Pers Ubiquit Comput (2006) 10: 241–254
DOI 10.1007/s00779-005-0045-4

ORIGINAL ARTICLE

Dan Smith · Ling Ma · Nick Ryan

Acoustic environment as an indicator of social and physical context

Received: 5 October 2004 / Accepted: 15 April 2005 / Published online: 7 October 2005

Abstract Acoustic environments provide many valuable cues for context-aware computing applications. From the acoustic environment we can infer the types of activity, communication modes and other actors involved in the activity. Environmental or background noise can be classified with a high degree of accuracy using recordings from microphones commonly found in PDAs and other consumer devices. We describe an acoustic environment recognition system incorporating an adaptive learning mechanism and its use in a noise tracker. We show how this information is exploited in a mobile context framework. To illustrate our approach we describe a context-aware multimodal weather forecasting service, which accepts spoken or written queries and presents forecast information in several forms, including email, voice and sign languages.

Keywords Acoustic environment · Context awareness · Classification · Machine learning · Adaptive feedback · Mobile computing

1 Introduction

The ability of a system to use contextual cues to choose appropriate behaviour for a situation and to adapt that behaviour in response to changes in the user or device context is essential to the development of adaptive information systems. Context awareness is particularly important for mobile computing, where a user's situation may change rapidly and interaction is constrained by both limited device capabilities and the user's activity. Early context-aware applications mostly focused on sensing and understanding absolute location, user identity and time, but there has been an increasing interest in identifying different user activities.

We are interested in obtaining and exploiting location and activity information through the analysis of the acoustic environment. The acoustic environment is a rich information source for deriving information about a user's current situation, potentially enhancing the description of both the location and the user's activities. It is almost always available and can be readily sampled by a wide range of devices.

To show how the acoustic environment can be exploited in context-aware applications we describe a prototype context-aware weather forecasting application, which is capable of modifying its users' requests according to their current environment and known preferences—exploiting context information to infer and expand information requests by exploiting knowledge of their daily routines. More immediately, the acoustic environment allows context-aware applications to modify their output modalities, for example giving audio information to drivers, and visual information to participants in a meeting. (The sources of information on user preferences, routines, etc. are beyond the scope of this paper.)

In order to test the feasibility of environmental noise recognition in mobile devices we have developed and evaluated a Hidden Markov Model (HMM) classifier. Our initial work, to classify the typical environments of our daily life, such as an office, car and street, has been reported elsewhere [1, 2]. In this paper we summarise these and further series of experiments which address a number of issues, including requirements for building applications on limited capability devices, confidence measures, adaptive learning and single-sourced sound event classification.

The context models of environmental noise have been integrated into our prototype context tracker application. The tracker is able to recognise acoustic environments and adapt to new environments. It uses current

D. Smith (✉) · L. Ma
School of Computing Sciences, University of East Anglia,
NR4 7TJ Norwich, UK
E-mail: Dan.Smith@uea.ac.uk
E-mail: Ling.Ma@uea.ac.uk

N. Ryan
Computing Laboratory, University of Kent,
CT2 7NF Canterbury, Kent, UK
E-mail: N.S.Ryan@kent.ac.uk

data to reinforce existing environmental noise models. Users are able to create and train personalised models based on their daily routine. All recognised and confirmed environments are recorded into a tracking file for later retrieval and are annotated for use by other components in multimodal systems.

Our approach using environmental noise in context-aware applications differs from the previous work, with its emphasis on rapid recognition, minimising computation cost, adaptive learning and easy training. We classified 12 types of environmental noise based on a user's daily routine. By classifying short-duration (3 s) samples, the application can rapidly recognise an environment. We have found that the recognition based on lower-quality data (8 kHz, 8 Bit PCM) provides usable results with limited capability devices and low-bandwidth communications.

The remainder of this paper is structured as follows: Sect. 2 is a review of related work on context-aware applications and auditory sound scene analysis. In Sect. 3 we describe our new series of environmental noise classification experiments for low-bandwidth communication and analyse the results. In Sect. 4 we describe our tracker application. The mobile infrastructure we are using is described in Sect. 5. Section 6 contains a description of our weather forecast information exemplar. Our conclusions are presented in Sect. 7.

2 Related work

Context-aware computing is a pervasive or ubiquitous computing paradigm in which the interaction with a computer is driven by an externally derived or implicit context. There has been a broad scope of topics in context-aware computing, such as understanding of context, software architecture, interface design, infrastructure, sharing context and privacy and security of context awareness [3]. Technologies and applications focus on the sensing, capturing, presenting and modelling of context [4].

The notion of context-aware computing was first proposed in the early 1990s. The Active Badge system [5]—probably the first context-aware application—was published in 1992 and Schilit [6] introduced the term "context-aware" in 1994. Dey's definition and discussion of context [7, 8] remains widely accepted today. Anhalt et al. [9] point out that a pervasive computing environment must be context-aware. They define an activity-attention framework for context-aware computing and introduce a pervasive computing architecture. Brown [10] discusses software design to make the creation of context-aware applications easy. Other early works defining context and the development of context-aware applications can be found in several surveys [6, 11–13].

Location-awareness has been an especially popular research area in context-aware computing. The technologies mainly used are short-range IR and RF signals for indoor applications and GPS outdoors. Examples include services such as Conference Assistant [14], Office Assistant [15] or guides, such as Cyberguide [16], Shopping Jacket [17] and C-MAP [18] and memory aids such as Stick-e notes [19], Memoclip [20] and Cybre-Minder [21]. Jimminy [22] is a wearable personal note-taking and note-archival application; the reported experiments suggested that physical context alone was a poor predictor for retrieving relevant notes, but that it was useful when combined with information from any current notes.

Recent research has explored context from a variety of media, such as image, audio, video and a wider variety of environmental and personal sensors.

Audio is an appropriate medium for context-aware computing and it plays an important role in many projects. ComMotion [23] is a location-aware computing environment using GPS, but the core set of reminder creation and retrieval functions can be managed completely by speech in order to use it on mobile devices. The Nomadic Radio project [24] developed interaction techniques for managing voice- and text-based messages in a nomadic environment. It employs auditory I/O, with speech and non-speech audio, for navigating and delivering messages. Gerasimov and Bender [25] describe using sound for device-to-device and device-to-human communication and have explored the possibility of using the existing audio capability in many commonplace devices. Audio Aura [26] explored the use of background auditory cues to provide serendipitous information coupled with peoples' physical location in the workplace. Schmandt et al. [27] considered some aspects of message delivery including minimising interruption, adaptation to the user, location awareness, and non-intrusive user interfaces in their Everywhere Messaging project. Hindus and Schmandt [28] focused on ubiquitous audio for acquiring and accessing the contents of conversations. They described applications for capturing and structuring audio from office discussions and telephone calls for later retrieval. Huang et al. [29] used audio and video information to detect activity in an office.

Many research approaches on audio data can be explored from the perspective of content-based audio classification or audio information retrieval. Foote [30] and Wold et al. [31] reviewed audio classification systems using multiple features and a variety of classification techniques including static modelling and dynamic modelling. Content-based audio classification and retrieval research has been mainly based on speech and music, focusing on searching and indexing. Much work has been done on speech recognition, word spotting, speaker identification, speech interfaces and music information retrieval.

Some research has contributed to the real world background noise classification; following [32], Couvreur [33] has introduced three classifiers (statistical, adaptable and adaptive and HMM-based classifiers) to be used in separated noise event recognition and has attempted to develop a classifier for multiple simultaneous signals.

Statistical classifiers have the major drawback of their sensitivity to variations in utilisation conditions. Adaptable and adaptive classifiers that adapt to changes in spectra can solve this problem, but do not take advantage of the time-evolving structure of the spectra. HMM-based classifiers provide a dynamic solution. Gaunard et al. [34] have implemented an HMM-based classifier to recognise five noise events (car, truck, moped, aircraft and train). They observed that the frame length in noise recognition is larger than that in speech recognition. Their best results are from a five-state HMM using LPC-cepstral features, which gives better results than human listeners. We have conducted a small series of experiments which show similar results. Nishiura et al. [35] have classified 92 types of environmental sounds using an HMM framework for robust speech recognition. Their experimental results show how the HMM can accurately identify and classify single-occurrence environmental sound sources, speech and speech with sounds.

The classifiers described above recognise individual sound events. Some classification systems have been proposed to recognise auditory scenes. Peltonen et al. [36] demonstrated that mel-frequency cepstral coefficients out-performed other feature representations. Sawhney [37] classified five everyday noise types, comparing several approaches, of which filterbank with Nearest Neighbour (NN) clearly out-performed the others. Their results indicate that frequency bands generated from filterbank analysis of frame-by-frame audio windows provide robust features. Other work has been directed towards recognising both the scene and sound subjects in it, focusing on exhaustively identifying sound events and their relationships in a continuous classifier.

Much work on environmental noise has been done to separate speech from background noise for robust speech recognition (e.g. [40]). Only a few context-aware applications have attempted to use environmental noise. The focus of Sawhney's [37] work is on distinguishing longer duration (15 s) environmental sounds into pre-defined classes (people, voices, subway, traffic and other), using near-real-time classification techniques; the system reported in [38, 39] is designed around an adaptive structure that relied on unsupervised training for segmentation of sound scenes. Their system detects and classifies both events and scene using an HMM framework.

3 Experimental work

In this section we summarise the results of our experimental work in environmental noise and sound event classification and recognition. These results demonstrate the feasibility of classifying acoustic environments using consumer devices. In subsequent sections we show how this classifier can be used in context-aware applications, and how acoustic environment information is incorporated into a general context infrastructure.

The corpus of recordings and samples used in all the experiments summarised here is available for further research [1]. This section summarises the experimental results presented in [1, 2].

The first set of experiments used a Sony MD recorder/player and electret condenser microphone clipped to clothing. For the second series we used an MP3 player/recorder (Samsung YP55H) attached to the strap of a shoulder bag as the recording device to capture the environmental noise from a typical daily routine. We have not addressed the issues of the microphone being obscured by, for example, being placed in a pocket, as we have found it easy to mount the microphone in permanently exposed locations.

3.1 Environmental noise classification

Our initial experiments were conducted with high-quality sound recordings sampled at 22.050 kHz using 16-bit quantisation, where we achieved an overall accuracy of 92.27% with our best model. However, many context-aware applications are intended to run on resource-constrained mobile devices. In order to test the feasibility of implementing the models in mobile devices, we conducted a further series of experiments to test our noise classifier's suitability for low-bandwidth connections and for real-time processing in personal mobile computing platforms. The acoustic environments recorded for these experiments are intended to reflect a user's daily routine.

Every environment has its characteristic consistent and periodic background noise. Our focus is on modelling slow-changing attributes of the environmental noise in the audio signal, as we are concerned with the overall acoustic environment as an indicator of the user's activity.

The overall accuracy of the low-bandwidth experiments is 95.83% (1,150 out of 1,200 samples correctly identified), slightly better than the initial experiments. The recognition accuracy of individual environments ranged from 81 to 100%, with the building site, urban driving, office, presentation, city street and supermarket giving 100% classification accuracy for the 100 examples tested for each. All incorrectly classified samples were recognised as similar environments; for instance bus noise was confused with other forms of transport and shopping mall samples were confused with supermarkets.

The accuracy of noise recognition depends on a number of factors such as the amount and coverage of the training data, the feature extraction component, the allowable computational complexity and the model parameters. The noise classifier designed in this system is not capable of recognising multiple simultaneous environmental noises. For example sitting in an office while a car is passing by would cause a conflict.

[1] http://www.cmp.uea.ac.uk/research/noise_db

3.2 Sound event classification

The environmental noise models' flaw is the possibility of incorrect classification when a sound event occurs close to the microphone, for example a telephone ringing in an office. The 3-s recording would then be full of ringing sound rather than the general office background sound, potentially causing an incorrect classification. One solution to the problem of classifying multiple and simultaneously occurring environmental noises is to build models of those sound events to give a cue to the environment. These sound event models could provide a context-dependent knowledge of the environment. Hence contextual cues could be incorporated in classification with the environment models.

The experiments using sound event models were conducted using the data of the RWCP Sound Scene Database in Real Acoustic Environment [35] models for sound events can be obtained using our HMM classifier. However there are subtle differences between recognising isolated sound events and identifying environmental noise. Environmental noise is a complex sound made up of a mixture of different events with no constraint on what these sounds can be. For example, consider an office environment, where a stationary component may come from an air-conditioning fan, a quasi-stationary component from key-clicks and non-stationary events from people moving around, opening doors and talking. In contrast, sound events are produced from single sources and are constrained to single locations. The sound event sample has limitations on the character of sound and has clear states in it. A typical sound event can be divided into five states, silence–begin–middle–end–silence. When we add the two emit states in HTK [41], a seven state HMM topology is considered to be the best model of the sound event.

We obtained an overall accuracy of 84.6% for the 105 types of sound events from the 2,909 testing data samples. Results show that the HMM classification works very well with the characteristic sounds (95% accuracy), especially instruments, electronic and mechanical sounds. Accuracy on the action sounds (including clapping, rubbing, dropping) was 87% but the model gives 70% accuracy with the collision sounds. These apparently poor results are a consequence of the finer subdivisions used in the collision sounds and disappear when a coarser classification granularity of classification—similar to that of the other groups—is used.

3.3 Adaptive learning

In order to be practically useful in context tracking, a classifier must be adaptable to new environments, and applications that use the tracking information typically need confidence estimates on the classification.

The recogniser features a confidence scoring mechanism using a simple method of N-best likelihoods of these environmental noise HMMs. Let L_1 be the log likelihood of the best matching model, and L_2 be the log likelihood of the second best matching model. The confidence measure for the recognition is then computed as:

$$\text{conf} = \frac{L_1 - L_2}{L_1} \times (-1).$$

The acoustic environment is dynamic, in that the characteristics of a location or activity may change (e.g. moving from a quiet individual office to a large air-conditioned office), and we encounter new environments. Most environments also change incrementally (e.g. through different bus models being used on a regularly used route). Therefore, adapting to changes of environment and continuously learning new environments are important requirements for acoustic environment trackers.

Such changes require continuously updating existing models with new training data and adaptively building more robust models. However, if the training set size increases continuously, the models become more ambiguous and diverse. To avoid these problems we remove old samples from the training set as new 'good' samples are added; this approach is similar to [42]. We believe this strategy improves the recognition accuracy for environments where the initial performance is poor and allows us to learn new environments with minimal initial training data.

3.4 Hierarchical noise model

The classification granularity of acoustic environments and sound events is largely determined by the purposes for which the classifications are designed. Similarly, suitable sampling strategies are determined by the applications using the context information. The focus of the work described here is on meeting information needs of people in their daily environments and routines.

In this initial hierarchical noise model there are just two levels, environments (office, supermarket, street, etc.) and sound events (telephone ring, hand clap, etc.). The classifier first compares the unknown sample against the set of environment models. If an uncertain environment is detected (measured by confidence score), the sample is compared against the set of sound events to find a cue. If confusion still exists, other contextual information might aid in reducing ambiguities.

The noise model also supports hierarchies of environments, for example allowing bus, train and car to be grouped under a more general transport label. The model allows user-defined hierarchies to be created. The use of suitable acoustic environment hierarchies allows us to assign samples with poor confidence scores to a more general class. In this case the model chooses the lowest class in the hierarchy that subsumes the first and second matches (i.e. those used in the confidence score).

4 Acoustic environment tracker

Context-aware applications gain substantially from environmental noise classification because the typical auditory environment of a mobile user is dynamic and structured. Our HMM classifier has high accuracy and negligible computational cost and so can easily run on limited capacity devices. We have investigated the feasibility of this approach through a prototype context-aware application using the built-in microphone in the mobile device with periodic audio buffering of environmental noise.

We have developed an application for a context-aware weather information service. The focus of this application is on rapidly recognising a user's current environment by short duration sound recordings, using the trained models, together with contextual information from other sources. This application also adapts to changes in the environment and to new environments which are not pre-classified by the initial models. Besides a set of general models, personalised models can be trained to satisfy individual needs, requiring only initial human input to identify and name the new environment. Context tracking and transitions are recorded in a log file. The logical architecture is shown in Fig. 1.

4.1 Client

The client prototype consists of five modules: environmental noise capture and recognition, training, context tracking, model update and browser. Some components are developed using the Java sound API [43]. Figure 2 shows the user interface of the capture/recognition module.

The application senses two contexts: environmental noise and time. The microphone is a sensor for environmental noise and the built-in clock is the time sensor. The application periodically invokes an audio recorder to sample the environmental noise. Timestamps are automatically added to each captured noise sample.

The sample duration and the sampling interval are user-controllable. By default, the duration is 3 s and interval is 60 s. The user can also control the quality of the samples depending on the device capability; the default audio format is PCM 8.000 kHz, 8 Bit, Mono.

The recorded sample is sent to the noise recogniser and the result is shown on the screen together with confidence and captured time (see middle part the interface). If the confidence is below the threshold, samples are taken continuously, regardless of the interval setting, until it reaches the threshold. If an environment change is detected, samples are taken more

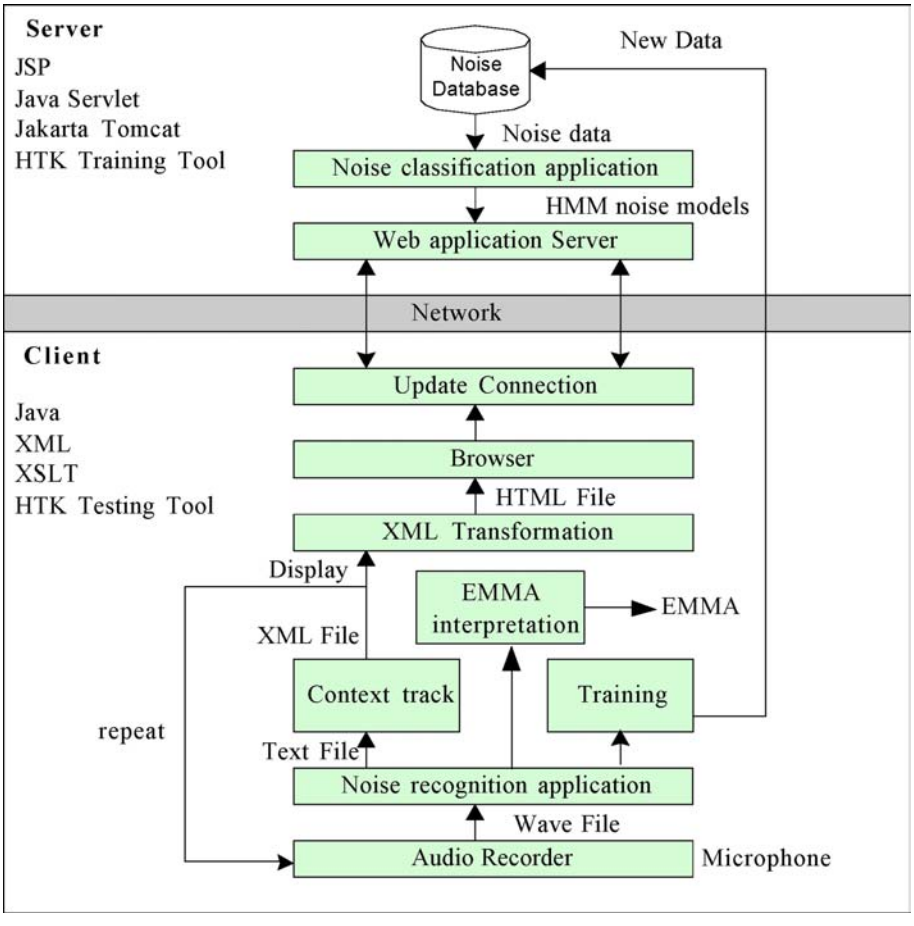


Fig. 1 Tracker logical architecture

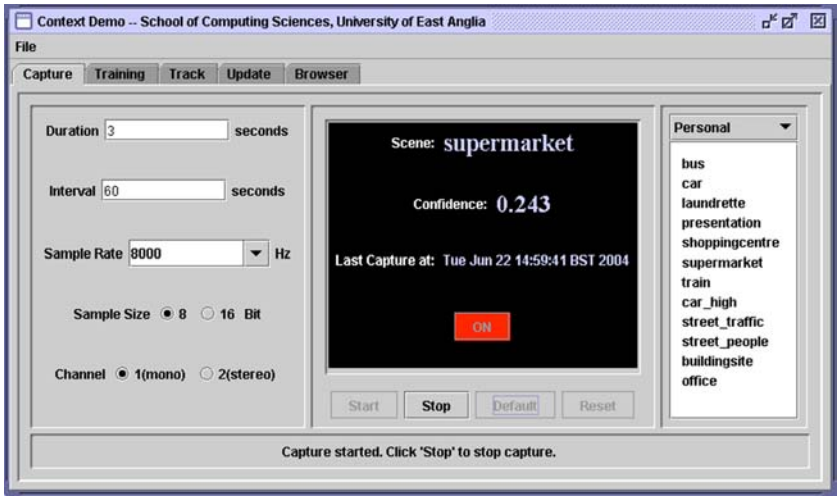


Fig. 2 Environmental noise capture/recognition

frequently until the change is confirmed. When low confidence results continue for a long period, the application prompts the user for information about the environment. If the user is unable to respond immediately, for example they are driving or in a meeting, training starts automatically and the user is prompted to input the environment name at a later time.

Every new user gets a set of general models as the starting point. The general models may not work well due to differences between the real environments and those used to train the model. For instance, the office used for training was a small, quiet, air-conditioned office. A user with a large office without air-conditioning and with windows opening over a busy street, might expect a poor initial match. Once the personalised models are trained, users can switch between the general models and the personal models.

The default sampling regime is to take a 3-s sample every 60 s. If the confidence in the sample is low, or if the recognised environment has changed, the system takes continuous samples until a sequence of samples is recognised with high confidence as coming from a known environment or the system invokes the training module. This dynamic sampling behaviour allows the system to react rapidly to changes in the users' acoustic environments (and, by inference, changes in their activities). A topic for future work will be to investigate different adaptive sampling strategies.

Training is a core module of this application for adaptive learning. Training can be performed manually by the user to train the personalised model or started automatically when an uncertain environment is found.

Figure 3 shows the training interface of the application. The user needs to specify a name of the new environment and record a sample for a few minutes. They may control the recording quality and length. If it is a new environment, the user can input a name for it, for example 'My Lab'. Alternatively, they may select a name from a list if it is an existing environment. Selecting a name from the list can also avoid multiple definitions of similar environments, for example 'bar' and 'pub'. When an automatic training session starts, it uses the default setting, which is PCM 8.000 kHz, 8 Bit, Mono, 5 min length. The recorded sample is then sent to the server to perform training.

The server periodically (e.g. daily) gathers new samples and trains models. It updates models and information once the new models have been trained and are

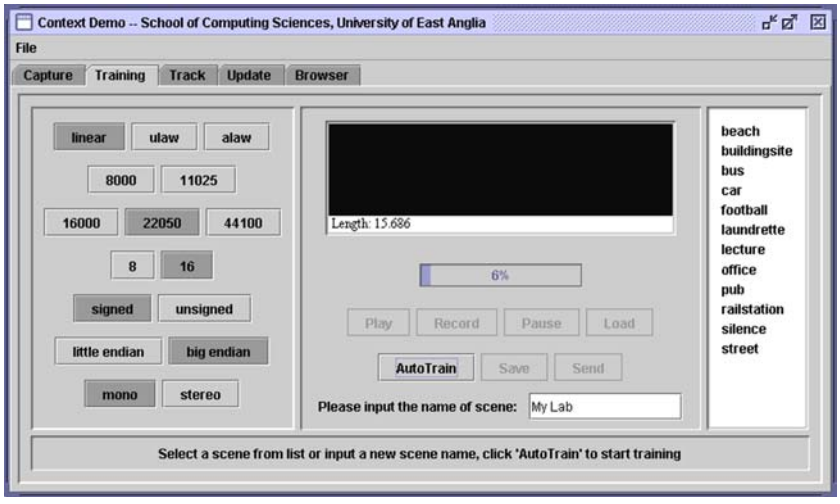


Fig. 3 Training

ready to be downloaded to the client. Users are then able to update the general models, personal models and configuration files (Fig. 4).

4.2 Application for PDA

We have also developed a version of the tracker application which runs on a PDA. It has a substantially simplified interface, and the recogniser can be moved to the server side to further reduce the tracker's footprint on the PDA (Fig. 5).

4.3 Server

The server comprises the environmental noise database, training tool, web application server, a set of environmental noise models and a service module. There are two parts to the environmental noise database; the original manually recorded samples and the new samples automatically uploaded by the user. The original samples were used to train a set of environmental noise models which were delivered to the client application as the initial general models. The service application continues to accept and save new samples which are uploaded from client applications into the new sample database. These new samples include samples of new environmental noise models and samples of existing environmental noise models. These are then used to build new models or improve existing models. Training is performed on a daily basis or depending on the number of new samples.

5 A context-aware infrastructure

In order to use the results produced by the environmental noise classifier as contextual input to context-aware programs, we have adapted the classifier to work as a

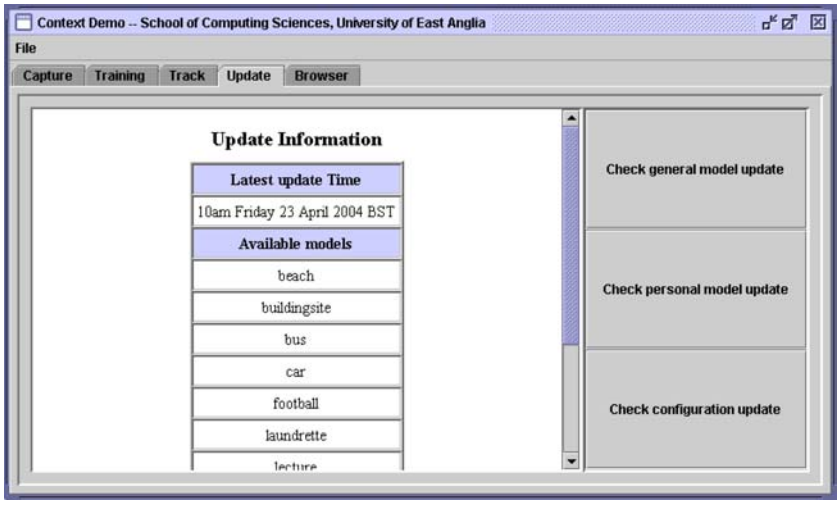
Fig. 5 Interface on PDA

Tracker component of the Kent MobiComp infrastructure [44]. Built on earlier work in context-aware field recording tools [45], MobiComp was originally developed to support context sharing in a range of mobile applications [46, 47]. The current version provides a simple API for building distributed ubiquitous computing and context-aware applications.

The core element of the infrastructure is the ContextService (Fig. 6), a simple interface to a tuplespace [48, 49], extended with event notification. This acts as a store for context elements and enables coordination between the components of context-aware applications. The approach here is similar to that employed in several other ubiquitous computing support infrastructures, for example the Stanford Event Heap [50].

The storage components behind the ContextService interface may be configured to support different scales of context-aware applications: for simple, stand-alone, applications, for multiple applications on a single device or for distributed storage. In the latter case, servers at well-known addresses can be employed as proxies for

Fig. 4 Updating



mobile devices where their network connections (e.g. via GSM/GPRS) prevent direct requests from the Internet to the device.

The ContextService interface provides 'put', 'get' and 'remove' methods to support the tuplespace model, registration and notification methods to support an event-based model and avoid the need for continuous polling for interesting events. In addition, a general query interface is being developed to enable clients to enquire about the content of the store, retrieve element schemas and to extent the simple 'get' interface with general-purpose XQuery [51] requests.

Context producers (trackers) register their availability and capabilities by putting appropriate elements into the tuplespace. Their purpose is to collect raw context data from sensors such as GPS receivers, other dynamic sources such as the noise classifier described above, or static sources such as configuration files for device capabilities and user preferences. Trackers transform their input into context elements which are then put into the tuplespace. The noise classifier client described in Sect. 4.1 above has been modified to act as a tracker, logging its output to the ContextService shown in Fig. 6.

Context elements take the form of a subject–predicate–object triple, relating an entity identifier to a, possibly complex, named context value. Additionally, elements carry a production timestamp, a default validity period, and a privacy level indicating how they may be disseminated through the ContextService. The object part of a context element may be arbitrarily complex, and different trackers might produce elements with similar names but different semantics. Equally, similar information may be packaged in different forms. As a first step towards wider interoperability, trackers are required to supply XML Schema fragments for each element they may produce as part of their initial registration with the ContextService.

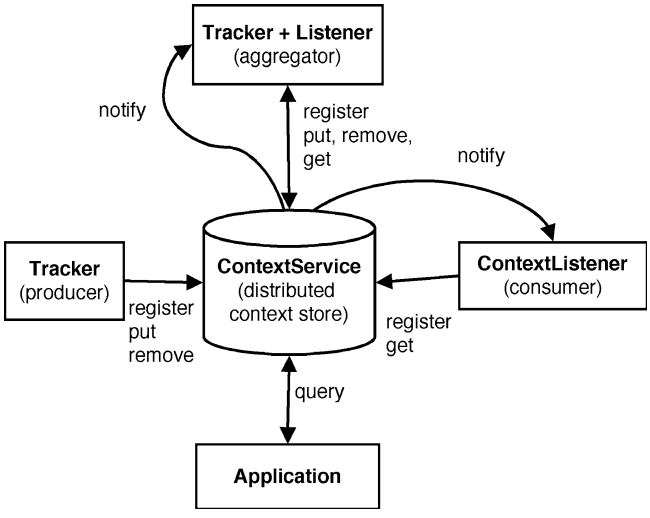


Fig. 6 The Kent MobiComp infrastructure

Element communication between infrastructure components takes the form of XML documents based on the ConteXtML schema which has been developed to support the infrastructure. Typical location, velocity and noise classification elements are shown in Fig. 7.

ContextListener components consume context elements. They typically register an interest in one or more entities and/or particular elements and receive event notifications whenever a corresponding element is put into or removed from the tuplespace. On receiving a notification, the listener may get the element from the tuplespace and use it as required. Context aggregators may be constructed by combining tracker and listener functions. Here, the tracker monitors events from the ContextService, rather than a sensor device, and applies a transformation before returning a new element to the tuplespace. Aggregators can combine several low-level sensor elements to produce an element at a higher level

```
<?xml version="1.0" encoding="UTF-8"?>
<context xmlns="http://www.mobicomp.org/ConteXtML/">
<contextElement privacy="public" timestamp="2004-09-
24T11:18:45" lifetime="600">
 <subject>4da941681d39c92095e73fc59f2</subject>
 <predicate>location.point</predicate>
 <object type="SpatialObject">
  <spatial srs="BNG" source="GPS"><point>
    <x>553264.4</x><y>252387.3</y><z>98.3</z>
   </point></spatial></object>
</contextElement>
<contextElement privacy="public" timestamp="2004-09-
24T11:18:45" lifetime="120">
 <subject>4da941681d39c92095e73fc59f2</subject>
 <predicate>velocity</predicate>
 <object type="Velocity">
  <velocity source="GPS">
   <speed unit="m/s">13.8</speed>
   <dir>266.0</dir></velocity></object>
</contextElement>
<contextElement privacy="public" timestamp="2004-09-
24T11:18:52" lifetime="600">
 <subject>4da941681d39c92095e73fc59f2</subject>
 <predicate>environment.noise</predicate>
<predicate>environment.noise</predicate>
 <object type="NoiseClassification">
  <NoiseClass id="noise-20040924111849">
   <start>2004-09-24T11:18:49</start>
   <end>2004-09-24T11:18:52</end>
   <model> http://sll12.../contextaware/.../personal.xml"</model>
   <process> http://sll12.../contextaware/noiserec.xml</process>
   <source> http://sll12../contextaware/.../mymic</source>
   <signal> http://sll12.../contextaware/.../n31053.wav</signal>
   <signal-codec>audio/pcm</signal-codec>
   <confidence>0.725</confidence>
   <medium>acoustic</medium>
   <device>Samsung-mp3</device>
   <mode>noise</mode>
   <environment>car</environment>
  </NoiseClass></object>
</contextElement>
</context>
```

Fig. 7 ConteXtML representation of MobiComp context elements for location, velocity and noise classification (Fig. 9). ContextElement example

of abstraction. For example, information from temperature, door, window and light sensors might be used to determine room occupancy. Other aggregators may perform simple transformation services, such as converting latitude and longitude coordinates from a GPS sensor to coordinates on an appropriate local or national grid.

Many non-trivial context-aware applications take the form of complex context aggregators, for example, the FieldMap application described in [45]. In Sect. 6 we describe a context agent that uses location, activity, device capabilities and user preferences to refine queries before passing them to a target web application. When the result is returned to the ContextAgent it is modified according to user preferences, device capabilities and other constraints.

6 Exemplar: weather forecast information

The information on user contexts which links output modalities, devices and request domains provided by the ContextService makes it possible to add context-aware capabilities to conventional information services. Here we describe this in relation to weather information provision. We present the logical architecture of our prototype weather forecast information system. It uses contexts obtained from the ContextService to modify both queries and responses to meet constraints of location, environment, device capabilities and user preferences.

The following scenario motivates the weather information application, showing how different environments can modify requests (input modalities) and responses (output modalities). A crane operator, while at work, is normally only interested in wind speed and prefers to receive this as an SMS message on his mobile phone; he may simply submit the query 'weather', from a building site environment, which is expanded to be the wind forecast at his current location for the rest of the day. His preferences may state that the same query issued from a heavy vehicle environment is expanded to be the general forecast between his current location and his depot. In this instance, the application recognises that the context is a vehicle and delivers the information as a voice message. (It is safer to assume that the request is made by the driver—a passenger can override this assumption through further interaction.) The same request—'weather'—issued from his PDA at the weekend in a coastal environment will give him the detailed coastal forecast as in text format on his screen. The remainder of this section describes the architecture and functionality of the application.

6.1 Logical architecture overview

The range of context information that may be used in modifying the original query is dependent on what sources are available (as elements in the ContextService tuplespace) and whether the application can exploit that information. In this example we are looking at acoustic environment and input modality, combined with static information on the user, device and output format preferences.

The overall logical architecture of the system is shown in Fig. 8. The core of the system is the ContextAgent which includes the language model agent, data source agent and modality agent. The ContextAgent interacts with the ContextService, input modality, required application and output modality agents. The goal

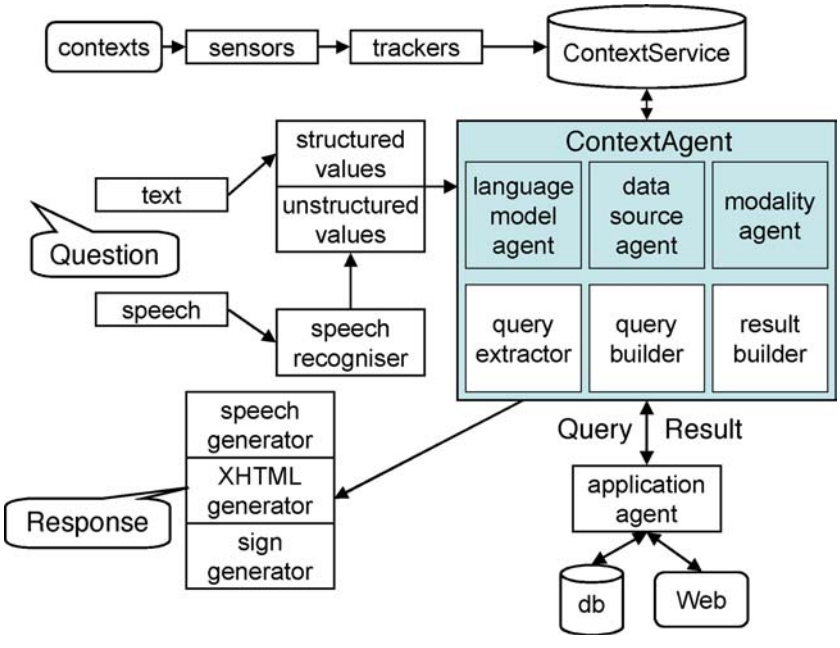


Fig. 8 Logical architecture for context-aware weather forecasting application

of the ContextAgent is to use contexts to respond to users in the most appropriate way.

First, user contexts are sensed by context trackers and stored in the ContextService. When a user asks a query, the language agent extracts structured values and unstructured text from it in order to understand the query category (e.g. weather forecast, hotel booking or flight information) and send the query to the question extractor. The question extractor extracts useful information from the raw query and passes it to the query builder, which enhances it with relevant user preferences, e.g. individual, group or default.

The query builder passes the query to the data source agent, which sends it to the relevant application agent. The application agent formats the query in a form suitable for its sources (e.g. SQL, XQuery), dispatches it, receives the response(s) and returns the result to the ContextAgent. Once the ContextAgent has the result and context information, the result builder refines the result (e.g. by removing unwanted information), formats it and delivers it to the user with the modality best suited to the user's preferences, modified by current constraints such as device availability. In a fully developed application we would anticipate using additional context sources covering spatial and temporal location, and users' activity patterns.

6.2 Context retrieval

We have described the acoustic environment tracker and ContextService in detail in Sects. 4 and 5. Our approach here is to plug the acoustic environment tracker into the context infrastructure and use the acoustic environment as a constraint in a modality agent.

The acoustic environment tracker periodically obtains environmental noise and adds an element in the ContextService. This may take the form of a detailed noise classification element as in Fig. 7, or a simple location classification as shown in Fig. 9. The modality agent retrieves the user's current contexts from the ContextService and creates a user-context description.

6.3 Input modalities and query processing

We provide both voice and HTML interfaces in the system to allow the user to submit queries. For the HTML interface, a user can open a browser and type in the query. For the voice interface, user may speak naturally to the system. We use VoiceXML, implemented on the IBM voice server SDK [52], to deal with voice queries. In our weather forecast exemplar, the application can accept questions about general weather forecasts, as well as information on temperature, rain, wind, humidity, sunrise, etc.

```
<contextElement timestamp="2004-08-01T12:37:26"
privacy="public">
  <subject>8a655bbfde7491a5d98f2d3f6018640d</subject>
  <predicate>location.acousticEnvironment</predicate>
  <object type="java.lang.String">office</object>
</contextElement>
```

Fig. 9 ContextElement example

We attempt to understand all in-domain queries and do not constrain the user at any point in the dialogue. We currently use a JSGF [53] which accepts general weather queries like: "What will the weather be in Chichester on Sunday?", or more specific queries like: "What's the chance of hail tonight in Wellingborough?", or shorter phases, such as "weather in London tomorrow?", or "weekend Nottingham fog?". The vocabulary currently used by the system contains 474 words, which include 350 city and locality names in UK; 21 time terms e.g. "tomorrow", "Friday", "weekend"; 37 weather terms, e.g. "rain", "snow", "temperature"; 66 general terms, e.g. "please", "chance", "what", "how" etc. The weather terms in this model are based on a set of terms arrived at following discussions with local weather forecasters.

Further development of the language model requires a suitable corpus of real user queries in various modalities. In the absence of such a corpus one of the best available sources is the Excite 2001 query corpus of one million queries (described in [54, 55]). It contains 1,307 distinct queries containing weather terms from our language model, of which 28% are of the form <weather term, place>, 21% are looking for a service (e.g. CNN Weather Channel), and 43% are queries implicitly requesting local weather forecasts; the remainder are mostly searches for weather records.

The place, time and weather terms are extracted from the query by the question extractor. The current version can deal with a variety of variant names and misspellings. It has a comprehensive list of weather terms and their mapping into standard forecasting terminology, and a default which is used if no specific weather types are contained in the request. The recognition of temporal phrases is limited to those covered by the scope of the WeatherQuest database, defaulting to "today" if nothing else is recognised. These are subsequently converted into dates. Place names are recognised by, first, trying to match the place exactly in the list of known places, second, trying to match elements of multiple word place names, and finally to look for a close match (using minimum edit distance) between target words and known places. If no good match is found, a default of "here" is used. "Here" is the user's current location determined by GPS, or their home city if no other location information is available. This information can be obtained from the ContextService. The city or other location information is mapped into a forecast locality to query the forecast database.

The location coordinates, forecast period and weather terms then go into the query builder to generate a SQL statement for querying the forecast database or other statements for querying Internet sources.

6.4 Modality agent

In order to select the best output modality, the modality agent requests a set of user contexts which is related to the query. The current version of the agent uses a simple rule-based strategy for matching the different constraints in order to find the best output modality. The rules serve as a model that provides a detailed description for different contexts. The rules identify four high-level categories of contexts: the user preferences, the input modality, the device and the user's activity. Activity is inferred from the acoustic environment, location and other context information, particularly location.

The user-preference rule defines the possible output modality according to user preference, for instance, a deaf user who requires sign language as output. In this case the only allowed output is sign language. If a user's preference is for graphic output, we give graphic the first priority, and also allow text and voice.

In many situations, the static user preference cannot satisfy the user's current needs. For example, a user's preference is text but he/she is driving, or the preference is voice but he/she is in a conversation. In such cases, other constraints need to be applied.

The input modality indicates whether the query modality is voice or text. If it is voice, that is given the first priority, we also allow graphic and text. If it is text, we give the output order as: graphic, text, voice.

The device also gives constraints for the output format. At a high level, we classify devices as either a PC or PDA. Then lower-level device attributes such as screen resolution can be retrieved as necessary from the ContextStore.

The user's current activity is an important constraint which can override user preferences and input modality. A user's current activity, as inferred from the acoustic environment is subject to frequent changes. Focus is given on taking into account some pre-defined constraints and working to optimise output. For instance, if the user's environment is a car, we assume the user is driving, so the only allowed output modality is voice. If the user's environment is in a lecture we allow graphic or text output modalities, but not voice.

The modality agent selects the best output format depending on user's preference, input modality, device and current activity. The selection is applied on the available output modalities and their priorities based on the rules.

6.5 Data source agent

The data source agent is a mediator which directs the generated query to specific information sources—in this example weather forecasts. There are many public and commercial sources of weather forecast information available at granularities from national synoptic forecasts to specialised local forecasts of particular facets of the weather, covering periods from a few hours to seasons. We are working with WeatherQuest to explore a range of novel service prototypes. WeatherQuest is a startup company from the University of East Anglia School of Environmental Sciences which provides specialist forecasting services in the UK, but does not currently offer comprehensive coverage of the country. For places not covered by WeatherQuest forecasts we supply a general synopsis from the publicly available UK Meteorological Office forecast. The WeatherQuest database can provide weather forecasts for much of the UK for up to 6 days, and can provide a wide range of weather properties such as temperature, wind speed, humidity, rain type, sunrise, etc. Currently, if the locality is not covered by WeatherQuest, the national synoptic forecast is provided as a default response.

```
<?xml version="1.0" encoding="UTF-8"?>
<dataroot xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-
instance">
  <Forecast>
    <Rain_Day_type>drizzle</Rain_Day_type>
    <Daynum>2</Daynum>
    <Wind12dirn>WSW</Wind12dirn>
    <Rain_Day_pct>30</Rain_Day_pct>
    <Max_temp>15.0</Max_temp>
    <Sunshine>2</Sunshine>
    <Cloud_cover>5.0</Cloud_cover>
    <Forecast_outlook>Becoming brighter with sunny periods and
          scattered showers, colder with fog and night frosts later.
    </Forecast_outlook>
    <Wind12spd>18</Wind12spd>
    <Date>2003-11-19 00:00:00</Date>
    <Min_temp>7.0</Min_temp>
    <Wind12stb>3</Wind12stb>
    <Square>217</Square>
  </Forecast>
</dataroot>
```

Fig. 10 XML output for example query

```
<gml>
  <gmlsign>
    <lexsign>
      <mocapdata filename="BSL_Tomorrow"/>
      <mocapspeed percent="100"/>
    </lexsign>
  </gmlsign>
  <gmlsign>
    <lexsign>
      <mocapdata filename="BSL_From_West_South_West"/>
      <mocapspeed percent="100"/>
    </lexsign>
  </gmlsign>
  <gmlsign>
    <lexsign>
      <mocapdata filename="BSL_Light_Rain"/>
      <mocapspeed percent="100"/>
    </lexsign>
  </gmlsign>
</gml>
```

Fig. 11 SiGML fragment for example query

6.6 Output modalities

The query result is directly fed into an XML weather forecast generator, which outputs just the information that the user requested (Fig. 10) along with an XML schema to document it. A set of XSLT scripts provides multiple interfaces to the weather forecast report; currently we support HTML, plain text (which can be sent via email), VoiceXML, WML and British Sign Language (BSL), using a SiGML prototype. The signing system also supports output in Dutch and German sign languages.

HTML pages are available in a format which uses an avatar to display the weather information as sign language and text. These pages are produced using a template which is supplied with the appropriate text and a list of motion capture files or SiGML content (Fig. 11). This is implemented through an XML template to fetch the motion capture files we need; this requires the client side to have installed the signing avatar and relevant motion files (Fig. 12). The avatar extends previous work, described elsewhere [56, 57]. Planned developments will provide the motions as XML inside an HTML page, avoiding the need to download the motion capture files to a client machine.

7 Conclusions

In this paper we have summarised the results of our experimental work in environmental noise and sound event classification and recognition. We have described a model and implementation of a mobile context-aware framework and have shown how we can use this to capture a variety of context information. We have illustrated this with an experimental implementation of a multimodal weather forecast information system. Our initial experiences indicate that there are a number of open issues concerning user behaviour capture and representation, inference from context information to query and output modification, and learning from feedback to modify the future behaviour of the system.

The contribution of this work is to demonstrate the feasibility and desirability of using a general framework for reporting and maintaining a wide variety of context information that can be exploited to improve both the questions and responses from conventional systems. Our experimental results show the feasibility of capturing environmental noise using current PDAs, although there are several implementation issues that need to be addressed before integrated systems can be deployed.

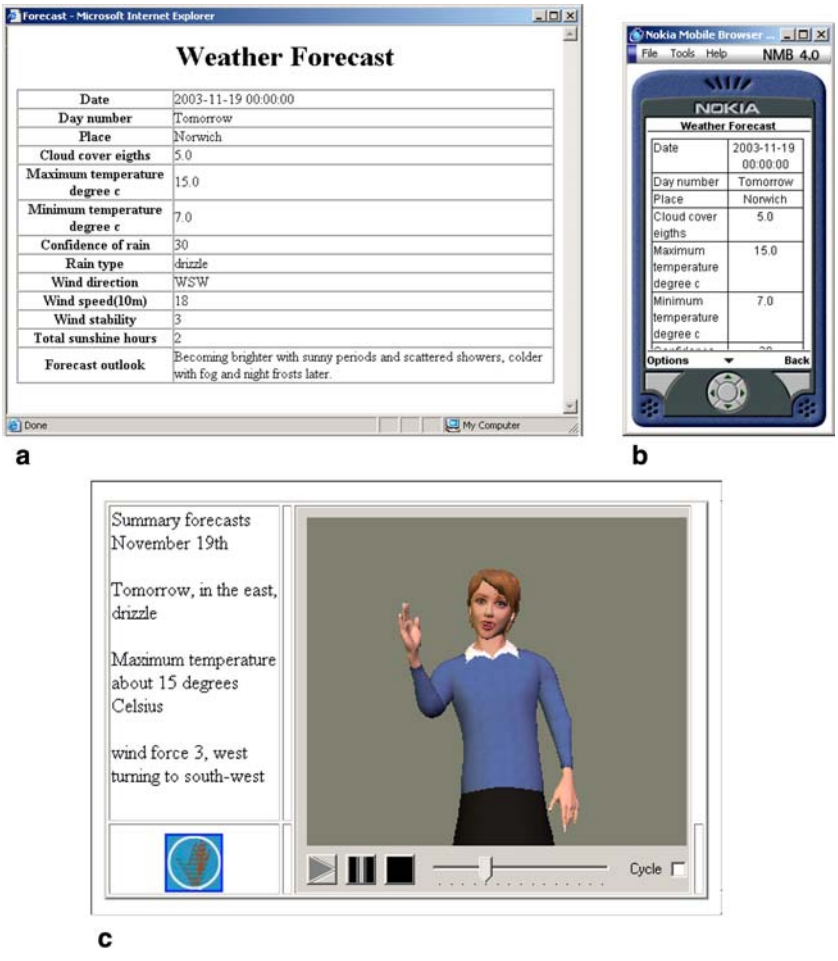


Fig. 12 Example output: **a** HTML, **b** WML, **c** SiGML (weather forecast data supplied by WeatherQuest)

The work described above shows that the acoustic environment is a rich source of context information which can be recognised with a high degree of accuracy and can be used as a good indicator of current activity.

Acknowledgements We wish to thank, first, the VisiCast project team at UEA, particularly Judy Tryggvason, for the signing output, and Mike Lincoln for discussions on the language model, second, Steve Dorling and others from WeatherQuest, and, third, the anonymous reviewers whose comments have been much appreciated in revising this paper.

References

1. Ma L, Smith DJ, Milner BP (2003) Context awareness using environmental noise classification. In: Proceedings of the Eurospeech 2003, Geneva, Switzerland, pp 2237–2240
2. Ma L, Smith DJ, Milner BP (2003) Environmental noise classification for context-aware applications. In: Proceedings of the DEXA 2003, (LNCS 2736), pp 360–370
3. Moran TP, Dourish P (2001) Introduction to special issue on context-aware computing. Human-Comput Interact 16(2–4):87–94
4. Dey AK, Salber D, Abowd GD (2001) A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. Human-Comput Interact 16(2–4):97–166
5. Want A, Hopper A, Falcao V, Gibbons J (1992) The active badge location system. ACM TOIS 10(1):91–102
6. Schilit BN, Adams NI, Want R (1994) Context-aware computing applications, workshop on mobile computing systems and applications. Santa Cruz, pp 85–90
7. Dey AK (2001) Understanding and using context. Pers Ubiquit Comput 5(1):4–7
8. Dey AK, Abowd GD (2000) Towards a better understanding of context and context-awareness. In: CHI 2000 workshop on the what, who, where, when, and how of context-awareness
9. Anhalt J, Smailagic A, Siewiorek DP, Gemperle F, Salber D, Weber SM, Beck J, Jennings J (2001) Toward context aware computing: experiences and lessons. IEEE Intell Syst 16(3):38–46
10. Brown PJ, Bovey JD, Xian C (1997) Context-aware applications: from the laboratory to the marketplace. IEEE Pers Commun 4(5):58–64
11. Abowd GD, Dey AK, Orr RJ, Brotherton J (1998) Context-awareness in wearable and ubiquitous computing. Virtual Real Soc Int J 3:200–211
12. Chen G, Kotz D (2000) A survey of context-aware mobile computing research, Department of Computer Science, Dartmouth College http://www.cs.dartmouth.edu/reports/TR2000–381/
13. Lieberman H, Selker T (2000) Out of context: computer systems that adapt to, and learn from, context. IBM Syst J 39(3–4):617–632
14. Dey AK, Salber D, Abowd GD, Futakawa M (1999) The conference assistant: combining context-awareness with wearable computing. In: Proceedings third international symposium on wearable computers
15. Yan H, Selker T (2000) A context-aware office assistant. In: ACM international conference on intelligent user interfaces (IUI-2000), New Orleans, pp 276–279
16. Abowd GD, Atkeson CG, Hong J, Long S, Kooper R, Pinkerton M (1997) Cyberguide: a mobile context-aware tour guide. Wireless Networ 3(5):421–433
17. Randell C, Muller H (2000) The shopping jacket: wearable computing for the consumer. Pers Technol 4(4):241–244
18. Sumi Y, Etani T, Fels S, Simone N, Kobayashi K, Mase K (1998) C-MAP: building a context-aware mobile assistant for exhibition tours. Social interaction and communityware, Kyoto, Japan (LNCS 1519), pp 137–154
19. Brown PJ (1996) STICK-E NOTES: changing notes and contexts—the SeSensor module and the loading of notes, EP-odd
20. Beigl M (2000) MemoClip: a location based remembrance appliance. Pers Technol 4(4):230–233
21. Dey AK, Abowd GD (2000) CybreMinder: a context-aware system for supporting reminders. In: Proceedings of the second international symposium on handheld and ubiquitous computing (HUC), pp 172–186
22. Rhodes B.(2003) Using physical context for just-in-time information retrieval. IEEE Trans Comput 52(8):1011–1014
23. Marmasse N,. Schmandt C (2000) Location-aware information delivery with comMotion. In: Proceedings of the second international symposium on handheld and ubiquitous computing (HUC), Bristol, pp 157–171
24. Sawhney N, Schmandt C (2000) Nomadic radio: speech and audio interaction for contextual messaging in nomadic environments. ACM ToCHI 7(3):353–383
25. Gerasimov V, Bender W (2000) Things that talk: using sound for device-to-device and device-to-human communication. IBM Syst J 39(3–4):530–546
26. Mynatt ED, Back M, Want R, Baer M, Ellis JB (1998) Designing audio aura. In: Proceedings of the CHI'98, New York, pp 566–573
27. Schmandt C, Marmasse N, Marti S, Shawhney N, Wheeler S (2000) Everywhere messaging. IBM Syst J 39(3–4):660–677
28. Hindus D, Schmandt C (1992) Ubiquitous audio: capturing spontaneous collaboration. In: Proceedings of the computer-supported cooperative work (CSCW). Toronto, pp 210—217
29. Huang X, Weng J, Zhang Z (2004) Office presence detection using multimodal context information. In: Proceedings of the international conference on acoustics, speech, and signal processing, Montreal
30. Foote J (1999) An overview of audio information retrieval. Multimedia Syst 7(1):2–11
31. Wold E, Blum T, Keslar D, Wheaton J (1996) Content-based classification search and retrieval of audio. IEEE Multimedia 3(3):27–36
32. Brown GJ, Cooke MP (1994) Computational auditory scene analysis. Comput Speech Lang 8:297–336
33. Couvreur C (1997) Environmental sound recognition: a statistical approach. PhD thesis, Faculté Polytechnique de Mons, Belgium
34. Gaunard P, Mubikangiey CG, Couvreur C, Fontaine V (1998) Automatic classification of environmental noise events by Hidden Markov Models. Appl Acoust 187–206
35. Nishiura T, Nakamura S, Miki K, Shikano K (2003) Environment sound source identification based on Hidden Markov Model for robust speech recognition. EuroSpeech 2157–2160
36. Peltonen V, Tuomi J, Klapuri A, Huopaniemi J, Sorsa T (2002) Computational auditory scene recognition. In: Proceedings of the international conference on acoustic, speech and signal processing, Orlando
37. Sawhney N (1997) Situational awareness from environmental sounds technical report for modeling adaptive behavior (MAS 738), Pattie Maes, MIT Media Lab
38. Clarkson B, Sawhney N, Pentland A (1998) Auditory context awareness via wearable computing. In: Workshop on perceptual user interfaces, pp 37–42
39. Clarkson B, Pentland A (1999) Unsupervised clustering of ambulatory audio and video. In: Proceedings of the international conference on acoustics, speech, and signal processing, Phoenix
40. Ellis DPW (1999) Using knowledge to organize sound: the prediction-driven approach to computational auditory scene analysis and its application to speech/nonspeech mixtures. Speech Commun 27(3–4):281–298
41. The HTK Book (2001) Version 3.1, Cambridge University Engineering Department, http://htk.eng.cam.ac.uk
42. Vinsensius Vega SB, Bressan S (2003) Continuous naive Bayesian classifications, ICADL, pp 279–289
43. Java Sound API http://java.sun.com/products/java-media/sound

44. Ryan NS, Osbakk P (2003) The MobiComp Infrastructure, http://www.cs.kent.ac.uk/projects/ubi/infra/mobicomp/
45. Pascoe J, Morse DR, Ryan NS (1998) Developing personal technology for the field. Pers Technol 2:28–36
46. Ryan NS, Pascoe J, Morse DR (1999) Fieldnote: a handheld information system for the field. In: Laurini R (ed) Proceedings of the TeleGeo'99, 1st international workshop on TeleGeo-Processing, Lyon, pp 156–163
47. van Leusen M, Ryan NS (2001) Educating the digital fieldwork assistant. In: Burenhult G (ed) Proceedings of computer applications and quantitative methods in archeology conference (CAA 2001:), Gotland, pp 401–412
48. Ahuja S, Carriero N, Gelertner D (1986) Linda and friends. IEEE Comput 19(8):26–34
49. Gelertner D, Carriero N (1992) Coordination languages and their significance. CACM 32(2):96–107
50. Johanson B, Fox A (2002) The event heap: a coordination infrastructure for interactive workspaces. In: Proceedings of the 4th IEEE workshop on mobile computer systems and applications (WMCSA-2002), Callicoon, pp 83–93
51. XQuery (2004) 1.0 An XML query language, W3C working draft, 23 July 2004, http://www.w3.org/TR/xquery/
52. IBM WebSphere Voice Server Administrator's Guide, for information on deploying VoiceXML applications in a Voice over IP environment
53. Java speech grammar format (JSGF) Specification (2001) http://java.sun.com/products/java-media/speech/forDevelopers/JSGF/index.html
54. Ozmutlu S, Ozmutlu HC, Spink A (2002) Trends in multimedia web searching: excite queries. In: Proceedings of the ITCC 2002, pp 40–45
55. Ozmutlu S, Spink A, Ozmutlu HC (2004) A day in the life of web searching: an exploratory study. Inform Process Manage 40(2):319–345
56. http://www.visicast.cmp.uea.ac.uk/Visicast_index.html
57. Verlinden M, Tijsseling C, Frowen H (2002) A signing avatar on the WWW, Gesture and sign languages in human-computer interaction, pp 169–172