MIT Libraries | DSpace@MIT

MIT Open Access Articles

An implementation of auditory context recognition for mobile devices

The MIT Faculty has made this article openly available. ***Please share*** how this access benefits you. Your story matters.

Citation: Perttunen, M. et al. "An Implementation of Auditory Context Recognition for Mobile Devices." Mobile Data Management: Systems, Services and Middleware, 2009. MDM '09. Tenth International Conference on. 2009. 424-429. © 2009 IEEE.

As Published: http://dx.doi.org/10.1109/MDM.2009.74

Publisher: Institute of Electrical and Electronics Engineers

Persistent URL: http://hdl.handle.net/1721.1/60225

Version: Final published version: final published article, as it appeared in a journal, conference proceedings, or other formally published context

Terms of Use: Article is made available in accordance with the publisher's policy and may be subject to US copyright law. Please refer to the publisher's site for terms of use.

Massachusetts Institute of Technology

2009 Tenth International Conference on Mobile Data Management: Systems, Services and Middleware

An Implementation of Auditory Context Recognition for Mobile Devices

Mikko Perttunen[1], Max Van Kleek[2], Ora Lassila[3], Jukka Riekki[1]

[1]Department of Electrical and Information Engineering
University of Oulu
Oulu, Finland
{first.last}@ee.oulu.fi

[2]Computer Science and Artificial Intelligence Laboratory
MIT
Cambridge, MA, USA
emax@csail.mit.edu

[3]Nokia Research Center Cambridge
Cambridge, MA, USA
ora.lassila@nokia.com

***Abstract*—Auditory contexts are recognized from mixtures of sounds from mobile users' everyday environments. We describe our implementation of auditory context recognition for mobile devices. In our system we use a set of support vector machine classifiers to implement the recognizer. Moreover, static and runtime resource consumption of the system are measured and reported.**

Keywords: pattern recognition, classification, pervasive computing

I. Introduction

In addition to speech, humans routinely use other sounds emitted from mundane things happening around them in everyday environments [1]. It has been shown that they base their analysis on recognizing the sources of the sounds in the audio entering the ears. The combination of different sources produces a scene, for example, a person riding a bus with people chatting around her. The study of processing and understanding these audio signals by computers is called computational auditory scene analysis (CASA) [2].

It has been suggested that recognizing the overall audio scene could be useful in context-aware systems that react to changes in their usage environment [3-6]. Peltonen calls the classification of the mixtures of audio themselves into predefined classes, without trying to recognize the sources of audio, computational auditory scene recognition (CASR) [4].

Importantly, analyzing the auditory scene is unobtrusive and does not require extra instrumentation; microphones in mobile devices can be used to record excerpts of audio. Moreover, as opposed to most positioning systems, auditory scenes can be analyzed without using a fixed network infrastructure.

The main contributions of this paper are the design and implementation of auditory context recognition for mobile devices, and an evaluation of the resource consumption thereof.

The rest of this paper is organized as follows: In the next section we describe the design of the system. Section 3 goes through our experiments and results. Section 4 is about ongoing and future work. In section 5 related work is reviewed. Finally, in section 6 we summarize.

II. Design and Implementation

In this section we describe an implementation of auditory context recognition system for Symbian OS (version 9.1) mobile phones and for the Linux based mobile computing platform Maemo (http://maemo.org).

A. Architecture

The system components of our basic scene recognition application are shown in Fig. 1 and described in the following subsections. As shown in Fig. 2, the system uses a (microphone specific) threshold to discriminate between silence and all other auditory contexts. When the threshold is not exceeded, the more complex spectral features need not be calculated.

B. Feature Extraction

Feature extraction is done from 3s segment of audio (8kHz, 8bit). Stäger et al. [7] showed that the on-time of sensors dominate power consumption of audio and accelerometer based recognition. In our previous work [8], we showed that for the Ma et al. dataset [5], recognition accuracy is not improved significantly for segments longer than 3s. For these reasons, we fixed the segment duration to 3s in our experiments.

The segment is windowed using a Hamming window into 512-sample frames with an overlap. We used 0, 128, 256, and 384 sample overlaps in the

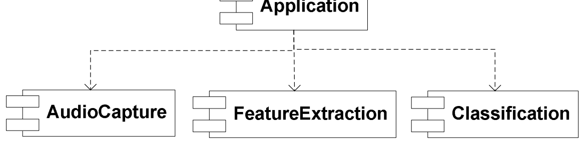

Fig. 1. System components

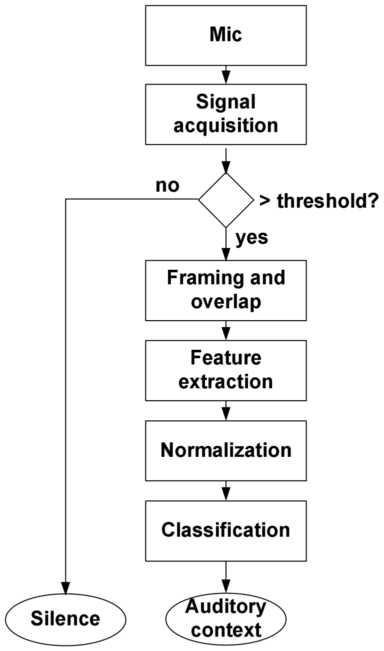

Fig. 2. System flowchart

experiments described in the following sections. Thus, FFT is extracted from 64ms windows. Then, triangular bandbass filters are applied to yield the Mel spectrum. Using the 40 Mel spectrum coefficients, 13 MFFCs for the current frame are determined through DCT. To get the final features, the MFCCs are averaged over all the frames in the segment through

$$\hat{f}_m = \frac{1}{K}\sum_{i=0}^{K} C_m^i, 0 \le m \le L-1 \tag{1}$$

where $\hat{f}_m$ is the mth averaged MFCC and K is the number of frames in the segment. Similarly, the Mel spectrum coefficients (E_j) are averaged by

$$\hat{f}_j = \frac{1}{K}\sum_{i=0}^{K} E_j^i, 0 \le j \le J-1 \tag{2}$$

where J=40, to yield the averaged Mel spectrum over the segment length. Both of these feature vectors are then normalized independently using

$$F_x^i = \frac{\hat{f}_x^i - \hat{f}_x^{min}}{\hat{f}_x^{max} - \hat{f}_x^{min}} \tag{3}$$

where $\hat{f}_x^i$ is the ith element (e.g. MFCC0) of a averaged spectral feature vector over the segment length, and denotes the final normalized feature value. Here, x refers to the outputs either of (1) or (2). The above formulas are adapted from [9].

As mentioned in [9] averaging has the advantage that the final feature set of a segment always has the same size, regardless of the segment length. This allows experimentation with different segment lengths in the case of SVMs without the feature set growing proportional to segment length. Intuitively, averaging has the effect of making the segment features look the more distinctive of the environment they are extracted from, the longer the segment length. For example, a random short sound might be averaged off. However, also a short but distinctive sound might vanish.

We do not use features directly related to audio volume; when deploying auditory context recognition to practical usage situations, the audio signal energy captured from semantically same contexts varies. This could be due to a phone being carried in a hand versus in a bag. In our previous experiments we quantified this through evaluating recognition accuracy of energy-dependent feature vectors against a dataset with and without applying gain [8].

The feature extraction implementation is based MeapSoft[1] , the Mel spectrum and MFCC extraction implementations of which we converted to ANSI C.

C. *Classification*

We utilize support vector machines [10] to implement the auditory context recognizer. As LibSVM [2] [11] version 2.85 is used to train and implement our support vector machine based auditory context recognizers, the "one-against-one" [10] multi-class classification scheme is applied. In this scheme k(k-1)/2 SVMs are trained and classification is done through voting: each classifier output is considered a vote and the predicted class is determined through maximum number of votes.

To be able to run classification on our target devices, we first ported and built the feature extraction and classifier subset of the LibSVM [11] from C++ to ANSI C, and then used the compiler settings for the targets devices as described in Section 4.1.

As with our previous investigation, models were pre-trained on our workstation using dataset 1 from Ma et al. which constituted 5-minute recordings of 12 classes each, and tested on another set of 5-minute recordings (dataset 2).

LibSVM was set to produce a classifier of type "C-SVC" [11, 12], with probability estimation enabled. For RBF kernel the regularization parameter and gamma were tuned using values from 0.1 to 1000 with a decade step. For polynomial kernel, kernel degrees from 1 to 4 were tried in addition, resulting in a three dimensional

[1] http://labrosa.ee.columbia.edu/meapsoft/
[2] http://www.csie.ntu.edu.tw/~cjlin/libsvm

grid of parameter values. The resulting models were then transferred to each of the devices, and loaded into our test application.

D. *Test Aapplication*

The test application, shown in Fig. 3, was built to allow a simple investigation of how pre-trained audio scene classifiers would work on real devices in real usage situations "in the field". To enable controlled analysis, we additionally incorporated the facility to keep captured audio for later analysis, as well as to "re-play" e.g., simulate previously recorded audio against the classifier on each device.

III. EXPERIMENTS AND RESULTS

A. *Platform Details*

The devices we chose were a standard smartphone and a handheld linux-based touchscreen Internet tablet. The smartphone is Nokia E61 (http://www.forum.nokia.com/devices/E61) with 21MB free RAM and ARM 9 CPU at 220 Mhz, running Symbian OS 9.1 with S60 3rd edition. Binaries were built without debug information.

The internet tablet was a Nokia N800 with 128 MB RAM and 256 MB internal flash memory and an TI OMAP2420 ARM 11 CPU at 400Mhz running Internet Tablet OS 2008 edition. For compiling the code, we used the scratchbox build environment and gcc. To optimize the code and to use floating point unit on N800, the optimisation flags *mfpu=vfp, mfloat-abi=softfp, and mcpu=arm1136j-s* were used.

B. *Static Measures*

Table I shows the dependency of recognition accuracy on the window overlap used in feature extraction. LibSVM models with 3s segment duration using Mel spectrum (40 features) or MFCC (13 features) as features were used. For RBF kernel the regularization parameter and gamma were tuned using values from 0.1 to 1000 with a decade step. For polynomial kernel, kernel degrees from 1 to 4 were tried in addition, resulting in a three dimensional grid of parameter values. For both kernel types, the kernel providing the best accuracy is reported. Accuracy was determined by training on dataset1 and testing against dataset2 of Ma et al. [5]. All models were trained to output probability estimates. Although accuracy does not increase monotonically with increasing overlap it can be seen that either 128 or 256 sample overlap is suitable. This is significant for runtime characteristics; we chose to use 128 sample overlap in our runtime experiments, described in the next section.

From Table II it can be seen that the resulting model file sizes, varying from 80kb to 210kb, lend themselves to be loaded into modern mobile devices. Moreover,

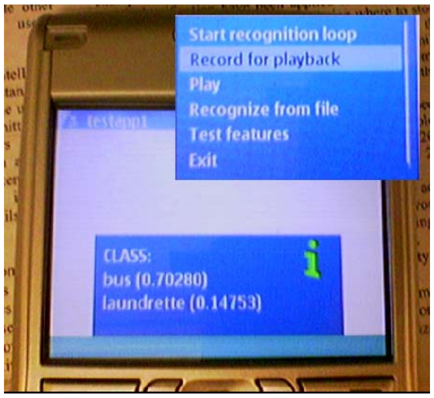

Fig. 3. Test application on a Symbian phone (Nokia E61). The figure shows the two most likely auditory contexts for the processed audio segment.

using MFCC features produces models with more support vectors, but as each vector is shorter than in the case of Mel spectrum features, the model sizes are smaller.

The file size of executable (including the test application) on N800 varies from 80kb to 100kb with the different build options. On Symbian, the DLLs of both feature extraction and classifier are a bit over 10kb. In the next section we study runtime characteristics of these models.

C. *Runtime Performance*

Both feature extraction and the classifier implementations use floating-point calculations. According to the Symbian documentation[3] floating-point calculations are multitudes of slower than integer calculations on a device with no floating-point unit.

Thus, we wanted to evaluate the runtime performance on a device with and without a floating-point unit. On the Nokia N800 device we run tests both with and without using floating-point unit, and compared these results with the corresponding performance on a Symbian phone without a floating-point unit, namely Nokia E61. We note that our code assumed floating point capability, and that if we had converted it manually to use only fixed-point arithmetic, our results would likely be different.

The runtime measurements were done using timing and memory monitoring facilities built into each device's operating system platform. We averaged each metric over ten runs for our reported results. For classification, only the time taken to produce a prediction is measured; that is, the time taken to load model into memory was not measured, because that needs to be done only once. However, it can be said that

[3] http://www.symbian.com/developer/techlib/v9.1docs/doc_source/index.html

our test application produces no noticeable delay at startup, when the model is loaded.

It should also be noted that because in our scheme the length of feature vector is independent of the analyzed segment length, runtime resource consumption of classification is independent of the segment duration, respectively. This is not the case when using a HMM-based classifier (as in e.g. [5, 6]), because the features extracted from one window constitute an element in the HMM input sequence.

Table III and Table IV show the results for the E61 and N800 devices. The use of the floating point unit of this device can be enabled and disabled with build flags of the Maemo SDK, causing alternative instruction sets to be used. The measurements without using floating point unit are shown in parenthesis. Comparing the results in these tables, it can be seen that floating point unit of the N800 device contribute a large difference in running times. As the tables show, the feature extraction takes far longer than SVM prediction (classification).

Table IV shows that using floating point unit on N800, the feature extraction and classification takes about 0.3s combined for polynomial kernel with 40-element Mel spectrum as features. This is fast enough for near real time operation, because recording audio and feature extraction can be done in parallel. Thus, the latency of recognition equals the duration of the segment (here, 3s). The same task takes about 1.5s on the Symbian device (Table III); latency is still clearly equal to the segment length.

From Table IV it can be also noticed that using our system, MFCC calculation consumes about double the time that Mel spectrum determination takes. Due to this, and due to the fact that the running time of classification is dominated by the feature extraction, using Mel spectrum instead of MFCC features seems to be more efficient.

IV. Ongoing and future work

The work described in this paper is part of our ongoing effort at finding methods for sensing and extracting context that can be directly applied to make applications more aware of their user and environment using readily available hardware and software platforms today. Initially, we collected a small database of recordings from our typical auditory environments for training and testing classifiers using LibSVM. The auditory contexts included subway, office, walking on a crowded street, and home with TV on. Additionally, a

TABLE I. Dependency of recognition accuracy on window overlap in feature extraction.

	RBF		Polynomial	
window overlap	*MFCC*	*Mel*	*MFCC*	*Mel*
0	78.5%	85.7%	79.6%	85.4%
128	86.8%	93.6%	87.1%	93.5%
256	87.6%	95.5%	88.5%	95.3%
384	88.6%	94.5%	89.6%	94.8%

TABLE II. Detailed model information for the models produced using 128 sample overlap, corresponding to Table I

	RBF		Polynomial	
	MFCC	*Mel*	*MFCC*	*Mel*
#training examples	1200	1200	1200	1200
#support vectors	468	380	415	329
model size (kb)	110	210	80	180
kernel degree	-	-	4	2
gamma	1	0.1	0.1	0.1

TABLE III. Runtime measurements on Nokia E61 (3s segment, 8bit, 8000Hz). This device does not have a floating point unit

	RBF		Polynomial	
	MFCC	*Mel*	*MFCC*	*Mel*
Feature extraction (s)	3.70±0.01	1.39±0.01	3.70±0.01	1.39±0.01
Classification (s)	0.12±0.00	0.14±0.01	0.11±0.00	0.12±0.00

TABLE IV. Runtime measurements on Nokia N800 (3s segment, 8bit, 8000Hz)

	RBF kernel		Polynomial kernel	
	MFCC	*Mel*	*MFCC*	*Mel*
	with fpu (without fpu)	*with fpu (without fpu)*	*with fpu (without fpu)*	*with fpu (without fpu)*
Feature extraction (s)	0.59±0.01 (2.44±0.03)	0.28±0.02 (0.72±0.01)	0.59±0.01 (2.44±0.03)	0.28±0.02 (0.72±0.01)
Classification (s)	0.02±0.01 (0.07±0.01)	0.02±0.01 (0.06±0.01)	0.02±0.00 (0.05±0.01)	0.02±0.01 (0.06±0.01)
Peak memory (MB)	4.84 (4.84)	4.84 (4.84)	4.80 (4.80)	4.80 (4.80)

separate SVM was trained to discriminate between music and speech. With this dataset, we were able to perform simple ‘action research’, that is, we carried the mobile phone running our recognizer application in those environments where the recordings were from and observed its functioning. As these tests were informal, it can only be mentioned that environments such as riding subway, office, and street with traffic and people seemed to be well recognized in practice. As a next step we plan to measure the power consumption of our implemented system.

We have also begun to integrate the auditory context recognizer with our user modeling framework [13] to collect more comprehensive data about user's computer activity.

In addition to the above work, we are planning a comparative study of the effects of microphone characteristics on scene classification performance. For this, we are in the process of collecting a dataset using a variety of microphones, both embedded in consumer devices and professional-grade audio equipment, such as laptops, palmtops and mobile phones, Bluetooth headsets, and studio-quality field recording gear.

Finally, we aim to improve our scheme by utilizing temporal smoothing over a longer duration, perhaps using heuristic rules or HMMs to model the possible transitions between contexts, to filter out unlikely context changes from the recognizer output.

V. Related work

We limit the content of this section on research on auditory scene and auditory context recognition and leave out other work on context-awareness. Perhaps the first attempt at recognizing auditory contexts is due to Sawhney [3]. In the work following his introduction the most relevant to ours is Ma et al., who classified 12 auditory contexts using a HMM-based classifier [5]. They achieved a reported 96% true positive accuracy by using 9-stage left-to-right HMMs. The authors used 12 MFCCs and a log energy term with their respective deltas, resulting in a 39-element feature vector.

Eronen et al. developed a HMM-based classifier for 28 auditory contexts, using a dataset they collected [6]. Eronen et al. performed also a study on the human perception of the same set of auditory contexts. This was done through listening tests, where the time taken to make a decision was recorded, and the resulting recognition accuracy compared to that of their system. In these experiments, 18 contexts were used as test inputs, and both the system and the test subjects had to choose from the full set of 28 contexts. The average accuracy for the human subjects was 69%, and the decision time averaged 14s. By comparison, their automatic system showing a steady increase in recognition accuracy until 20s, and a plateau of 72% accuracy at 60s.

Korpipää et al. use naïve Bayes classifiers to recognize 7 auditory contexts [14] from 1s segments. However, they used temporal smoothing (majority voting) in a 5s window to improve the classification results. This yielded about 90% accuracy.

In [15], Lu et al. apply support vector machines (SVMs) for classifying among four classes: non-pure speech, pure speech, background sound, demonstrating 80% to 96% accuracy from 0.1 to 1 second duration audio segments, respectively. Like our approach presented in this paper, they derive a single set of MFCC based features (means and variances) for each segment, instead of treating the problem as sequence classification task using HMMs.

Ward et al. [16] classified 9 workshop activities (e.g. drilling and sawing) using sound and acceleration. Using sound only, their system achieved 95.6% accuracy for user-dependent and 77.2% accurady for user-independent training. The authors also evaluated segmentation methods for continuous workshop audio.

Stäger et al. [7] studied the tradeoff between recognition accuracy and power consumption. A wearable device containing both accelerometers and a microphone were used in the study. Different methods and parameters were tested for feature extraction and classification. The authors found that through parameter optimization power consumption may be reduced by a factor of 2-4, causing only a slight degradation in recognition performance. A particularly interesting finding was the large difference in power consumption of a mic (<0.5mW) and an accelerometer (~3mW) in the sampling stage of a typical setup. Moreover, the various feature sets did not contribute remarkably to the overall power consumption, which was dominated by on-time of sensors and the length of FFT sample array. A limitation of the study is that it considers only power consumption, whereas key contributors such as memory consumption and model size were not thoroughly analyzed.

VI. Summary

In this paper we have described an implementation of auditory context recognition for mobile devices and analyzed its resource consumption. The results show that static model sizes of SVM-based auditory recognition system are not restrictive for typical mobile devices. Moreover, the runtime memory consumption and computational load are manageable. The results also demonstrate that, while MFCCs are often used for this task, the use of the Mel spectrum features seems to provide better performance both in terms of resource consumption and in terms of accuracy. Classification ran in almost negligible time for all used models. These results may be used as a basis when selecting components for such systems, and as a baseline for comparison in any future studies using other devices or variations of the recognition system.

Acknowledgment

The authors would like to thank the Nokia Research Center. The first author would like to thank the Fulbright - Technology Industries of Finland and the Finnish Graduate School in Electronics, Telecommunication, and Automation (GETA).

REFERENCES

[1] A.S. Bregman, Auditory Scene Analysis: The Perceptual Organization of Sound. Cambridge, MA: The MIT Press, 1994.

[2] D. Wang and G.J. Brown, Computational Auditory Scene Analysis - Principles, Algorithms, and Applications. Wiley-IEEE Press, 2006.

[3] N. Sawhney, "Situational Awareness from Environmental Sounds," url: http://web.media.mit.edu/~nitin/papers/Env_Snds/EnvSnds.html.

[4] V. Peltonen, J. Tuomi, A. Klapuri, J. Huopaniemi, and T. Sorsa, "Computational auditory scene recognition," Proc. Acoustics, Speech, and Signal Processing (ICASSP '02), IEEE Press, May 2002, pp. 1941-1944.

[5] L. Ma, B. Milner, and D. Smith, "Acoustic environment classification," ACM Trans.Speech Lang.Process., Vol. 3, 2, pp. 1-22, doi: 10.1145/1149290.1149292.

[6] 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," Audio, Speech, and Language Processing, IEEE Transactions on, Vol. 14, Jan 2006, pp. 321-329, doi: 10.1109/TSA.2005.854103.

[7] M. Stäger, P. Lukowicz, and G. Tröster, "Power and accuracy trade-offs in sound-based context recognition systems," Pervasive and Mobile Computing, Vol. 3, June 2007, pp. 300-327.

[8] M. Perttunen, M. Van Kleek, O. Lassila, and J. Riekki, "Auditory Context Recognition Using SVMs," Proc. Mobile Ubiquitous Computing, Systems, Services and Technologies (UBICOMM'08), IEEE Press, Oct. 2008, Valencia, Spain, pp. 102-108, doi: 10.1109/UBICOMM.2008.21.

[9] C. Lee, C. Chou, C. Han, and R. Huang, "Automatic recognition of animal vocalizations using averaged MFCC and linear discriminant analysis," Pattern Recognition Letters, Vol. 27, Jan. 2006, pp. 93-101, doi: 10.1016/j.patrec.2005.07.004.

[10] C.M. Bishop, Pattern Recognition and Machine Learning. Singapore: Springer, 2006.

[11] C. Chang and C. Lin, "LIBSVM: a library for support vector machines", 2001.

[12] C. Cortes and V. Vapnik, "Support-Vector Networks," Mach.Learn., Vol. 20, Sep. 1995, pp. 273-297, doi: 10.1023/A:1022627411411.

[13] M. Van Kleek and H. Shrobe, "A Practical Activity Capture Framework for Personal, Lifetime User Modeling," Lecture Notes In Artificial Intelligence; Vol. 4511, Springer-Verlag 2007, pp. 298-302, doi: 10.1007/978-3-540-73078-1_33.

[14] P. Korpipää, M. Koskinen, J. Peltola, S. Mäkelä, and T. Seppänen, "Bayesian approach to sensor-based context awareness," Personal Ubiquitous Comput., Vol. 7, July 2003, pp. 113-124, doi: 10.1007/s00779-003-0237-8.

[15] L. Lu, H. Zhang, and S.Z. Li, "Content-based audio classification and segmentation by using support vector machines," Multimedia Systems, Vol. 8, April 2003, pp. 482-492, doi:10.1007/s00530-002-0065-0.

[16] J.A. Ward, P. Lukowicz , G. Troster, and T.E. Starner, "Activity Recognition of Assembly Tasks Using Body-Worn Microphones and Accelerometers," Pattern Analysis and Machine Intelligence, IEEE Transactions on, Vol. 28, Oct. 2006, pp. 1553-1567, doi: 10.1109/TPAMI.2006.197.