An Extensible Modular Recognition Concept that Makes Activity Recognition Practical

Martin Berchtold[1], Matthias Budde[2], Hedda Schmidtke[2] and Michael Beigl[2]

[1]Braunschweig: Institute of Operating Systems and Computer Networks (IBR)
[2]Karlsruhe: Institute of Telematics, Pervasive Computing Chair, Karlsruhe Institute of Technology (KIT)

Published: 26.05.2010

http://www.digibib.tu-bs.de/?docid=00033761

An Extensible Modular Recognition Concept that Makes Activity Recognition Practical

Martin Berchtold[1], Matthias Budde[2], Hedda Schmidtke[2] and Michael Beigl[2]

[1] Institute of Operating Systems and Computer Networks (IBR), TU Braunschweig
[2] Institute of Telematics, Pervasive Computing Chair, Karlsruhe Institute of Technology (KIT)

Abstract. In mobile and ubiquitous computing, there is a strong need for supporting different users with different interests, needs, and demands. Activity recognition systems for context aware computing applications usually employ highly optimized off-line learning methods. In such systems, a new classifier can only be added if the whole recognition system is redesigned. For many applications that is not a practical approach. To be open for new users and applications, we propose an extensible recognition system with a modular structure. We will show that such an approach can produce almost the same accuracy compared to a system that has been generally trained (only 2 percentage points lower). Our modular classifier system allows the addition of new classifier modules. These modules use Recurrent Fuzzy Inference Systems (RFIS) as mapping functions, that not only deliver a classification, but also an uncertainty value describing the reliability of the classification. Based on the uncertainty value we are able to boost recognition rates. A genetic algorithm search enables the modular combination.

1 Introduction

While many systems dealing with activity recognition in mobile and ubiquitous computing have emerged over the years, there are still challenges to which no solution has been presented so far. In this work, we try to tackle one of the biggest issues in mobile computing: the modular combination and extension of activity recognition systems. Conventionally, activity recognition is mostly done with a single monolithic classifier. Such classifiers have a large size and their usage is inflexible. Having large complex classifiers is problematic, since in mobile computing the processing power of devices is limited and the battery service life is an important factor. A comparison between monolithic and modular classifiers has been conducted in [3]. Classification systems that require large computing resources are not practical, as often real-time reaction is needed. For example, our prototype activity recognition system using a mobile phone should be able to react on a detected activity in real-time, without disturbing the phone's main functionality.

In activity recognition, inflexibility of classifiers is problematic. A general classifier would need to recognize common daily activities (e.g. 'walking'), but also activities specific for a certain group of persons (e.g. 'dancing') or activities that are rarely carried out (e.g. 'climbing'). Overall, a general recognizer would need to be very complex, covering hundreds of activities while only a few of them are actually required for one person. It is well known that large complex recognizers tend to have lower performance than small, focused classification systems. All this brings us to the need for a dynamic, modular classification system, in which it is possible to add or remove classifier modules on demand. In this paper we propose a modular classifier approach based on Recurrent Fuzzy Inference Systems (RFIS). We give a detailed case study

showing how the system reacts if a new classifier set is added to a pre-existing one. We present that, using a bit masking that is identified through a genetic algorithm, we can improve the progression of old and new modules in a dynamic queue.

In other application fields, the combination of classifiers has been investigated for many years now. In general, two approaches are differentiated: combination on the abstract level and on the measurement level. In abstract level combination, classifiers are computed separately and then logically combined, based on class labels or rankings of classes [13]. In [15], classifiers are combined on the measurement level to recognize handwritten Chinese characters. In their proposal, classifiers that map feature vectors onto measurement vectors are combined. [2] proposes a similar attempt, also for handwritten character recognition. Other publications on the topic of dynamic classifier combination with similar approaches are [10] and [11]. All the aforementioned approaches have in common, that either classifiers or other superimposed selection and activation methods are needed to combine all classifiers. This requires that the whole set of classes is known at design time, while in our system, selection and activation occur inherently on the module level. This means that new classes – and therefore classifier modules – can be added to a pre-existing classifier set and only one classifier is active at each point in time. This considerably reduces resource consumption.

There has been a lot of research in activity recognition over the years. An exemplary work is [12], in which a triaxial accelerometer sensor was used to detect eight activity classes. Several algorithms in various combinations are investigated in four different settings on their classification accuracy. Since the employed sensor is mounted on a fixed position in the test persons pelvic region and the algorithms that are used do not qualify for modular and dynamic usage, the recognition rates are not really comparable. [6] and [9] both do activity recognition for mobile phones, where [6] only uses sensors and resources native to the phone for sensor acquisition and classification. Both approaches do not qualify for modular classification, nor do they recognize a significant amount of activities with high accuracy.

The remainder of this paper is structured as follows: First, the modular classifier is explained, with its RFIS mapping function, the machine learning algorithm to identify the RFIS, and the modularity. Second, we describe how modularity can be enabled via a bit vector masking, which is found through a genetic algorithm. The results are presented in the evaluation section and the paper is concluded in the last section.

2 MODULAR RECURRENT FUZZY CLASSIFICATION

The classification process starts with (1) the feature extraction on the data delivered by the sensors. Then (2) the resulting feature vector is reduced to a one-dimensional value through a mapping function. This value is (3) assigned to a class, based on a set of fuzzy numbers. Steps (2) and (3) are carried out by the modules. As mapping function (step 2) we use a Recurrent Fuzzy Inference System (RFIS) [5]. Due to the used RFIS and the fuzzy numbers-based classification, our modular classifier system provides an uncertainty value, that is sensitive to the respective class that is detected. This solves another problem of activity recognition, namely that patterns are only separable to a certain degree, since the position of the mobile device (e.g. commodity phone) is not always fixed. If an uncertainty measure is present for each classification, a filtering upon this can improve reliability by far. Alternatively, sensors could be fixated on certain positions of the body: the resulting patterns then can be chosen to be activity specific, and thereby they are easily separable. In our case that is not possible, because we

only focus on devices that actually exist and are being used in real life. With an uncertainty measure, the classifications with high reliability can be separated from those with low reliability, thus increasing system accuracy. Furthermore, with FISs a bit masking is possible, which enables each dimension of every rule to be 'activated' or 'deactivated' separately. Thereby, the classifier modules can be optimized to react to each other in a dynamic queue. This is possible without losing the original capabilities of the RFIS.

In this section we first specify the RFIS, followed by an explanation of the machine learning algorithm which identifies the mapping function on training data. Lastly, the overall classifier system is explained and how its modularity works.

2.1 Recurrent Fuzzy Inference System

Takagi, Sugeno and Kang [14] (TSK-) FISs are fuzzy rule-based structures, which are especially suited for automated construction. In TSK-FISs, the consequence of the implication is not a functional membership to a fuzzy set, but a constant or linear function. In our case we use the linear functional consequences f_j, which are weighted with the respective input membership function μ_j in the overall output of the TSK-FIS. This output is later assigned to a tuple of class and uncertainty, which is described in 2.3. The consequence f_j of the rule j is a function of the input vector $\overrightarrow{v}_t$ at time t of the TSK-FIS:

$$\text{IF } \mu_j(\overrightarrow{v}_t) \text{ THEN } f_j(\overrightarrow{v}_t), \text{ with } f_j(\overrightarrow{v}_t) := a_{1j}v_1 + .. + a_{nj}v_n + a_{(n+1)j} \tag{1}$$

Since in activity recognition we deal with highly correlated features, we employ covariant Gaussian membership functions, depending on the covariance matrix Σ_j and the mean vector $\overrightarrow{\mathbf{m}}_j$, which are defined as follows:

$$\mu_j(\overrightarrow{v}_t) := e^{-\frac{1}{2}(\overrightarrow{v}_t - \overrightarrow{\mathbf{m}}_j)\Sigma_j^{-1}(\overrightarrow{v}_t - \overrightarrow{\mathbf{m}}_j)^T} \tag{2}$$

The antecedent membership function $\mu_j(\overrightarrow{v}_t)$ is multiplied with the consequence $f_j(\overrightarrow{v}_t)$, then the sum over all rules j is divided through the sum of all membership functions $\mu_j(\overrightarrow{v}_t)$. The resulting formula for the covariant TSK-FIS is defined as follows:

$$\mathbf{S}(\overrightarrow{v}_t) := \frac{\sum_{j=1}^{m} \mu_j(\overrightarrow{v}_t) f_j(\overrightarrow{v}_t)}{\sum_{j=1}^{m} \mu_j(\overrightarrow{v}_t)} \tag{3}$$

The outcome of the mapping at time t is fed back as input dimension n for the TSK-FIS mapping at $t + 1$. The recurrence not only delivers the desired uncertainty level, but also stabilizes and improves the mapping accuracy. Instead of 'Recurrent TSK-FIS' we use the simpler term RFIS in the remainder of this paper.

2.2 RFIS Identification Algorithm

The RFIS is completely described by the parameters a_{ij} of the linear functional consequence f_j and the mean vector $\overrightarrow{\mathbf{m}}_j$ and the covariance matrix Σ_j of the antecedence membership functions μ_j. These values are identified upon an annotated training feature set via a combination of clustering algorithms, linear regression and genetic algorithm generalization. The five step algorithm is displayed in figure 1 and described in the following.

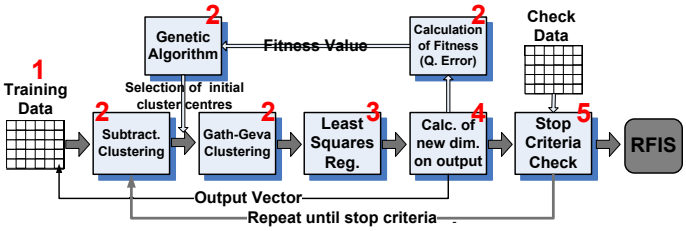

Fig. 1. RFIS identification algorithm.

1. Data Annotation and Separation: The training data $\mathcal{V}^{tr}$ is separated according to the class c_j the data pairs belong to. Clustering on each subset delivers rules that can be assigned to each class. **2. Clustering:** Subtractive clustering [7] gives an upper bound for the amount of clusters. Gath-Geva clustering [8] is performed upon the set of cluster centers determined through the subtractive clustering. Since the number of clusters for the subtractive clustering is higher then for Gath-Geva clustering (multivariant cluster shapes for covariant sensor data), a genetic algorithm searches for the best subset selection of initial cluster centers for the Gath-Geva clustering. The output of the Gath-Geva clustering is the number of rules m, the mean vector $\overrightarrow{\mathbf{m}}_j$ and covariance matrix Σ_j of the membership functions μ_j. **3. Least Squares:** Linear regression identifies the parameters a_{ij} of the linear consequence function f_j of the rules $j = 1, .., m$. Minimizing the quadratic error – the quadratic distance between the desired output and the actual output – leads to an overdetermined linear equation to be solved. **4. Recurrent Data Set:** The output of the TSK-FIS $\mathbf{S}$ is now calculated over the training data $\mathcal{V}^{tr}$. This output is shifted by one, with a leading zero, and then added to the training data set $\mathcal{V}^{tr}$ as additional dimension. All data pairs for time $t > 1$ have the output of the FIS mapping of $t - 1$ in the recurrent dimension n. For this data set the steps 1 to 3 are repeated. **5. Stop Criterion:** There are two values qualifying for a stop criterion: the mean quadratic error or the classification accuracy. While the mean quadratic error mostly improves the expressiveness of the reliability value and less the accuracy, optimizing on the classification accuracy only improves the percentage of correct classifications. For our case study, we decided on a fixed number of iterations.

2.3 Modular Classification

Since the abilities of monolithic classifiers are limited, we use a modular approach to cope with this problem. Instead of using one classifier to classify on all classes $\mathcal{C}$, we use several classifiers $\mathbf{M}_i : \mathcal{V} \rightarrow \mathcal{C}_i$ (with $i = 1, .., N$) each classifying on a small subset $\mathcal{C}_i \subseteq \mathcal{C}$ of classes. The subsets $\mathcal{C}_i$ are chosen according to the classes $c_{ij} \in \mathcal{C}_i$ semantics, therefore each subset $\mathcal{C}_i$ has its own meta semantic. We call this meta semantic 'conditional context'. To not only recognize the respective classes $c_{ij} \in \mathcal{C}_i$, but also the transition between classifiers $\mathbf{M}_i$, each module yields a complementary class $\overline{c}_i$ as well. All classifiers are chained in a dynamic queue, where the last classifier classifying on a class different from the complementary class is put first in queue. The idea behind this re-organization of the classifier queue is, that modules are successive in the queue, which are successive in recognition of input features. Therefore, when the currently 'active' module recognizes the complementary class $\overline{c}_i$, preferably not the whole queue needs to be tested for capabilities of classifying this feature vector $\overrightarrow{v}_t$, but only the next module in the queue.

To train this kind of queued classifiers, we need to train them on the respective classes $c_{ij} \in \mathcal{C}_i$ and on the complementary class $\overline{c}_i$. The training $\mathcal{V}_i^{tr}$ and check data sets $\mathcal{V}_i^{ck}$ for a classifier module $\mathbf{M}_i$ are unified with a selection of input data pairs of all other classifiers $\mathcal{V}_i^{\overline{c}} \subset \bigcup_{k \neq i} \mathcal{V}_k^{tr}$. This selection is labeled zero – which indicates the complementary class $\overline{c}_i$ in every module $\mathbf{M}_i$ – and added to the normal training and check data sets of this classifier. The actual training and check data is therefore $\mathcal{V}_i^{tr \cup \overline{c}}$ and $\mathcal{V}_i^{ck \cup \overline{c}}$, which are called $\mathcal{V}_i^{tr}$ and $\mathcal{V}_i^{ck}$ throughout the rest of this paper for reasons of simplicity.

The output of the RFIS $\mathbf{S}(\overrightarrow{v}_t)$ at time t is the normalized weighted sum of the functions $f_j(\overrightarrow{v}_t)$ of the rules j. The returned values numerically encode the classes. The assignment of the RFIS mapping result $\mathbf{S}_i(\overrightarrow{v}_t)$ to a class is done fuzzily, so the result is not only a class identifier, but also a membership, representing the reliability of the classification process. Each class c_{ij} is interpreted by a set of a triangularly shaped fuzzy numbers [1] (eqn. 4):

$$\mu_{c_{ij}}(x) = \begin{cases} \max(0, 1 - \frac{c_{ij} - x}{\alpha}) \text{ , when } x \leq c_{ij} \\ \max(0, 1 - \frac{x - c_{ij}}{\alpha}) \text{ , when } x > c_{ij} \end{cases} \tag{4}$$

The mean of the fuzzy number is the identifier c_{ij} itself. The crisp decision – i.e. which identifier is the mapping outcome – is carried out based on the highest degree of membership to one of the fuzzy numbers in the class $\mathbf{K}_i$ (eqn. 5).

$$\mathbf{K}_i(x) = \begin{cases} (c_{i1}, \mu_{c_{i1}}(x)) \text{ , when } \mu_{c_{i1}}(x) = m_i(x) \\ \vdots \qquad\qquad\qquad \vdots \\ (\overline{c}_i, \mu_{\overline{c}_i}(x)) \quad \text{ , when } \mu_{\overline{c}_i}(x) = m_i(x) \end{cases} \tag{5}$$

$$m_i(x) = \max(\max_j(\mu_{c_{ij}}(x)), \mu_{\overline{c}_i}(x)) \tag{6}$$

The overall output of the classifier module $\mathbf{M}_i$ (eqn. 7) is a tuple $(c_{ij}, \mu_{c_{ij}})$ of a class identifier and the membership to it, where $c_{ij} \in \mathcal{C}_i \cup \overline{c}_i$ and $\mu_{c_{ij}} \in [0, 1]$.

$$\mathbf{M}_i(\overrightarrow{v}_t) := \mathbf{K}_i(\mathbf{S}_i(\overrightarrow{v}_t)) \tag{7}$$

3 ENABLING MODULARITY

In this paper we focus on the modular combination of many fuzzy classifiers, classifying on subsets $\mathcal{C}_i$ of the overall recognized set $\mathcal{C}$ of classes. Especially, we are analyzing the capabilities of adding new classifiers modules and therefore new classes to a pre-existing classifier module set. This is not possible without an adaption of the pre-existing set of modules, since the modules are aware of each other, so they can work together in a dynamic queue. This awareness is residing in the recognition of the previously described complementary class, which also needs to be trained on. Adaption is done in a way that the original capabilities still remain and can be regained through the use of a simple 'on' switch.

3.1 Generalization on new Modules

When adding a new classifier module to an existing classification system we need to modify the state transition in the dynamic queue. This transition is done when the active module $\mathbf{M}_i$

recognizes the complementary class $\overline{c}_i$ ($\widehat{=}\ 0$). Since adding new modules presumes that no data for identifying the complementary classes of the old modules is available, the old modules are not able to detect a transition onto the new modules. This means that the new modules rarely get activated and therefore have no chance to be part of the classification. The only possibilities of activation are detection errors and feature vectors that get mapped onto the complementary class by all original modules because they are outside the models' mapping space.

3.2 Bit Vector Adaption Mechanism

If new classifier modules are added to the existing queue, the old ones need to be adapted, so that they recognize transitions to the new classifiers. Through an adaption technique, in which the dimensions of each rule for each pre-existing classifier module are 'activated' or 'deactivated', transition capabilities improve without changing the classification accuracy. The original classifier module is preserved and can be restored at any time. The optimal combinations of 'active/inactive' rule dimensions is searched for with a genetic algorithm, since the full search has an exponential runtime. The data for optimum search is the training data that the new modules were identified on, combined with the original training data.

The adaption is done via a bit vector, that specifies the 'active' and 'inactive' dimensions of each rule for one module $\mathbf{M}_i$. Therefore the bit vector $bit_{\mathbf{M}_i}$ for module $\mathbf{M}_i$, which has n input dimensions and m_i rules, has a length of $b_i := n \cdot m_i$. To use the bit vector, an interpretation function $I(\mathbf{M}_i(\overrightarrow{v}_t), bit_{\mathbf{M}_i})$ is defined, that 'switches' the rules dimensions temporarily, without changing the module $\mathbf{M}_i$ permanently. The interpretation function I is defined as a function mapping a module $\mathbf{M} \in \mathbb{F}(\mathbb{R}^n, \mathbb{R})$ together with a bit vector $bit_{\mathbf{M}} \in \{0,1\}^*$ of appropriate length to a module $\mathbf{M}'$:

$$I : \mathbb{F}(\mathbb{R}^n, \mathbb{R}) \times \{0,1\}^* \longrightarrow \mathbb{F}(\mathbb{R}^n, \mathbb{R}) \tag{8}$$

More details on this bit vector approach can be found in [4].

3.3 Genetic Algorithm Search Space

To determine the respective classifier module's bit vector, a genetic algorithm is used. The space that has to be searched is 2^{b_i} for module $\mathbf{M}_i$. A complete search would therefore have a runtime of $O(2^{b_i})$, which makes it impossible to calculate in a reasonable amount of time. In our experience, the genetic algorithm can find a suboptimal, but appropriate solution in a time span that is acceptable for our application. Nevertheless, we are currently investigating methods to limit the search space.

4 EVALUATION

For evaluation we use a typical mobile computing device, the commodity phone. Nowadays commodity phones come with a variety of sensors, such a microphone, proximity sensor, GPS and accelerometers. We focus on accelerometer sensors, since with these sensors we can recognize most types of activity events. The phone is required to be with the user, e.g. in his pocket or held in his hand. With our modular classifier approach, we not only can recognize activity classes, but also the place the phone is positioned on. We call this 'conditional context' and get this directly through the respective activated classifier module. We need this conditional context to reach high recognition rates, as recognized acceleration patterns from the same activity differ heavily with the conditional context.

4.1 Application

The device chosen by us is the 'OpenMoko Freerunner' phone which comes with two 3-D accelerometer sensors. The sensors have a sampling rate of about 100Hz. With a window size of 8 samples we end up with 12.5 feature vectors per second. The features are mean and variance, since they are efficiently calculable and enable good recognition results. With this feature extraction, the resulting feature vector has 12 dimensions. Adding the recursive dimension makes a total of 13 dimensions. This feature vector is mapped onto the respective activity class via the currently 'active' module. If the active module classifies onto the complementary class, the next module is activated for this feature vector. This procedure is repeated for this feature vector until one of the modules classifies onto a class different from the complementary one.[1] This classifier module is then put first in queue and the next feature vector is processed. To increase reliability – and thereby accuracy – filtering upon the uncertainty is done. The filtering reduces the amount of output classes, but for most applications a few classes per second are more than enough, where the reliability is most important. Here we assume that most activities are not changing faster than in seconds.

The original classifier module queue classifies onto ten classes and recognizes four conditional contexts. To evaluate the addition of new classifiers, two more modules are put in queue. These recognize five classes and two conditional contexts. All activity classes, conditional contexts and respective classifier modules are shown in table 1.

Conditional Context	Context Class	Class No.	Classifier Module
Phone in users trouser pocket: *no movement*	user is sitting	1	$\mathbf{M}_1$
	user is standing	2	
	user is lying	3	
Phone in users trouser pocket: *movement*	user is walking	4	$\mathbf{M}_2$
	user is climbing stairs	5	
	user is cycling	6	
Phone on table:	no movement	7	$\mathbf{M}_3$
Phone in users hand:	just holding	8	$\mathbf{M}_4$
	talking on phone	9	
	typing text message	10	
Phone in users trouser pocket:	user is sitting in bus	11	$\mathbf{M}_5$
	user is standing in bus	12	
Phone in users trouser pocket: *dancing*	user is dancing (style 1)	13	$\mathbf{M}_6$
	user is dancing (style 2)	14	
	user is dancing (style 3)	15	

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

The training data $\mathcal{V}_i^{tr}$ for classifier module $\mathbf{M}_i$ consisted of 400 data pairs and the check data $\mathcal{V}_i^{ck}$ of 200 pairs for each class c_{ij}. The data for training the complementary class $\overline{c}_i$ consisted of 800 and for the check data of 400 pairs, which was randomly selected out of the training and check data for the other classifiers. Both sets $\mathcal{V}_i^{tr}$ and $\mathcal{V}_i^{ck}$ were randomized in slices of 30 pairs, so the recurrence could be trained and tested. Each classifier modules' $\mathbf{M}_i$ RFIS mapping function $\mathbf{S}_i$ was trained for 100 epochs, after which the RFIS achieving the best classification accuracy for combination of training and check data was chosen.

The evaluation data $\mathcal{V}^{ev}$ had about 2000 data pairs per class, so 160 seconds per activity. This evaluation set was also randomized according to slices of 20 data pairs ($1,67$ seconds), which in our experience is the quickest transition occurring between activities. Since most of the false classifications are occurring when changing from one activity class to another (due to recurrence), this is a stress test for the classifier modules. In a real application we estimate even better results than presented in the following, because activities change less often.

[1] in case all classifiers including the last one classify onto $\overline{c_i}$, the overall output is the complementary class. This means the feature vector cannot be classified correctly by the given queue.

4.2 Evaluation Results

First we have a closer look into the results of classification, when all classifier modules get trained on a collective data set. When no filtering upon the uncertainty value is done, the overall recognition rate lies at only 58%. This is too low for practical use, but with our recurrent fuzzy classifier approach, we can improve the results significantly. As mentioned before, activity data from sources without fixed position is hardly separable. For this a filtering is needed, where the classifications with low reliability are separated from the recognitions with high reliability.

With the filtering upon the uncertainty value with a threshold of $\tau = 0.96$, we can boost the overall recognition rate by 27.4 *pp* (*percentage points*) up to 85,4%. The corresponding confusion matrix is shown in table 2. This is a recognition rate for a system which can be

	91%										**75%**				
	$\mathbf{M}_1$			$\mathbf{M}_2$			$\mathbf{M}_3$	$\mathbf{M}_4$			$\mathbf{M}_5$		$\mathbf{M}_6$		
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	99.7	0.0	0.4	0.1	1.5	2.2	0.4	0.0	0.0	1.1	0.4	5.1	0.0	1.5	5.2
2	0.0	90.9	0.1	0.0	0.0	2.1	0.0	0.0	0.0	0.3	0.3	0.0	0.4	0.7	0.7
3	0.0	0.0	98.9	0.0	0.0	5.1	0.0	0.0	0.1	0.0	0.0	0.0	0.0	0.0	3.0
4	0.0	0.1	0.0	92.6	1.5	0.6	0.0	0.0	0.0	0.3	0.0	0.2	6.1	2.2	0.7
5	0.0	0.1	0.0	1.0	49.5	0.1	0.0	0.0	0.0	0.0	0.0	0.0	17.0	0.7	0.0
6	0.0	0.0	0.1	0.0	0.0	88.1	0.0	0.0	0.0	0.0	0.0	0.0	0.4	0.0	6.7
7	0.0	0.0	0.0	0.1	1.5	0.0	98.2	0.0	1.3	3.4	0.0	0.0	0.0	2.2	7.5
8	0.2	0.2	0.0	0.0	0.5	0.3	0.0	98.3	0.1	0.0	0.0	0.0	0.4	2.9	3.7
9	0.0	0.0	0.0	0.1	0.0	0.1	0.2	0.6	98.3	0.3	0.0	0.0	0.0	2.2	0.0
10	0.0	0.0	0.0	0.0	0.5	0.0	0.5	0.0	0.0	91.3	0.0	0.0	0.0	0.0	0.0
11	0.0	0.0	0.0	0.0	1.0	0.1	0.0	0.0	0.1	0.0	99.1	0.0	0.4	0.7	0.7
12	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	86.9	0.0	5.8	9.7
13	0.1	8.8	0.4	5.9	43.9	0.4	0.0	0.0	0.1	1.3	0.1	0.2	75.1	0.7	0.0
14	0.0	0.0	0.0	0.1	0.0	0.6	0.6	1.0	0.0	2.1	0.1	1.7	0.4	67.2	14.9
15	0.0	0.0	0.0	0.1	0.0	0.1	0.1	0.0	0.0	0.0	0.0	5.9	0.0	13.1	47.0
	69.3	60.3	73.7	15.1	6.2	26.5	56.8	42.1	50.3	27.7	40.8	25.0	9.2	4.4	4.6

Table 2. Theoretical optimum without knowledge of all training data. Filter threshold: $\tau = 0.96$. Overall recognition rate: 85,4%.

used in real applications. But the filtering reduces the amount of classifications the recognition system has as output. The percentage of remaining classifications for each class after filtering is displayed in each table in the last row of the confusion matrices. Since the classifier modules originally produce 12.5 classifications per second, a percentage of less than 8% remaining classifications could result in an effective output of less than one class per second. Usually, a person's activities do not change that fast, but it is still possible that the low effective polling rate could result in missing an activity event.

Taking a closer look, we can see that most of the classes (9 out of 15) are recognized with an accuracy of over 90%, as required for activity recognition applications. The classes with lower recognition percentages could have overlapping membership functions in the RFIS mapping, which is indicated through mutually misclassified classes. A good example for this circumstance is given by classes no. 5 and no. 13, where $43,9\%$ of data for classifier no. 5 is misclassified on no. 13 and 17% from no. 13 on no. 5. These two provide a good example for classes whose patterns are hardly separable. The overall recognition rates in our evaluation are significantly lowered due to the fact that we have classes with overlapping membership functions in our system.

Next, we examine the classification accuracies when a new set of classifiers is added to a pre-existing one. The original classifier set consisted of modules $\mathbf{M}_1$ to $\mathbf{M}_4$. To this, the set containing modules $\mathbf{M}_5$ and $\mathbf{M}_6$ is added. The results of the union without bit masking the

	95%										17%				
	$\mathbf{M}_1$			$\mathbf{M}_2$			$\mathbf{M}_3$	$\mathbf{M}_4$			$\mathbf{M}_5$		$\mathbf{M}_6$		
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	99.3	0.2	1.7	0.4	0.8	2.8	0.5	0.2	0.2	4.5	8.6	0.0	0.8	7.7	2.3
2	0.0	98.9	0.9	0.4	0.1	0.6	0.0	0.1	0.2	0.7	73.8	1.2	0.3	16.9	6.8
3	0.0	0.0	92.9	0.0	0.0	0.2	0.0	0.1	0.0	0.0	0.0	26.0	0.0	26.2	34.1
4	0.0	0.0	0.0	86.4	2.1	0.0	0.1	0.0	0.1	0.0	0.0	0.0	4.1	1.5	0.0
5	0.0	0.3	1.9	12.9	95.4	0.2	0.2	0.1	0.3	1.5	0.0	0.6	92.2	20.0	18.2
6	0.0	0.0	1.7	0.0	0.6	95.4	0.0	0.0	0.0	0.0	0.8	0.0	0.2	6.2	6.8
7	0.2	0.0	0.0	0.0	0.0	0.0	99.0	0.0	0.0	11.2	0.4	0.0	0.0	0.0	0.0
8	0.3	0.2	0.6	0.0	0.4	0.7	0.1	99.4	0.0	2.2	0.0	15.0	0.9	13.8	20.5
9	0.0	0.0	0.2	0.0	0.0	0.0	0.2	0.1	99.2	0.0	0.0	1.2	0.3	0.0	2.3
10	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	79.9	0.0	0.0	0.2	0.0	0.0
11	0.0	0.0	0.0	0.0	0.0	0.2	0.0	0.0	0.0	0.0	16.4	0.0	0.0	3.1	0.0
12	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	55.5	0.0	1.5	2.3
13	0.0	0.4	0.0	0.0	0.6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.2	0.0	0.0
14	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.6	0.0	3.1	0.0
15	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	6.8
	51.0	75.7	14.3	29.4	26.9	21.4	59.1	79.3	67.5	9.8	7.5	3.8	22.0	2.1	1.5

Table 3. Modular approach without the need for knowledge of all training data in all training steps that is not masked with a genetically identified bit vector. Threshold: $\tau = 0.96$. Overall recognition rate: 68,6%.

classifiers are shown in table 3. Here, the recognition rate for the original modules is 95% as required. The added modules $\mathbf{M}_5$ and $\mathbf{M}_6$ are rarely activated, because the only case where they could be activated is when misclassifications occur and all original modules recognize the complementary class. After the genetic algorithm has found a bit masking for the original

	89%										73%				
	$\mathbf{M}'_1$			$\mathbf{M}'_2$			$\mathbf{M}'_3$	$\mathbf{M}'_4$			$\mathbf{M}_5$		$\mathbf{M}_6$		
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	99.6	0.1	0.3	0.2	1.4	3.1	0.3	0.0	0.0	0.8	0.3	5.1	0.0	1.5	5.8
2	0.0	92.1	0.1	0.0	0.0	2.7	0.0	0.0	0.0	0.0	0.2	0.1	0.4	0.8	2.2
3	0.0	0.0	96.1	0.0	0.0	5.9	0.0	0.0	0.2	0.0	0.0	0.0	0.4	0.0	2.9
4	0.0	0.2	0.2	87.2	7.7	0.3	0.0	0.1	0.0	0.3	0.1	0.3	12.4	3.8	1.4
5	0.0	0.1	0.1	1.6	46.4	0.2	0.0	0.0	0.0	0.0	0.0	0.0	16.3	0.0	0.7
6	0.2	0.0	2.7	0.0	0.0	84.9	0.0	0.0	0.0	0.0	0.0	0.0	0.8	0.0	5.8
7	0.0	0.1	0.0	0.2	1.9	0.0	98.1	0.0	2.1	1.4	0.0	0.0	0.0	2.3	7.2
8	0.1	0.2	0.0	0.0	0.5	0.3	0.0	98.4	0.1	0.0	0.0	0.0	0.4	3.1	3.6
9	0.0	0.0	0.0	0.2	0.0	0.2	0.2	0.6	97.3	0.5	0.0	0.0	0.0	2.3	0.0
10	0.0	0.0	0.0	0.0	0.0	0.0	0.6	0.0	0.0	94.8	0.0	0.0	0.0	0.0	0.0
11	0.0	0.0	0.0	0.0	1.0	0.2	0.0	0.0	0.1	0.0	99.1	0.0	0.0	0.8	0.7
12	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	86.4	0.0	6.9	8.7
13	0.1	7.4	0.5	10.0	41.1	0.6	0.0	0.0	0.1	0.8	0.2	0.2	69.0	0.0	0.0
14	0.0	0.0	0.0	0.2	0.0	1.0	0.8	0.9	0.0	1.4	0.1	1.5	0.4	64.1	13.8
15	0.0	0.0	0.0	0.2	0.0	0.6	0.1	0.0	0.0	0.0	0.0	6.5	0.0	14.5	47.1
	69.0	59.9	65.7	7.9	6.6	24.6	57.9	43.2	47.5	26.8	42.4	24.6	8.6	4.2	4.8

Table 4. Modular combination without the need for knowledge of all training data in all training steps, masked with a bit vector. Threshold: $\tau = 0.96$. Overall recog. rate: 84,0% (compared to optimal 85%).

classifiers, the recognition rates increase significantly (table 4): the overall rate for all classes is just 2 *pp* lower compared to the upper limit we achieved when training all modules together. The still high recognition rates for the ten original classes indicate, that the original classification capabilities of these modules have not changed much. A recognition rate of 6 *pp* less compared to the non bit masked classifier modules and an increase to 15 recognized classes is a really good result compared to other activity recognition systems.

5 CONCLUSIONS

We have presented an approach for modular classification in activity recognition feasible for mobile devices, where Recurrent Fuzzy Inference Systems (RFIS) are utilized. This approach solves two main issues in practical activity recognition. One is the addition of new classifier modules to a pre-existing set of modules. We solve this problem with a bit vector masking of the classifier modules, so that the new set of modules are recognized in the dynamic queue of classifiers. The other issue is, that activity classes are often hardly separable, especially when the sensor data sources have no fixed position. This problem is solved through the used RFIS in combination with a filtering. The filtering separates reliable from unreliable classifications according to the uncertainty value provided by the RFIS mapping in the classifier modules.

In a case study, we have shown, that the modular combination of new and pre-existing classifiers has nearly the same recognition rates (only 2 *pp* less accurate), compared to a classification system, where all modules get trained together. Also the filtering on the uncertainty value achieves a boost of recognition up to 95%, which in the end makes activity recognition practical.

References

1. AbuAarqob, O.A., Shawagfeh, N.T., AbuGhneim, O.A.: Functions defined on fuzzy real numbers according to zadehs extension. International Mathematical Forum 3(16) (2008)
2. Aksela, M., Laaksonen, J.: Adaptive combination of adaptive classifiers for handwritten character recognition. Pattern Recogn. Lett. 28(1), 136–143 (2007)
3. Berchtold, M., Riedel, T., Beigl, M., Decker, C.: Awarepen - classfication probability and fuzziness in a context aware application. Ubiquitous Intelligence and Computing (2008)
4. Berchtold, M., Riedel, T., van Laerhoven, K., Decker, C.: Gath-geva specification and genetic generalization of takagi-sugeno-kang fuzzy models. Proceedings of the SMC08 (2008)
5. Berchtold, M., Beigl, M.: Increased robustness in context detection and reasoning using uncertainty measures: Concept and application. In: Proceedings of the AmI '09 (2009)
6. Brezmes, T., Gorricho, J.L., Cotrina, J.: Activity recognition from accelerometer data on a mobile phone. In: Proceedings of the IWANN '09. pp. 796–799 (2009)
7. Chiu, S.: Method and software for extracting fuzzy classification rules by subtractive clustering. IEEE Control Systems Magazine, 1996, vol. pp. 461-465 (1996)
8. Gath, I., Geva, A.B.: Unsupervised optimal fuzzy clustering. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol 11(7), pp 773-781 (1989)
9. Györbíró, N., Fábián, A., Hományi, G.: An activity recognition system for mobile phones. Mobile Networks and Applications 14(1), 82–91 (2009)
10. Ianakiev, K.G., Govindaraju, V.: Architecture for classifier combination using entropy measures. In: Proceedings of the MCS '00 (2000)
11. Kirchhoff, K., Bilmes, J.A.: Dynamic classifier combination in hybrid speech recognition systems using utterance-level confidence values. In: ICASSP '99. pp. 693–696 (1999)
12. Ravi, N., D, N., Mysore, P., Littman, M.L.: Activity recognition from accelerometer data. In: In Proceedings of the 17th IAAI. pp. 1541–1546 (2005)
13. Singh, S., Singh, M.: A dynamic classifier selection and combination approach to image region labelling. Signal Processing: Image Communication 20(3) (2005)
14. Tagaki, T., Sugeno, M.: Fuzzy identification of systems and its application to modelling and control. Systems, Man and Cybernetics (1985)
15. Xiao, B., Wang, C., Dai, R.: Adaptive combination of classifiers and its application to handwritten chinese character recognition. Int. Conference on Pattern Recognition (2000)