Adaptive Audio-Based Context Recognition

Waltenegus Dargie, *Member, IEEE*

***Abstract*—Context recognition is an essential aspect of intelligent systems and environments. In most cases, the recognition of a context of interest cannot be achieved in a single step. Between measuring a physical phenomenon and the estimation or recognition of what this phenomenon represents, there are several intermediate stages which require a significant computation. Understanding the resource requirements of these steps is vital to determine the feasibility of context recognition on a given device. In this paper, we propose an adaptive context-recognition architecture that accommodates uncertain knowledge to deal with sensed data. The architecture consists of an adaptation component that monitors the capability and workload of a device and dynamically adapts recognition accuracy and processing time. The architecture is implemented for an audio-based context recognition. A detail account of the tradeoff between recognition time and recognition accuracy is provided.**

***Index Terms*—Audio-signal processing, context awareness, context reasoning, context recognition, context-recognition accuracy, context-recognition time.**

I. Introduction

CONTEXT-AWARENESS is an essential aspect of intelligent computing systems. It deals with the collection of raw data from various sensors which are placed in various places and devices and the processing of these data to extract meaningful higher level activities and social situations (higher level contexts). The successful recognition or estimation of higher level contexts enables intelligent computing systems to perform the following functions: 1) seamlessly adapt to a perceived change; 2) augment human perception; and 3) provide relevant services in a proactive manner.

Several smart systems and collaborative environments have been proposed in the recent past. For example, the iBadge [1] wearable system monitors the social and individual activities of children in a nursery school. It incorporates sensing, processing, communication, and actuating units. The sensing unit includes a magnetic sensor, a dual-axis accelerometer, a temperature sensor, a humidity sensor, a pressure sensor, and a light sensor. It includes also an ultrasound transceiver and an RF transceiver for position and distance estimations. The processing unit includes speech and sensor data processing. A server side application assists a teacher by receiving and processing location, orientation, ambient, and audio contexts from the iBadge to determine the social and learning status of a child. The location and orientation contexts are used to determine whether a child is isolated or associates with other children. The audio context is used to determine whether a child is sociable or aggressive.

Dargie and Tersch [2] use acoustic signals and audio digital-signal processing to determine more that 20 different activities in a university campus. The context-recognition process involves modeling frequency-domain audio features and building a Bayesian network. Similarly, Dargie and Hammann [3] use Bayesian networks to estimate the whereabouts of a mobile user. The Bayesian network models stochastic features of data taken from various sensors (humidity, temperature, and light).

Likewise, Magee *et al.* [4] introduce the nonintrusive communication-interface system called EyeKeys. It runs on an ordinary computer with a video input from an inexpensive Universal Serial Bus camera and works without special lighting. EyeKeys detects and tracks the person's face using multiscale template correlation. The symmetry between left and right eyes is exploited to detect if the person is looking at the camera or to the left or to the right side. The detected eye direction is used to control applications such as spelling programs or games.

Several system architectures have been proposed to develop intelligent systems and environments. The architecture of Dargie and Tersch [2] consists of raw-data extraction, atomic-feature extraction, atomic-scene recognition, and context recognition. The architecture employs a knowledge base to model various everyday human activities. The iBadge system discussed earlier is built with the Sylph architecture [1], which consists of sensor modules, a proxy core, and a service-discovery module. Wang *et al.* [5] propose the semantic space framework that consists of context wrappers, a knowledge base, aggregators, a context-query engine, and a context reasoner. Chen *et al.* [6] propose the CoBra middleware, which consists of a knowledge base, a context-reasoning engine, a context-acquisition module, and a policy-management module. Similarly, Korpipää *et al.* [7] propose a distributed architecture for context recognition and management. The architecture consists of a context manager, a resource server, a context-recognition service, a change-detection service, and a security service. The context manager shields applications from the concern of context acquisition by functioning as a central server. All the other services post their output to it.

There is a remarkable similarity between the proposed architectures. First of all, they all support the separation of context acquisition from context usage. This is done by providing context widgets, context wrappers, context-acquisition modules, and context-query engines. Second, they support the presentation of a context at various abstraction levels—this is the task of interpretors, aggregators, and reasoning engines. Third, they support the dynamic binding of context sources by introducing service-discovery mechanisms.

While these architectures identify the conceptual steps of a context recognition, they are, however, higher level, in that

Manuscript received January 8, 2008; revised May 8, 2008 and October 1, 2008. First published April 24, 2009; current version published June 19, 2009. This paper was recommended by Associate Editor Y. Lin.

The author is with the Technical University of Dresden, 01062 Dresden, Germany (e-mail: waltenegus.dargie@tu-dresden.de).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TSMCA.2009.2015676

they rarely address the associated system complexity. System complexity is rather a crucial issue for several practical reasons. For example, the scope and usefulness of a context of interest depends on its timeliness and accuracy—both metrics being application-specific. Furthermore, the workload of the intelligent system, which is a dynamic aspect, influences both metrics, and itself depends on the available resources (such as the energy reserve, CPU speed, communication bandwidth, active memory, and storage). Variable system resources, in turn, directly affect the way sensed data can be obtained, processed, and communicated. Subsequently, a context-recognition architecture should take these dynamic aspects into account.

We provide an adaptive context-recognition architecture which has two essential aspects.

1) It takes the workload of a device into account and adapts the accuracy and duration of a context-recognition process.
2) It takes the capability of a device into account to select a suitable complexity class. This enables us to recognize a context of interest on heterogeneous devices.

The implementation of the architecture will be demonstrated for auditory-based context recognition. In earlier reports [3], [8], a part of the architecture (i.e., without the adaptation component) was implemented to recognize the whereabouts of a mobile user (rooms, corridors, or outdoor). The raw data were obtained from temperature, humidity, and light sensors.

The rest of this paper is organized as follows. In Section II, we present related work. In Section III, we present the conceptual architecture for computing context as an abstraction of real-world settings. In Section IV, a scenario is given for which the architecture is implemented. In Section V, we report the implementation details of the conceptual architecture. In Section VI, we provide a detailed account of the adaptation aspect of our architecture. In Section VII, we discuss our experience and provide comparisons of our result with previous results and close this paper with concluding remarks.

II. Related Work

A. Adaptive Architecture

As far as adaptive context recognition is concerned, to the best of our knowledge, previous contribution is limited in this area. The adaptive context-recognition approach of Nam *et al.* [9] and Young *et al.* [10] focus on filter fusion to support a robust face recognition under uneven illumination (image processing). The system's working environment is learned, and the environmental context is identified (bright, normal, or poor illumination). Based on the initial context, a group of classifiers that are most likely to produce accurate output is generated for each environmental context. A combination of the results of multiple classifiers is determined using a t-test decision model.

Laasonen *et al.* [11] propose an adaptive framework for identifying the whereabouts of a mobile user from cellular-network data. Adaptation is defined as a dynamic thread-off between accuracy and resource consumption. The authors define three concepts (bases, areas, and routes) on the basis of which the complexity of a context-recognition process is estimated. Past and present locations, as well as mobility information, are used to reduce the resource consumption of a context-recognition task.

Stäger *et al.* [12] provide an empirical design process for audio-based context recognition. The process is a result of examining the tradeoff between power consumption and context-recognition accuracy. Given a hardware and its nominal power-consumption profile, the process tunes audio parameters (sampling rate, frame size, size of feature vector, etc.) to satisfy the power-consumption constraint. Based on this design guideline, they developed a wearable context-aware system that recognizes the activity of a user in a kitchen. The sources of audio data are a microwave, a coffee maker, a hot-water nozzle, a coffee grinder, and a water tap.

B. Acoustic Context Recognition

Most existing or proposed auditory-based context-recognition schemes focus mainly on computational aspects, namely, on the accuracy, processing time, and power consumption of a context-recognition task.

Even though auditory-based context-recognition shares several similarities with speech recognition, there are also notable differences. For example, in speech recognition, knowledge of human perception (tone, pitch, loudness, etc.) is useful to disambiguate an uttered speech. This is possible because of the following conditions: 1) the speaker is not far from the microphone and speaks sufficiently clearly and loudly, and 2) there is no significant hindrance between the speaker and the microphone. This is not the case in auditory-based context recognition.

To begin with, the amplitude of the audio signal representing a user's surrounding is not appreciably large, since the audio sources are usually far away from the user (microphone). Second, the device in which the microphone is embedded can be hidden inside a suitcase or a pocket. Third, whereas the frequency of interest in speech recognition is well below 4 kHz, the signal collected from a user's surrounding may incorporate frequencies that are up to and above 10 kHz. All these facts should therefore be taken into account when designing an auditory-based recognition system.

Eronen *et al.* [13] identify time- and frequency-domain features, as well as stochastic features, to classify various everyday outdoor and indoor scenes (streets, restaurants, offices, homes, cars). They report that, by using Mel-frequency cepstral coefficients (MFCCs) and hidden Markov Models (HMMs), they were able to achieve a recognition accuracy of up to 88%. The recognition accuracy as a function of the test-sequence length appears to converge after about 30–60 s. Interestingly, they report that human's recognition accuracy of the same data set was 82% with an average reaction time of 14 s.

Korpipää *et al.* [14] employ a naive Bayesian classifier and an extensive set of audio features derived partly from the algorithms of the MPEG-7 standard. The classification is based mainly on audio features measured in a home scenario. With a resolution of 1 s in segments of 5–30 s and using leave-one-out cross validation, they achieve a recognition rate of 87% of true positives and 95% of true negatives. The result is averaged over nine 8-min scenarios containing 17 segments of different lengths and nine different contexts. The reference accuracies measured by testing training data are 88% (true positive) and 95% (true negative), suggesting that the model is capable of covering the variability introduced in the data on purpose.

Reference recognition accuracy in controlled conditions is 96% and 100%, respectively.

Ma *et al.* [15] also employ HMMs on MFCCs to recognize ten auditory scenes. By varying the hidden states of the HMMs, they achieve different recognition rates. For example, with just three states, the classifier achieves a context-recognition accuracy of 78%, while with 15 states, the recognition accuracy reaches 91.5%. Remarkably, the authors observe a decline in context recognition for higher hidden states. Smith *et al.* [16] extend the work of Ma *et al.* [15] by introducing a belief-revision mechanism to improve the recognition rate (92.27%) and to increase the number of contexts that can be recognized (namely, 12).

C. Summary of Related Work

This paper is more similar to the work of Eronen *et al.*, Korpipää *et al.*, and Stäger *et al.* There are, however, significant differences. First, while they offer no reusable and extensible system architecture, we provide one that can be used beyond audio-based context recognition. Second, even though they extensively investigate the impact of spectral parameters on recognition accuracy, they do not exploit knowledge of the dynamic workload of a device to support adaptation (Stäger *et al.*, for instance, demonstrate the influence of power consumption on context recognition, but runtime power-aware context recognition is not supported). Third, they do not take recognition time into account, which is rather a vital aspect, as the relevance of a piece of context is determined by its timeliness.

This paper complements these approaches by providing an adaptation component. Depending on the capability and workload of a device it performs the following tasks: 1) determines how much resource should be made available for a context-recognition task, and 2) informs the user about the expected processing time and accuracy.

III. Architecture

Recognition of the social and conceptual settings in which computing devices operate cannot be captured in a single step. Between the measuring of audio signals and the recognition of what these signals represent, there are normally several intermediate stages. These stages consume significant resources. Defining different abstract stages enables rapid prototyping and reuse of components. If required, this approach enables also gradual implementation. Subsequently, we provide a conceptual architecture for a context recognition. Its basic differences from existing or proposed architectures can be summarized as follows.

1) The provision of a belief (uncertain knowledge) modeling component.
2) The provision of an adaptation component that exploits knowledge of the resource profile and dynamic workload of a device when computing a context.

To better present the architecture, we separate the recognition components from the adaptation component. The recognition component is shown in Fig. 1, while the adaptation component is shown in Fig. 2. The recognition part of the architecture consists of a set of primitive context servers (PCSs), an aggregator, an empirical ambient knowledge (EAK) component, and a composer.

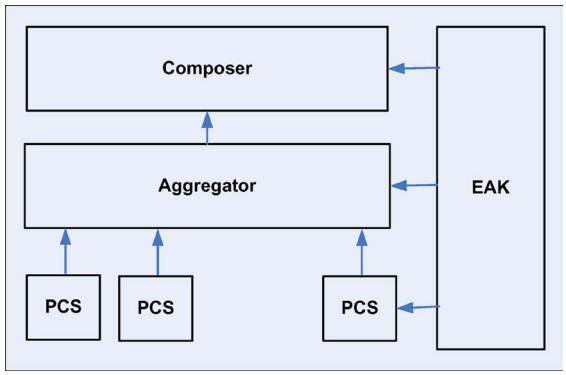

Fig. 1. Conceptual architecture for computing a context.

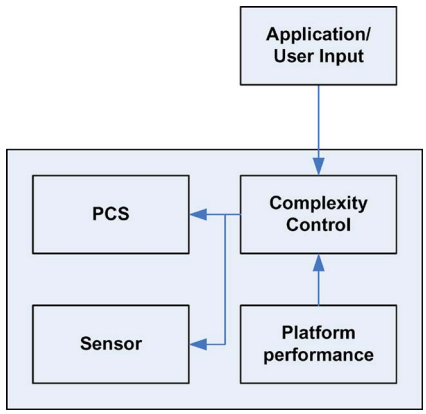

Fig. 2. Adaptation component that ensures the recognition of a context on several devices.

A. Primitive Context Server (PCS)

A PCS abstracts from other components (such as an aggregator) the details and complexities of extracting a meaningful feature or an atomic context from a physical sensor. This feature is not application-specific or situation-specific and can be shared by multiple applications or situations. It refers to a single aspect of a certain phenomenon or a real-world object.

B. Aggregator

A piece of context obtained from a single source may not be sufficient to appropriately model a real-world situation [17]. The real world is far too complex to be captured in complete detail in this way. Aggregation deals with the association, correlation, and combination of data from single or multiple sources to achieve a refined estimation. Subsequently, an aggregator gathers and processes data from multiple sensors which are spatially and temporally related. The outcome is a feature or a set of features that is (are) meaningful for a single application or a set of applications.

C. Empirical Ambient Knowledge (EAK)

A prior knowledge of entities (places, devices, persons, etc.) is useful both for modeling a situation of interest and for appropriately interpreting sensor measurements. In most cases, this knowledge is taken as a fact however incomplete. However, as far as dealing with physical sensors is concerned, whatever

knowledge we have cannot be taken as fact but as an uncertain knowledge or belief. Here are some examples of beliefs.

1) Human thermal and humidity perception varies from season to season. A temperature ranging from 20 °C to 23.6 °C is perceived as comfortable in winter, while in summer, the range from 22.8 °C to 26 °C is perceived as comfortable. Likewise, a relative humidity ranging from 30% to 60% is perceived as comfortable in winter, while in summer, the range from 40% to 60% is perceived as comfortable.
2) In winter, a person is more sensitive to drought than in summer. Hence, the acceptable air velocity inside a room should be below 0.15 m/s, while in summer, it could be up to 0.25 m/s.

The above beliefs can be modeled as uncertain knowledge because they may hold true for most places, but they must not be taken as fact. For these reasons, our architecture separates facts from beliefs. The EAK associates conditional probabilities (or fuzzy membership functions, basic probability mass functions, etc.) to describe the degree of truthfulness of the uncertain knowledge it stores.

D. Composer

The composer computes a higher level context (setting) by classifying freshly acquired evidence from sensors according to the facts and beliefs stored in the EAK.

E. Adaptation

The problem with the recognition architecture is that, in itself, it does not guarantee equal support of all users independent of their end devices, i.e., devices such as laptops, mobile phones, and PDAs. It also does not take into account the dynamic workload of a device. To address these concerns, we have introduced an additional adaptation component that can be plugged into the architecture. It consists of two subcomponents, namely, the platform-performance monitor and the complexity control. This is shown in Fig. 2.

1) Platform-Performance Monitor: A platform has a static and a dynamic aspect. Its static aspect refers to its nominal resource profile, maximum available power, processor speed, memory, networking capability, storage, etc. Its dynamic aspect refers to its present workload and remaining resources. Both aspects should be taken into account to perform context recognition, since the accuracy, as well as the processing time, depends on these aspects. The platform-performance monitor provides the complexity control with an updated information regarding the platform's present workload and available resources.

2) Complexity Control: Context recognition is a tradeoff between recognition accuracy and processing time. The complexity control receives from the consumers of a context (a user or an application) an upper and lower threshold on these two recognition metrics and allocates resources for the recognition of the context accordingly. If a context recognition takes a processing time below a lower threshold, it increases the complexity level of the process to improve accuracy (i.e., by sacrificing more resources). If, on the other hand, a context-recognition process exceeds the upper threshold time, it reduces the complexity in favor of reducing the processing time below the specified upper bound. The complexity itself is determined by a set of parameters. The complexity control dynamically adjusts the accuracy and processing time according to the actual workload of the device.

In Section VI, we will give a detailed explanation of an audio-based context recognition and the parameters that can be tuned to adjust recognition accuracy and processing time. Parameter tuning will be made according to the user's requirement and the device's capability and present workload.

IV. Scenario

We implemented the conceptual architecture for audio-based context recognition. The implementation automates the recognition of various human activities in two different settings.

The first setting is a university campus. At the Faculty of Computer Science (Technical University of Dresden), a Chair occupies several rooms and uses them for different purposes. Some of these rooms are offices, laboratories, a conference room, a library, and a kitchen. There are also other rooms which are shared with other Chairs. The employees may be interested in some of these rooms for various activities including project meetings, thesis presentation, student consultations, impromptu chats among each other, brief discussions with visitors, celebration of a birthday party (graduation), etc.

Some Chairs have online room-reservation systems, but there are times when these systems are not flexible enough. This is because employees should reserve a room well ahead of the intended purpose. Our aim is to complement a room-reservation system and not to replace it. For any impromptu activity, an employee creates a query request in order to find out which rooms are free or occupied by less imperative activities (such as bilateral discussions or a casual chat). The system responds to the query by returning the names and locations of the rooms that satisfy the request without compromising privacy, i.e., without actually providing specific contents pertaining to the activities.

The approach can also be useful in other places. For example, in several big companies, a significant number of employees are mobile and hold universal keys. The keys give them access into their company buildings anywhere in the world. A context recognition can be useful to dynamically locate meetings and seminars; kitchens, special occasions, unoccupied rooms, etc. Recognition of the activities discussed earlier can also be helpful to mobile devices to dynamically adapt to the environments wherein they operate. For example, a mobile phone can dynamically switch from a ringing mode to a vibration or silence mode when an employee enters into a meeting room or holds a presentation.

In the second setting, the activities in trains and trams are recognized. The intention is to help and simplify the task of a human controller. Given the size and the number of carriages in a train, a limited number of controllers may not be able to effectively monitor what is taking place in a passenger train. Dynamic activity recognition will simplify this task.

For the first setting, the contexts of interest are casual talk, party, group discussion, lecture (presentation), and quietness.[1] For the second setting, the contexts of interest are fighting (aggression), loudness (such as drunken sport fans shouting), casual talk, and quiet.

[1] A quiet room is assumed to be empty.

For the two scenarios, there are two different deployment strategies. In the first setting, microphones are carefully placed in different rooms, and the signals from these microphones are processed centrally by a resource-rich computer.[2] Other devices place a higher level query request to this computer in order to learn what is taking place in certain places or where certain activities are taking place. In the second deployment setting, the microphones embedded in mobile devices are used to gather data, and the mobile devices themselves process the data. The second deployment setting does not relay on the existence of any established sensing infrastructure. For both settings, however, adaptation is useful, as the processing time and accuracy are dependent on the capability of the device as well as its present or anticipated workload.

V. Implementation

This section reports the implementation details of the conceptual architecture discussed in Section III for the scenario presented in Section IV.

A. PCS

The PCS is implemented for abstracting the acquisition of audio signals from ordinary microphones which were embedded in ordinary laptop computers. The PCS can be configured or reconfigured at any time to determine the beginning, duration, sampling rate, and resolution of the audio signal being sampled. Once acoustic signals are sampled, it extracts time- and frequency-domain parameters.

In order to extract suitable features, the PCS divides the audio data stream into small time frames. This is useful to model a nonstationary signal as quasi-stationary. There should be a 25%–50% overlap between adjacent frames to compensate the loss of information due to frequency leakage. Frequency leakage occurs due to the abrupt separation of neighboring frames; as a result of which, high-frequency components will emerge at the edges of each frame. This should be removed by a process called windowing, i.e., each frame is multiplied by a window function that decays rapidly toward the edges.

The length of a frame is mostly between 10 and 50 ms, and it influences the recognition accuracy as well as the computation time. To extract temporal features, further processing is not necessary. To extract spectral features, however, at least two additional steps are necessary: frequency-leakage correction and fast Fourier transformation (FFT). After windowing, an FFT is applied on each frame to obtain the magnitude of the power spectrum of each frame.

B. Aggregator

The aggregator extracts application- and domain-specific features from the time- and frequency-domain properties of each frame. The time-domain features include a frame's zero-crossing rate, bandwidth, band energy, and average energy. The frequency-domain features include spectral centroid, spectral roll-off, linear spectral energy, log-spectral energy, and spectral flux. For us, however, the most interesting features are the MFCCs. These are representations of the frequency bands which are Mel-scaled to approximate the human auditory perception more accurately than the linearly spaced frequency bands obtained directly from the FFT. The MFCCs enable context-recognition schemes to "perceive" their surroundings as humans would perceive theirs.

[2]Krysander and Frisk [18] propose an algorithm for optimal selection and placement of sensors to meet a diagnosis requirement specification concerning fault detectability and fault isolability.

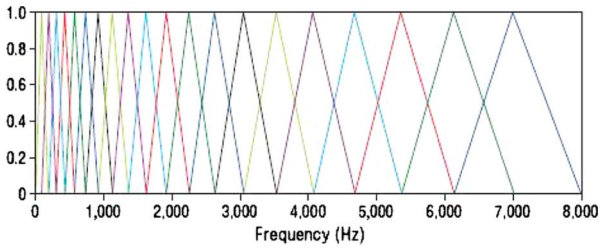

Fig. 3. Triangular filter bank for extracting the MFCCs.

Therefore, the main task of the aggregator is to transform the linear-frequency spectrum obtained by the PCS into MFCCs. This is achieved by first scaling the frequency spectrum logarithmically using the so-called Mel filter bank $H(k,m)$

$$X'(m) = \ln\left(\sum_{k=0}^{N-1} |X(k)| \times H(k,m)\right) \tag{1}$$

for $m = 1, 2, \ldots, M$, where M is the number of filter banks and $M \ll N$. The Mel filter bank is a collection of triangular filters defined by the center frequencies $f_c(m)$, given as

$$H(k,m) = \begin{cases} 0, & \text{for } f(k) < f_c(m-1) \\ \frac{f(k)-f_c(m-1)}{f_c(m)-f_c(m-1)}, & \text{for } f_c(m-1) \le f(k) < f_c(m) \\ \frac{f(k)-f_c(m+1)}{f_c(m)-f_c(m+1)}, & \text{for } f_c(m) \le f(k) < f_c(m+1) \\ 0, & \text{for } f(k) \ge f_c(m+1)\,. \end{cases} \tag{2}$$

The size of the triangular filter bank is variable. These filters are equidistant in the Mel-frequency domain, and there is a 50% overlap between adjacent filters. Fig. 3 shows 20 triangular Mel filters.

The center frequencies of the filter bank are computed by approximating the Mel scale with

$$\phi = 2595 \times \log_{10}\left(\frac{f}{700} + 1\right). \tag{3}$$

The center frequencies on the Mel scale are given by

$$c(m) = m\Delta_\phi \tag{4}$$

where $m = 1, 2, \ldots, M$ and M is the number of filter banks. Δ_ϕ is described as

$$\Delta_\phi = \frac{\phi_{\max} - \phi_{\min}}{M+1} \tag{5}$$

where $\phi_{\max}$ is the highest frequency of the filter bank on the Mel scale and $\phi_{\min}$ is the lowest frequency in the Mel scale.

Finally, the MFCCs are obtained by computing the DCT of $X'(m)$ using

$$c(l) = \sum_{m=1}^{M} X'(m) \cos\left(l \frac{\pi}{M}\left(m - \frac{1}{2}\right)\right) \quad (6)$$

for $l = 1, 2, \ldots, M$, where $c(l)$ is the lth MFCC.

Finally, to reduce the effect of very low and very high MFCC components (at both edges of the Mel spectrum), the so-called *liftering* process[3] is performed. Equation (7) shows a typical liftering function.

$$C'(l) = \left(1 + \frac{L}{2} \times \sin\frac{\pi l}{L}\right) \times C(l) \quad (7)$$

where L is a constant.

C. Separation of Concern

There are two essential reasons for separating feature extraction (by aggregators) from preprocessing (by PCS).

1) Signal processing is independent of any feature, i.e., the outcome can be shared by multiple aggregators that are interested in extracting different features. For example, an aggregator can extract MPEG-7 features instead of MFCCs for an entirely different context-recognition assignment.
2) A significant amount of resource is consumed during signal processing—this will be demonstrated in Section VI. Therefore, it is more efficient for the adaptation component to deal with a single component (a PCS) instead of two.

It must be noted, however, that there is a tradeoff between flexibility and efficiency. Obviously, combining the two processes as a single monolithic process avoids the extra cost of running two independent components. But then, if one is interested in using the same lower level features for different recognition schemes, the monolithic approach does not work. The separation avoids repeating the preprocessing, thereby saving significant computational resources.

D. Composer

The composer receives the most representative and independent higher level audio features from the aggregator and performs estimation or recognition. This is done by computing the likelihood probabilities of individual context types in a well-defined context space. We have experimented with various techniques, including Bayesian networks and HMMs. We choose HMM because it is most convenient to train and requires little prior knowledge.

An HMM is a deterministic, stochastic, and finite-state machine. A Markov chain or process is a sequence of events (called states) of which the probability of each is entirely dependent on the event immediately preceding it. An HMM represents stochastic sequences as Markov chains; the states are not directly observed, but are associated with observable symbols (or evidences), called emissions, and their occurrence probabilities depend on the hidden states. The generation of a random sequence is the result of a random transition in the chain. In order to model a process with an HMM, the following elements should be available.

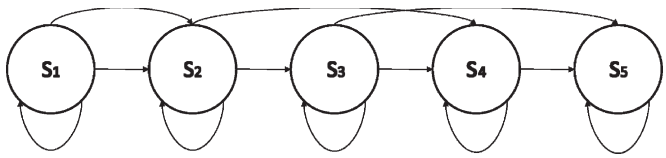

Fig. 4. Left-to-right HMM with five states.

1) The number of states in the model N.
2) The number of observation symbols M, as well as a probability distribution matrix B, in each of the states describing the occurrence of observable symbols.
3) The state-transition probability matrix A.

Given the number of states in the model, $N = (S_1, S_2, \ldots, S_n)$, the state-transition probability matrix, and the current state (at time t) of the model, it is possible to predict the model's state at time $t + 1$.

The state-transition matrix is a square matrix in which each element describes the probability of the model being in state S_j at time $t + 1$ given its immediate preceding state

$$a_{ij} = P(q_{t+1} = S_j | q_t = S_i), \quad 1 \le i, j \le N \quad (8)$$

where q refers to a state. The transition probabilities between all states build a state-transition matrix A of size $N \times N$.

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1N} \\ a_{21} & a_{22} & \cdots & a_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ a_{N1} & a_{N2} & \cdots & a_{NN} \end{bmatrix}. \quad (9)$$

The probability of observing the symbol O_i from a set of symbols $(O_1, O_2, \ldots, O_M)$ when the state machine is in state S_j at time t is given as follows:

$$b_{ij} = P(o_t = O_j | q_t = S_i), \quad 1 \le i \le N, 1 \le j \le M. \quad (10)$$

The probability of observing a symbol in state S_j is independent of all previous states and observed symbols. Subsequently, the $N \times M$ observation matrix can be described as follows:

$$B = \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1M} \\ b_{21} & b_{22} & \cdots & b_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ b_{N1} & b_{N2} & \cdots & b_{NN} \end{bmatrix}. \quad (11)$$

To complete the description of an HMM, knowledge of the model's initial state is required: $\pi = (\pi_1, \pi_2, \ldots, \pi_N)$. The initial state of the model, the transition probability matrix, and the observation matrix together make up an HMM, signified by λ.

$$\lambda = \{\mathbf{A}, \mathbf{B}, \pi\}. \quad (12)$$

The topology of an HMM depends on the process model. Fig. 4 shows the so-called "left-to-right" model which sets a restriction on the way state transitions should be observed—at any given time, either there will be no transition at all or transition should occur in a forward direction only. Even though this restriction is not applicable for audio-based context

[3]The linear-frequency-domain equivalent process is bandpass filtering, while the time-domain equivalent process is smoothing [19].

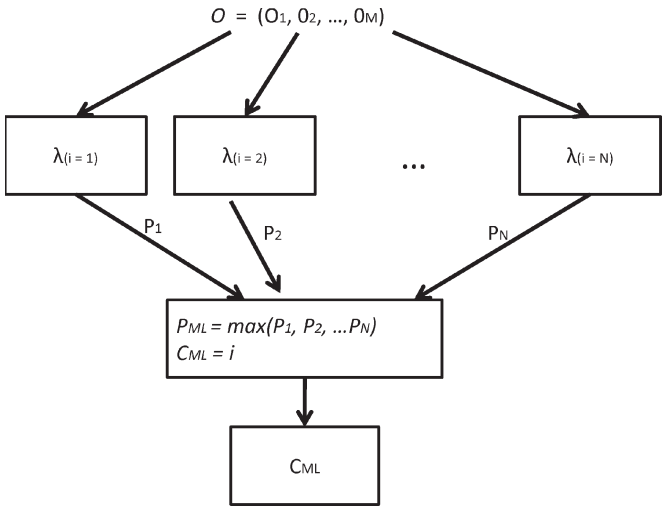

Fig. 5. Recognition of a context using HMMs.

recognition (for example, an audio data may reveal events that repeat themselves), the "left-to-right" topology is useful in modeling time dependences between a sequence of events. For context recognition, the frame duration is normally very short (in the range of milliseconds), and compared to this duration, a meaningful audio scene lasts over several audio frames. Therefore, it is possible to describe an audio scene as a time sequence of several frames (events).

1) Training: An HMM can be trained to configure its model parameters $\lambda = \{A, B, \pi\}$ to a sequence of observed symbols $O = (o_1, o_2, \ldots, o_M)$. In other words, given an initial description of the model parameters[4] λ_0 and a sequence of observation symbols O, the model parameters should be optimized such that the probability $P(O|\lambda)$ is maximum

$$P(O|\lambda) = \sum_{q} \pi_1 b_{q1}(O_1) \cdot \prod_{t=2}^{M} a_{qt-1qt} b_{qt}(O_t). \qquad (13)$$

This is a formidable challenge, since there is no analytic approach to tackle it. However, there are a plethora of estimation algorithms that produce satisfactory results for most real-world situations. We adopt the *Baum–Welch* algorithm and maximize $P(O|\lambda)$ locally.

2) Recognition: During the recognition process, the task of the composer is to determine which of the HMMs (one for each higher level context) the audio data best fit. In other words, given an HMM, λ, and a sequence of observable symbols O, it computes a sequence of states Q that maximize $P(Q|O, \lambda)$. $P(Q|O, \lambda)$ is called the likelihood probability.

We employed the *Viterbi* algorithm to compute the log-likelihood probability distribution because, as it is, $P(Q|O, \lambda)$ is very small.

3) Vector Quantization: It is not possible to directly feed the continuous n-dimensional feature vectors (MFCC) to an HMM, since the model is made up of discrete observation symbols and states. Therefore, the composer maps the n-dimensional feature vectors into a single vector or codebook. The size of the codebook can be tuned between 64 and 256. Fig. 5 shows the sequence of symbols being provided to different HMMs, each of which computes the likelihood probability of the sequence according to its model parameters. The context with the highest likelihood probability is chosen to be the one that is best represented by the audio signal.

[4]This can be an approximation based on a prior knowledge of the context type or an arbitrary assignment of parameter values.

TABLE I
RELATIVE TIME DISTRIBUTION OF A CONTEXT-RECOGNITION PROCESS

Recognition process	Time [ms]	Relative time [%]
Pre-processing	65	20.2
FFT	192	59.8
13 MFCC	25	7.8
VQ code-book size 256	5	1.6
HMM classification	34	10.6
Total	321	100.0

E. EAK

The EAK manages the features extracted from the training data set, establishes conditional dependences between the contexts and the features, and revises its beliefs whenever new data sets are available. Moreover, it stores the structure and the model parameters of the HMMs.

VI. ADAPTATION

A. Defining Complexity Classes

The accuracy and processing time of a context recognition depends on the available resources in a device. It is possible to tune various parameters according to the resource profile and workload of the device. As a result, the context-recognition accuracy and time can be adapted. Parameter tuning requires knowledge of each parameter and its contribution to a context recognition. Table I is an overview of the intermediate stages of the auditory-based context recognition and the relative processing time required by each stage. To produce Table I, we recorded several audio scenes with a nominal duration of 10 s. The audio data were sampled at the rate of 48 kHz, and the context recognition was performed on a laptop computer with a processor speed of 700 MHz and a random-access memory of 256 MB; it had a 4% average workload over a period of 30 min. The frame size was 512 samples with an overlapping percentage of 50%. Each frame had a width of 23.21 ms.

The preprocessing time includes offset-compensation, pre-emphasis, framing, and windowing. Apparently, a large portion of the device's resources is consumed by signal processing (including the FFT) and not by feature extraction or classification. We varied the hidden states of the HMMs and the size of the codebook but could not observe any appreciable change during the entire processing time. This was an essential observation. Since signal processing is done inside the PCS, the adaptation component should deal with it.

Dealing with PCS means tuning various signal-processing parameters. For example, if time instead of accuracy is more important to an application, the PCS reduces the raw auditory data by reducing the sampling rate. The minimum sampling rate is 8 kHz. A sampling rate below this does not fulfill Shannon's sampling requirement and does deteriorate the recognition accuracy significantly. At the same time, increasing the sampling

TABLE II
THREE CLASSES OF RESOURCE PROFILES

Audio Features	Class Low	Class Middle	Class High
Sampling rate [S/s]	8000	11025	22050
Frame Size [samples]	256	256	512
Frame Duration [ms]	46.4	23.2	23.2
Overlapping [%]	25	50	50
Record Duration [s]	1	5	10
Distance between measurements [s]	9	5	0

rate above 22.05 kHz does not improve the recognition accuracy appreciably. Additional parameters that can be tuned by the PCS are the frame size and percentage of frame overlapping. Frame overlapping offers a greater flexibility to adjust processing time and can be varied between 50% and 25%. A 25% overlap can reduce the raw auditory data by more than 30%.

Another possibility of adapting recognition and processing time is by defining different complexity classes (profiles). This classes are taken as static templates into which the computation profiles of mobile devices should fit.

The main challenge here is to define a quantitative relationship between recognition time and accuracy, on the one hand, and resource consumption and CPU workload, on the other hand. We performed an extensive experiment to model relationship between these parameters. This turned out to be a very difficult task. Instead, we applied a heuristic judgment to define three complexity classes that represent the computational capability of different mobile devices. We defined the three complexity classes by assuming that, at any given time, a mobile user may own either a laptop, a PDA, or a mobile phone. This same assumption also enables us to model the different workload of a single device, such as a laptop. According to the classification, we set the upper and lower bounds on recognition time, which can be used to appropriate computing resources for a recognition task. The three classes of profiles are defined as Class Low, Class Medium, and Class High.

Table II provides the description of the three complexity classes for the context types identified in the two scenarios, eight context types in all.

B. Upper and Lower Bounds on the Recognition Time

The time required for computing a context determines the usefulness of the context. We used benchmarking to estimate the upper and lower bounds of the recognition time that can be achieved by each complexity class. A benchmarking can be explained as follows: A device will be given a task of known complexity, and the time required for accomplishing the task is measured. Usually, the chosen task is similar in complexity to the one the device should carry out afterward. If the computation time is not acceptable by the application developer or the end-user, a runtime reconfiguration (for example, reducing the sampling rate) is made so that the complexity of the task is reduced. This implies that the duration of the audio signal representing the audio scene, the preprocessing, the size of the auditory features, the size of the codebook, etc., can be dynamically adjusted.

Table III shows the lower and upper bounds of the recognition time we computed for the three complexity classes.

TABLE III
UPPER AND LOWER BOUNDS OF A CONTEXT-RECOGNITION TIME

Device Class	Lower bound	Upper bound
Class *Low*	50ms	-
Class *Middle*	100ms	200ms
Class *High*	-	400ms

TABLE IV
OVERVIEW OF THE AUDITORY SIGNAL USED FOR TRAINING AND RECOGNITION

Context Type	Total Length of Record Duration
Office	3m 30s
Fight	15m
Party	11m 20s
Loudness	27m 50s
Tram	11m 40s
Train	30m 40s
Presentation/lecture	30m
Casual talk	18m 20s

TABLE V
EFFECT OF FRAME LENGTH ON ACCURACY

Sample Size [samples]	128	256	512	1024	2048
Length[ms]	5.80	11.61	23.22	46.44	91.02
Recognition Accuracy [%]	58.75	80.17	83.46	80.49	77.20

C. Accuracy

Accuracy in the context of this paper should be understood as the number of correct decisions the context-recognition scheme makes. This depends on the following factors:

1) the number of contending context types;
2) the degree of similarity between these contending context types;
3) the amount of auditory data to be processed (by implication, the duration of context-recognition time).

The remaining part of this paper focuses on the third factor. To compute the accuracy of context recognition and to attribute the result to the various recognition parameters, we made repeated experiments. It should be noted that we used ordinary microphones and did not pay much attention to the position or orientation of the laptop in which the microphone was embedded. Our intention was to appropriately represent the way ordinary users handle their mobile devices.

Table IV describes the training scenario. The audio signals were processed according to the three profile classes, namely, the sampling rates were set to 22 050, 11 025, and 8000 Hz, while the record length was adjusted for each context to 10, 5, and 1 s. We used the recorded auditory signals to train and test the HMMs. As mentioned before, during the testing phase, the HMMs computed a likelihood probability distribution for all the contexts of interest.

1) Frame Length: An HMM attempts to recognize a context of interest by constructing a time sequence of audio frames and by creating conditional dependences between these frames. An audio frame represents the smallest unit of information. If the frame duration is too short, an auditory event of longer duration can be divided into many frames, represented falsely as several events; on the contrary, if the frame is too long, several evanescent events can be mistaken for a single long-duration event. Table V displays the effect of frame length on the accuracy of context recognition. As can be seen from the table, the optimal frame duration which yields the highest

TABLE VI
EFFECT OF FRAME OVERLAPPING ON CONTEXT-RECOGNITION ACCURACY

Overlapping [%]	0	12.5	25	50
Number of Audio Frames	43	49	57	86
Recognition Accuracy [%]	78.37	79.85	83.46	82.12

TABLE VII
EFFECT OF AUDIO FEATURES ON ACCURACY

Audio Features	Recognition Accuracy [%]
14 MFCC	79.85
12 MFCC	81.55
10 MFCC	79.68
8 MFCC	78.77
12 MFCC without Liftering	69.78
12 MFCC + log-Energy	83.46
12 MFCC + ZCR	79.43
12 MFCC + spectral centroid	78.05

accuracy is achieved at the sampling frequency of 22 050 Hz, namely, 23.22 ms, with the sample size of 512.

2) Overlapping: Through the windowing process (to attenuate the effect of high-frequency components during the abrupt separation of frames), some information is lost at the two edges of a frame. That is the reason why a frame overlapping is required. In the literature (speech recognition), a 50% overlap is recommended, but surprisingly, for context recognition, the optimal frame overlapping was achieved at 25%.

This can be explained as follows: For speech recognition, the spectral bandwidth is below 4 kHz, whereas for context recognition, the bandwidth is significantly larger (ca. 10 kHz). As a result, a 25% frame overlapping was sufficient to compensate for the lost information at the edges of a frame. Table VI summarizes recognition accuracy as a function of frame overlapping.

3) Audio Features: The selection of the right type and amount of audio features depends on the spectral aspect of the audio scene being processed. For example, for audio events that are made up of frequently changing scenes having both low- and high-frequency components, a large number of MFCCs may be necessary. This way, it is possible to ensure the inclusion of a wide range of frequencies in the extracted features. However, merely increasing the spectral features above a certain threshold may not have any impact on the recognition accuracy.

Our context types involve events that have both slowly and quickly changing scenes as well as low- and high-frequency components. Therefore, we varied the MFCCs from 8 to 14. The result is presented in Table VII.

As can be seen in the table, the number of MFCCs that achieved the highest recognition accuracy is 12 (and not 14!). Perhaps this number would be different for a different record (of the same scenes) or had we employed a different microphone. The best way to explain this is by relating the number of MFCCs with the number of Mel filters [see (6)]. Increasing the Mel filters directly affects the way a frame is subdivided in the frequency domain. This subdivision may result in fragmenting an audio scene into different subbands, as if they were independent events. This can cause an erroneous conclusion.

TABLE VIII
CODEBOOK SIZE VERSUS RECOGNITION ACCURACY

Codebook Size	Recognition Accuracy [%]
32	65.54
64	80.37
128	81.55
256	83.46
512	79.32

Apart from pure MFCCs, we experimented also with additional time- and frequency-domain features[5] to study their combined effect on recognition accuracy.

4) Size of the Codebook: The vector quantization reduces the infinite range of values of the feature vectors into a limited size of code vectors. During this process, if the size of the codebook is significantly small, then many features will be represented by a single vector, and the quantization error will become significant. Subsequently, information which can be vital to the recognition of a context can get lost. On the other hand, making the codebook size considerably large implies the need for a large amount of training data, since only code vectors that appear often in the HMMs during the training phase can be correctly classified. Moreover, the duration of the recognition time will increase significantly. Table VIII summarizes the effect the codebook size on the context-recognition accuracy.

D. Implementing the Adaptation Components

The platform-performance monitoring component stores the resource profile of the device in which a context recognition is performed. Moreover, it periodically or randomly (depending on the configuration) samples the CPU workload and considers the past n samples to estimate device capability. Likewise, the application that binds the recognition system specifies the minimum desired accuracy and processing time, which is the basis for parameter tuning. The complexity control periodically queries the platform-monitoring component to determine how much resource is available and whether the required accuracy and processing time can be achieved. Once again benchmarking is used for the estimation. If the desired accuracy and processing time cannot be achieved, the control component prompts the user to stop some running processes in favor of the context-recognition task or to accept a reduced accuracy or an increased processing time. Accordingly, the complexity control component selects a complexity profile (Class High, Class Medium, or Class Low) or adjust the preprocessing parameters.

VII. DISCUSSION

We evaluated the context-recognition accuracies of the three complexity classes by establishing a confusion matrix for each class. Table IX displays a confusion matrix for the complexity Class *High*. Table X summarizes the result of two confusion matrices for Classes *Middle* and *Low*. In Table IX, the rows represent the actual context types while the columns represent the recognized context types. At the end of each row is given the context-recognition accuracy for each context type. The last

[5]Including spectral centroid, which is the balancing point of the spectral power distribution, and a zero crossing, which is the number of times a time-domain signal crosses the zero reference [20].

TABLE IX
PERFORMANCE OF PROFILE CLASS *HIGH*

Context Type	Office	Fight	Loudness	Party	Casual Talk	Tram	Lecture/Pres.	Train	Correct Decision
Office	44	0	0	0	0	0	0	0	100.0
Fight	0	161	0	0	0	0	0	0	100.0
Loudness	0	0	143	0	0	0	0	0	100.0
Party	0	0	28	21	0	0	0	0	42.8
Casual Talk	0	9	19	1	102	13	0	5	68.4
Tram	0	14	0	0	0	72	1	13	72.0
Lecture/pres.	0	1	2	0	0	9	131	4	89.1
Train	0	0	0	0	0	1	38	111	74.0
Average									83.24%

TABLE X
PERFORMANCE OF PROFILE CLASSES *MIDDLE* AND *LOW*

Context Type	Office	Fight	Loudness	Party	Casual Talk	Tram	Lecture/Pres.	Train	Average [%]
Class L	85	99.7	90.0	17.5	49.3	71.0	40.4	59.7	64.33
Class M	73.3	95.3	68.6	35.1	25.3	79.0	41.3	63.0	60.96

TABLE XI
AVERAGE RECOGNITION TIME FOR THE THREE PROFILE CLASSES

Profile Class	Average computation time [ms]
Class *High*	359
Class *Middle*	109
Class *Low*	16

row provides the average overall accuracy that can be achieved. The average computation time for the three classes of profiles is summarized in Table XI.

Tables IX and X demonstrate that, while some context types (such as "loudness") can be recognized by all-complexity classes with appreciable accuracy, others are not. Similarly, other context types (such as "party," which is starkly mistaken for "loudness") is recognized poorly, regardless of the class types. Of all the parameters we tuned, the sampling rate has a significant impact on the accuracy and time of context recognition. This is not unexpected. Due to the difference in sampling rate, each MFCC represents a different spectral domain. Therefore, for each complexity class, a different codebook is generated in which the code vectors are distributed in different vector spaces.

The profile Class *Low* performs very poorly for all context types containing predominantly spectral components above 4 kHz. One may be led to conclude that Class *Low* devices may not be useful for auditory-based context recognition. However, this is not the case. As can be seen from the tables, even Class *Low* performs well in recognizing "train," "tram," "loudness," and "aggression." This implies that the suitability of a device for a context recognition is partly decided by the type of contexts.

A. Comparison

The preceding sections demonstrate that the time for context recognition and the associated accuracy depend on several factors. While it is essential to make quantitative comparisons between our result and the results of previous work, some reports conceal a wealth of information, making justifiable comparisons a difficult task.

Eronen *et al.* extensively experimented with different types of recognition schemes, changing the topology of HMMs and varying the test-sequence length of audio signals. In summary, they are able to recognize 24 everyday context types with an average recognition accuracy of 58% and 6 higher level contexts with an average recognition accuracy of 82%. With the profile Class *High*, ours is better by 1.24%, but this is a rather negligible figure. Moreover, we consider eight context types while they recognize nine.

Similarly, Korpipää *et al.* achieve a context-recognition accuracy of 96% true positives and 100% true negatives under controlled conditions—nine higher level contexts are considered which are in many respects similar to ours. They offer additional insight regarding uncontrolled environments, in which context transitions are not known beforehand and there are disturbances and undefined phenomena. Their result shows that the overall recognition accuracy falls to 87% true positives and 95% true negatives.

We employed ordinary microphones embedded in ordinary laptop computers, both during the training and the test phases. Furthermore, we recorded the audio signals without much preparation, to imitate the way users handle their mobile devices while moving or carrying out other more important activities. In contrast, Eronen at al. considered various configurations for their experiment: a binaural setup (Brüel & Kjaer 4128 head and torso simulator), a stereo setup (AKG C460B microphones), and a B-format setup (SoundField MkV microphone). The acoustic material was recorded into a digital multitask recorder in 16-bit and 48-kHz sampling-rate format and a Sony (TCD-D10) digital audio tape recorder in 16-bit and 48-kHz sampling-rate format. Likewise, the measurement system hardware of Korpipää *et al.* consists of an extra small sensor box attached to a shoulder strap of a backpack containing a laptop. When collecting scenario data, the backpack was carried around. The measurement system was controlled with a cordless mouse to mark the scenario phases. The microphone was a small omnidirectional AKG C 417/B.

Unlike Eronen *et al.* and Korpipää *et al.*, our evaluation goes beyond recognition accuracy and addresses the relationship and the tradeoff between recognition accuracy and processing time. We defined also three complexity classes to support adaptive context recognition. Moreover, the adaptation aspect of our system enables a user (or an application) to define quality metrics for a context-recognition task. When the specified quality metrics are not achievable because there are not enough resources for the task, the system prompts the user to stop

some running processes. Otherwise, it offers the user a reduced accuracy or an increased processing time.

B. Future Work

In this paper, we used laptop computers for context recognition. While we have tested and demonstrated that three different complexity classes can be emulated with a single device, all the three complexity classes satisfy the minimum resource requirements to process acoustic signals that are below 10 kHz. For example, the minimum sampling rate was 8 kHz. In the future, our aim is to deploy resource-efficient signal-processing algorithms on wireless sensor nodes, which are resource-constrained. The networks that can be established by these nodes promise several applications, but the scope and usefulness of the applications are defined and limited by the energy consumption of the networks. At present, extracting raw sensor data claims a large portion of the energy consumption of wireless sensor networks. Compact and efficient signal-processing algorithms can significantly reduce the data traffic in the networks, either by enabling efficient sampling and data compression or by supporting the extraction of higher level features locally.

ACKNOWLEDGMENT

The author would like to thank Dipl.-Ing. D. Hofmann for his contribution to this paper. During the writing of his Diploma thesis, Dipl.-Ing. Hofmann has performed an extensive laboratory experiment to test the adaptation subsystem of the architecture presented in this paper.

REFERENCES

[1] A. Chen, R. Muntz, S. Yuen, I. Locher, S. Park, and M. Srivastava, "A support infrastructure for the smart kindergarten," *Pervasive Comput.*, vol. 1, no. 2, pp. 49–57, Apr.–Jun. 2002.

[2] W. Dargie and T. Tersch, "Recognition of complex settings by aggregating atomic scenes," *IEEE Intell. Syst.*, vol. 23, no. 5, pp. 58–65, Sep./Oct. 2008.

[3] W. Dargie and T. Hamann, "A distributed architecture for reasoning about a higher-level context," in *Proc. IEEE Int. Conf. Wireless Mobile Comput., Netw. Commun. WIMOB*, 2006, pp. 268–275.

[4] J. J. Magee, M. Betke, J. Gips, M. R. Scott, and B. N. Waber, "A human–computer interface using symmetry between eyes to detect gaze direction," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 38, no. 6, pp. 1247–1261, Nov. 2008.

[5] X. Wang, J. S. Dong, C. Chin, S. Hettiarachchi, and D. Zhang, "Semantic space: An infrastructure for smart spaces," *Pervasive Comput.*, vol. 3, no. 3, pp. 32–39, Jul.–Sep. 2004.

[6] H. Chen, F. Perich, D. Chakraborty, T. Finin, and A. Joshi, "Intelligent agents meet semantic web in a smart meeting room," in *Proc. 3rd Int. Joint Conf. Auton. Agents Multiagent Syst.*, 2004, pp. 854–861.

[7] P. Korpipää, J. Mäntyjärvi, J. Kela, H. Kernen, and E.-J. Malm, "Managing context information in mobile devices," *Pervasive Comput.*, vol. 2, no. 3, pp. 42–51, Jul.–Sep. 2003.

[8] W. Dargie and T. Springer, "Integrating facts and beliefs to model and reason about context," in *Proc. 7th IFIP Int. Conf. Distrib. Appl. Interoperable Syst.*, 2007, pp. 17–31.

[9] M. Y. Nam, M. R. Bashar, and P.-K. Rhee, "Adaptive feature representation for robust face recognition using context-aware approach," *Neurocomputing*, vol. 70, no. 4–6, pp. 648–656, Jan. 2007.

[10] N. Young, M. Bashar, and P. Rhee, "Adaptive context-aware filter fusion for face recognition on bad illumination," in *Knowledge-Based Intelligent Information and Engineering Systems*. Berlin, Germany: Springer-Verlag, 2006, pp. 532–541.

[11] K. Laasonen, M. Raento, and H. Toivonen, "Adaptive on-device location recognition," in *Proc. Pervasive*, 2004, pp. 287–304.

[12] M. Stäger, P. Lukowicz, and G. Tröster, "Power and accuracy trade-offs in sound-based context recognition systems," *Pervasive Mob. Comput.*, vol. 3, no. 3, pp. 300–327, Jun. 2007.

[13] A. J. Eronen, V. T. Peltonen, J. T. Tuomi, A. P. Klapuri, S. Fagerlund, T. Sorsa, G. Lorho, and J. Huopaniemi, "Audio-based context recognition," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 14, no. 1, pp. 321–329, Jan. 2006.

[14] P. Korpipää, M. Koskinen, J. Peltola, S.-M. Mäkelä, and T. Seppänen, "Bayesian approach to sensor-based context awareness," *Pers. Ubiquitous Comput.*, vol. 7, no. 2, pp. 113–124, Jul. 2003.

[15] L. Ma, D. Smith, and B. Milner, "Context-awareness using environmental noise classification," in *Proc. Eurospeech*, 2003, pp. 2237–2240.

[16] D. Smith, L. Ma, and N. Ryan, "Acoustic environment as an indicator of social and physical context," *Pers. Ubiquitous Comput.*, vol. 10, no. 4, pp. 241–254, Mar. 2006.

[17] A. Padovitz, S. W. Loke, and A. Zaslavsky, "Multiple-agent perspectives in reasoning about situations for context-aware pervasive computing systems," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 38, no. 4, pp. 729–742, Jul. 2008.

[18] M. Krysander and E. Frisk, "Sensor placement for fault diagnosis," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 38, no. 6, pp. 1397–1410, Nov. 2008.

[19] M. Benzeghiba, R. De Mori, O. Deroo, S. Dupont, T. Erbes, D. Jouvet, L. Fissore, P. Laface, A. Mertins, C. Ris, R. Rose, V. Tyagi, and C. Wellekens, "Automatic speech recognition and speech variability: A review," *Speech Commun.*, vol. 49, no. 10/11, pp. 763–786, Oct./Nov. 2007.

[20] M. Döller and H. Kosch, *The MPEG-7 Multimedia Database System (MPEG-7 MMDB)*, vol. 81. New York: Elsevier, 2008, pp. 1559–1580.

Waltenegus Dargie (M'08) received the B.Sc. degree in electrical engineering from the Nazareth Technical College, Adama, Ethiopia, in 1997, the M.Sc. degree in electrical engineering from the Technical University of Kaiserslautern, Kaiserslautern, Germany, in 2002, and the Ph.D. degree in computer engineering from the Technical University of Dresden, Dresden, Germany, in 2006.

He was a Researcher with the Fraunhofer Institute of Experimental Software Engineering, Kaiserslautern, between 2002 and 2003, and with the Department of Electrical Engineering and Computer Science, University of Kassel, Kassel, Germany, between 2002 and 2005. He is currently a Researcher with the Technical University of Dresden. His research interests include autonomous computing, context-aware computing, wireless networks, and digital signal processing.