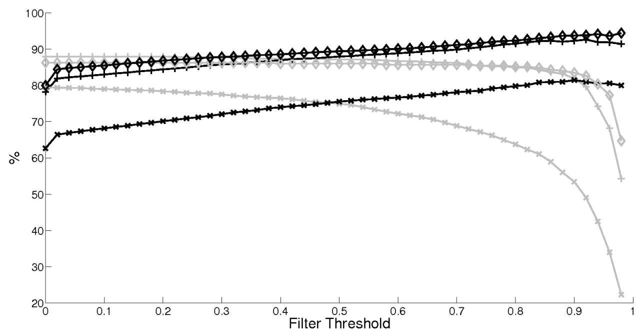

Figure 4: Graphs for percentage of successfully detected activity events (gray lines) and correct classifications (black lines) for different filter thresholds $\tau = 0, .., 0.98$ for activity recognition **without** sleep time scheduling. The different sequence periods for equal activities are 10 (x marker), 30 (+ marker) and 50 ($\diamond$ marker) possible classifications.

An activity event is detected if in a period of data pairs of one activity class at least one classification is the activity that actually happened. As suspected, the higher the filter threshold is, the less activity events could be detected. Here a tradeoff between classification accuracy and percentage of detected events has to be found, which is in our opinion the threshold where the two graphs intersect. But with different event periods, there are different intersection points, so we tend to chose the threshold $\tau = 0.45$ of intersection for the smallest event period of 10 possible successive classifications. This is in our opinion the worst case scenario and for normal human behavior the lower limit, because the human activity periods are normally not smaller than a second.

For this threshold of $\tau = 0.45$ the confusion matrix for an event period length of 30 classifications is shown in table 2. Most of the classes have reasonable recognition rates of close to or above 90%. Other classes could be recognized with over 80% accuracy, which is also an acceptable rate. Only one class (*sitting*, class no.1) has low recognition rates with only 63.5% correct classifications.

	1	2	3	4	5	6	7	8	9
1	63.5	1.9	0.6	2.5	2.0	1.3	10.7	0.4	3.1
2	1.5	82.1	0.6	0.5	1.1	0.2	1.0	1.1	1.4
3	0.0	0.4	92.9	0.1	0.0	0.0	0.0	0.0	0.0
4	0.0	1.3	0.2	91.3	5.4	2.0	0.1	1.3	0.3
5	1.3	0.8	0.5	3.4	89.0	1.5	0.1	0.3	0.7
6	0.0	0.0	0.0	1.0	0.2	93.6	0.0	0.0	0.0
7	32.2	6.4	1.1	1.0	1.6	1.0	86.5	0.3	1.9
8	1.1	5.9	4.0	0.1	0.5	0.3	1.3	96.7	1.2
9	0.3	1.2	0.0	0.0	0.2	0.1	0.2	0.0	91.4
	86.6	85.6	59.2	72.2	79.9	90.3	85.0	96.7	83.4

Table 2: Confusion matrices for activity recognition **without** sleep time scheduling for filter threshold $\tau = 0.45$ with 87.5% overall classification accuracy.

As can be seen, the activity *holding* (class no.7) strongly interferes with the class *sitting* (class no.1), which is due to the very similar position the phone has in the evaluation users pants pocket compared to holding the phone in the hand. This two classes are hardly separable with the available sensor patterns and therefore, the classifier decides for one or the other class. In this case the class *holding* is favored. To have a comparison to the classification results with a higher threshold of $\tau = 0.8$ the confusion matrix is shown in table 3. Here nearly all classes were recognized with over 90% accuracy.

	1	2	3	4	5	6	7	8	9
1	68.0	0.7	0.0	1.9	1.5	0.7	5.9	0.2	3.0
2	1.1	89.3	0.5	0.2	0.9	0.0	0.4	0.6	0.2
3	0.0	0.0	96.8	0.0	0.0	0.0	0.0	0.0	0.0
4	0.0	0.2	0.0	94.7	3.3	0.6	0.2	0.5	0.0
5	0.9	0.7	0.4	1.7	93.5	0.8	0.0	0.1	1.2
6	0.0	0.0	0.0	0.7	0.0	97.3	0.0	0.0	0.0
7	28.9	5.7	0.4	0.5	0.4	0.4	92.3	0.1	1.6
8	0.9	3.0	2.0	0.2	0.4	0.1	1.0	98.5	0.7
9	0.2	0.4	0.0	0.0	0.0	0.0	0.2	0.0	93.3
	51.9	69.1	53.1	31.0	41.1	80.7	45.7	91.9	38.4

Table 3: Confusion matrices for activity recognition **without** sleep time scheduling for filter threshold $\tau = 0.8$ with 91.5% overall classification accuracy.

5.3 Activity Recognition with Sleep Time Scheduling

Due to the structure of the activity classifier, the upper limit of a recognizer with only module scheduling can not be reached through a recognizer with also sleep time scheduling. This is because the modular classification includes a recurrent edge, where the output of the classification at time t is fed back at $t+1$. With this feedback a high recognition rate, a robust classification and the gain of a reliability measure is possible, but the recurrence interferes with the sleep time scheduling. Here the circumstance that with a sleep time scheduled classifier the last classification of time t that is fed back at time $t+n$ is not always the preceding pattern nor activity. This is especially the case, if the classification at t is originating in one activity and the sleep time scheduler activates the recognition at $t+n$ in a different activity.

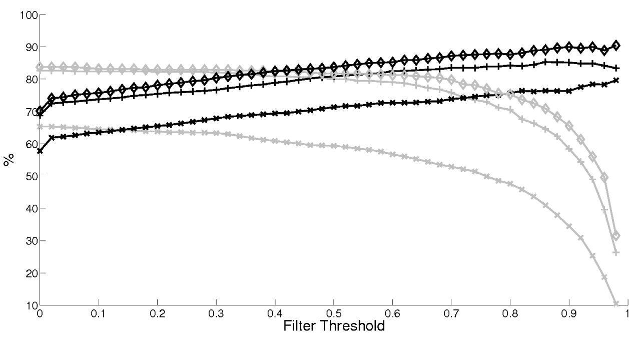

Figure 5: Graphs for percentage of successfully detected activity events (gray lines) and correct classifications (black lines) for different filter thresholds $\tau = 0, .., 0.98$ of sleep time scheduled activity recognition. The different sequence periods for equal activities are 10 (x marker), 30 (+ marker) and 50 ($\diamond$ marker) possible classifications.

The plots of the detected activity events and the accuracy of classification are shown in figure 5, again for different event periods and filter thresholds. As mentioned before, the graphs are generally lower than without sleep time scheduling, but are similar in shape. The worst results are for the smallest event period of 10 successive classifications, where with a maximum sleep period for the sleep time scheduled classification, no two classifications are of the same activity.

The remaining battery capacity is also, for the trial of a period length of 10 successive classifications, with 74% the lowest. This is the same energy consumption as for a unscheduled activity recognition. Here the recurrence stabilizes the recognition mostly at the end of a period, which results in oscillating classifications and therefore nearly no sleep phases. A more precise battery capacity measurement could show marginal improvements.

For the trial of 30 successive classifications periods, the energy consumption of the activity recognition improves by $2pp$ compared to the recognition without sleep time scheduling. The third trial with 50 samples activity event length has the highest remaining capacity of 78% and only $2pp$ more battery consumption than without any activity recognition at all.

Again, for the second trial (20 classifications), the confusion matrices for filter thresholds of $\tau = 0.45$ (left) and $\tau = 0.8$ (right) are shown in table 4 and in table 5.

	1	2	3	4	5	6	7	8	9
1	57.9	1.8	2.2	7.5	6.7	5.6	21.9	0.9	8.4
2	6.7	77.9	0.5	2.8	1.6	0.0	1.9	1.2	1.0
3	0.0	0.3	88.1	0.0	0.0	0.0	0.0	0.0	0.0
4	0.4	3.8	1.1	77.6	12.4	2.1	0.0	1.5	0.0
5	0.8	1.8	2.2	9.3	73.3	1.7	0.4	0.0	0.7
6	0.0	0.0	0.0	0.6	0.6	88.2	0.0	0.0	0.0
7	33.5	6.8	2.2	1.9	3.8	1.4	73.6	0.3	2.7
8	0.8	6.2	3.8	0.3	1.6	1.0	1.9	96.1	1.0
9	0.0	1.5	0.0	0.0	0.0	0.0	0.4	0.0	86.2
	71.3	79.6	51.5	67.2	64.8	79.5	75.5	93.5	75.0

Table 4: Confusion matrices for sleep time scheduled activity recognition for filter threshold $\tau = 0.45$ with 79.9% overall classification accuracy.

	1	2	3	4	5	6	7	8	9
1	54.7	0.4	1.4	7.0	6.7	2.6	10.5	0.9	9.6
2	2.7	88.4	0.7	0.0	1.3	0.0	0.8	0.6	1.8
3	0.0	0.0	91.8	0.0	0.0	0.0	0.0	0.0	0.0
4	0.0	0.8	0.0	86.0	10.7	0.4	0.0	0.3	0.0
5	1.4	0.4	1.4	4.7	74.5	0.9	0.0	0.0	0.0
6	0.0	0.0	0.0	0.8	0.7	94.9	0.0	0.0	0.0
7	39.9	5.6	2.1	0.8	4.0	0.9	85.5	0.0	3.5
8	1.4	3.2	2.7	0.8	2.0	0.4	2.4	98.1	0.9
9	0.0	1.2	0.0	0.0	0.0	0.0	0.8	0.0	84.2
	41.6	58.7	40.7	27.0	30.7	64.8	35.3	89.6	28.8

Table 5: Confusion matrices for sleep time scheduled activity recognition for filter threshold $\tau = 0.8$ with 84.2% overall classification accuracy.

The accuracy results are lower than those of an activity recognition with only module scheduling, but with a filter threshold of $\tau = 0.8$ many classes can be recognized with over 80% accuracy and more.

5.4 Evaluation Summary and Discussion

In the evaluation a direct correlation and therefore a trade-off between classification accuracy and activity event detection percentage could be found. A sleep time scheduling even tightens this problem, since due to the recurrence of the classifier, the recognition rates are lowering, and the scheduling reduces the classification periods to detect the events. Also, the filtering, used to eliminate the unreliable classifications and therefore further improving the recognition accuracy, lowers the amount of detected activity events again. The trade-off for the usage of the sleep time scheduling is therefore the accuracy of activity event detection and recognition against the power consumption of the activity recognition. All evaluation results are shown together in table 6.

		sleep time scheduling			module scheduling			no recog.
seq. size	τ	% det. seq.	% class. acc.	% rem. cap.	% det. seq.	% class. acc.	% rem. cap.	% rem. capacity
10	0	65.3	57.8	74	79.4	62.7	74	80
	0.45	59.8	70.3		75.7	74.6		
	0.8	47.6	75.7		63.8	79.8		
30	0	82.6	68.8	76	88.0	78.2		
	0.45	80.7	79.9		87.4	87.5		
	0.8	70.5	84.2		85.0	91.5		
50	0	83.8	70.1	78	86.3	80.0		
	0.45	82.1	83.1		86.0	89.1		
	0.8	75.4	87.7		85.2	92.4		

Table 6: Comparison of results for module scheduled, module and sleep time scheduled and no activity recognition. Table includes results and measurements for different activity sequence sizes and filter thresholds of percentage of detected activity events, correct detected activities and remaining battery capacity.

Especially the measurements of remaining battery capacity are too coarse grained to make general assumptions about the sleep time scheduling algorithm. It is expected, that even for the smallest event size of 10 successive classifications, the sleep time scheduled activity recognition is lower in energy consumption. Also, the evaluation setting where it is not possible to use the phone's sleep cycles distorts the results. If it would be possible to use the phones standby mode in between the sleep time scheduled activity recognition, the power consumption could significantly lowered compared to an "always on" recognition. At this point we were only able to evaluate the power savings for less CPU usage due to the sleep time scheduling, whereas the lion's share would be the utilization of the suspend modes of the mobile device. This is clearly an aspect to validate in future work.

6 Summary and Future Work

We have shown in this paper, that even very simple scheduling algorithms can reduce the power consumption of activity recognition. First the modular activity recognition scheduling was introduced and its benefits over a monolithic structure. This modular architecture has low computational effort for the recognition of a fairly high amount of classes. With this modular activity recognition scheduling we are able to reduce the power consumption to only 6*pp* (percentage points) more than without recognition. The sleep time scheduling algorithm further reduces the power consumption for activity recognition. The evaluation has shown the trade-offs between accuracy, activity event detection and sleep time scheduling.

In future work, further reduction of power consumption will be analyzed. Especially when the sleep time scheduling can be used to utilize the mobile device's suspend modes, little to no impact at all on the battery lifetime caused by the activity recognition should be possible.

References

[1] A. Schmidt, K. A. Aidoo, and A. Takaluoma, et al., "Advanced interaction in context," in *HUC'99*, ser. LNCS, 1999.

[2] T. Brezmes, J.-L. Gorricho, and J. Cotrina, "Activity recognition from accelerometer data on a mobile phone," in *Proceedings of the IWANN '09*. Springer, 2009, pp. 796–799.

[3] N. Györbíró, A. Fábián, and G. Hományi, "An activity recognition system for mobile phones," *MONET*, 2009.

[4] T. S. Saponas, J. Lester, and J. E. Froehlich, et al., "ilearn on the iphone: Real-time human activity classification on commodity mobile phones," *CSE Technical Report*, 2008.

[5] L. Bao and S. S. Intille, "Activity recognition from user-annotated acceleration data," *PERVASIVE*, 2004.

[6] A. Krause, M. Ihmig, E. Rankin, D. Leong, S. Gupta, D. Siewiorek, A. Smailagic, M. Deisher, and U. Sengupta, "Trading off prediction accuracy and power consumption for context-aware wearable computing," *International Symposium on Wearable Computing (ISWC)*, 2005.

[7] A. Benbasat and J. Paradiso, "A framework for the automated genaration of power-efficient classifiers for embedded sensor nodes," *SenSys'07*, 2007.

[8] P. Zappi, C. Lombriser, T. Stiefmeier, E. Farella, D. Roggen, L. Benini, and G. Troester, "Activity recognition from on-body sensors: Accuracy-power trade-off by dynamic sensor selection," *European Workshop on Sensor Networks (EWSN'08)*, 2008.

[9] M. Staeger, P. Lukowicz, and G. Troester, "Power and accuracy trade-offs in sound-based context recognition systems," *Pervasive and Mobile Computing*, 2007.

[10] L. X. Wang, *Adaptive Fuzzy Systems and Control*. Prentice-Hall, Englewood Cliffs, 1998.

[11] M. Berchtold, T. Riedel, M. Beigl, and C. Decker, "Awarepen - classfication probability and fuzziness in a context aware application," *Ubiquitous Intell. and Comp., LNCS*, 2008.

[12] M. Berchtold, T. Riedel, K. van Laerhoven, and C. Decker, "Gath-geva specification and genetic generalization of tsk fuzzy models," *Sys., Man and Cyb. (SMC08), IEEE*, 2008. [Online]. Available: http://www.ibr.cs.tu-bs.de/users/berch/publications/SMC08.pdf

[13] M. Berchtold, M. Budde, H. Schmidtke, and M. Beigl, “An extensible modular recognition concept that makes activity recognition practical,” *German Art. Int. (KI’10), LNAI*, 2010.

[14] M. Berchtold and M. Beigl, “Increased robustness in context detection and reasoning using uncertainty measures - concept and application,” *Ambient Intell. (AmI’09), LNCS*, 2009.