Multimed Tools Appl (2016) 75:10441–10469
DOI 10.1007/s11042-015-2903-z

Efficient acoustic detector of gunshots and glass breaking

Martin Lojka[1] · Matúš Pleva[1] · Eva Kiktová[1] · Jozef Juhár[1] · Anton Čižmár[1]

Received: 28 August 2014 / Revised: 13 August 2015 / Accepted: 20 August 2015 /
Published online: 11 September 2015

Abstract An efficient acoustic events detection system EAR-TUKE is presented in this paper. The system is capable of processing continuous input audio stream in order to detect potentially dangerous acoustic events, specifically gunshots or breaking glass. The system is programmed entirely in C++ language (core math. functions in C) and was designed to be self sufficient without requiring additional dependencies. In the design and development process the main focus was put on easy support of new acoustic events detection, low memory profile, low computational requirements to operate on devices with low resources, and on long-term operation and continuous input stream monitoring without any maintenance. In order to satisfy these requirements on the system, EAR-TUKE is based on a custom approach to detection and classification of acoustic events. The system is using acoustic models of events based on Hidden Markov Models (HMMs) and a modified Viterbi decoding process with an additional module to allow continuous monitoring. These features in combination with Weighted Finite-State Transducers (WFSTs) for the search network representation fulfill the easy extensibility requirement. Extraction algorithms for Mel-Frequency Cepstral Coefficients (MFCC), Frequency Bank Coefficients (FBANK) and Mel-Spectral Coefficients (MELSPEC) are also included in the preprocessing part. The system contains

✉ Matúš Pleva
matus.pleva@tuke.sk

Martin Lojka
martin.lojka@tuke.sk

Eva Kiktová
eva.kiktova@tuke.sk

Jozef Juhár
jozef.juhar@tuke.sk

Anton Čižmár
anton.cizmar@tuke.sk

[1] Department of Electronics and Multimedia Communications, Technical University of Košice, FEI TU Košice, Park Komenského 13, 041 20 Košice, Slovak Republic

Cepstral Mean Normalization (CMN) and our proposed removal of basic coefficients from feature vectors to increase robustness. This paper also presents the development process and results evaluating the final design of the system.

Keywords Acoustic Event Detection · Weighted Finite-State Transducers · Continuous Monitoring of Large Urban Areas

1 Introduction

EAR-TUKE system was designed to be a lightweight system used for detection of breaking glass and gunshots. The gunshot detection in urban areas was the main task initiated in INDECT[1] project work package oriented on the intelligent monitoring and automatic detection of threats in urban areas. Detection, classification and localization (using Acoustic Vector Sensor) could be done in separate modules as it is described in [16, 17]. There have been similar gunshot detection systems already presented in the papers based on MFCC, LPC (Linear Prediction Coefficients) & HMM [7], GMM with MPEG-7 and MFCC feature vector reduced using floating search vector feature selection method [8] and [30], pitch range (PR) of non-speech sounds and the Autocorrelation Function using Support Vector Machines (SVM) with Gaussian kernel and Radial Basis Function Neural Network classifiers [29] and recently MFCC and three layer Artificial Neural Networks (ANN) classifier [28]. Their detection results are discussed and compared in the conclusion section.

This kind of system can be built using any available toolkit for processing input audio signal [5] and toolkit for classification [27] or simply using an all-in-one system, for example a speech recognition engine [12, 13]. These systems, although they are capable of classification of the acoustic events if their acoustic models are provided, tend to be unnecessarily complicated containing redundant functions that slow down the whole system [20]. Often, they are not easily modifiable for continuous monitoring of the input stream because of their input audio stream length limitation. Also, most of the systems require additional dependencies that are used during operation of the system or during compilation.

Our system is designed as a complete solution to the acoustic detection and contains a preprocessing part and a detection part together with the classification function. It is programmed entirely in C language for fast processing (all core processing, no C++ libraries called) and C++ language is only used for logical and easy to understand organization of the source code. The system is lightweight and made without any external dependency allowing easy compilation on different platforms. Additionally, the system utilizes weighted finite-state transducers (WFSTs) allowing easy extensibility [14]. This way, although the system was mainly designated for detection of breaking glass and gunshots, it is not restricted to this task alone. When needed, the system can be easily adapted to detect more types of acoustic events (an example of adjusted configuration is depicted in Fig. 14 which tells if an event should produce an alarm, and no recompiling is needed) or even to perform completely different task such as keyword spotting.

We developed the EAR-TUKE detection system taking into account our past research and development experience in acoustic events detection [15, 21] and automatic speech

[1] http://www.indect-project.eu/

recognition [25]. We have included new features for real-time special events monitoring, long-term stability, speed and memory profile enhancements in several iterations.

Compared to automatic speech recognition (ASR) task, this system does not need any language models nor language processing [9] (as required for automatic continuous speech recognition) but only very simple foreground / background sets described later. The ASR systems usually do not have to operate for a long time continuously (it could be restarted after completion of a sentence recognition), so their memory management and noisy background are not a big problem as described in [20]. The features used for ASR are mainly limited to MFCC and LPC coefficients, but for acoustic events more types should be considered as it is described later. In the end we have decided to build a completely new system which uses the HMM Viterbi decoding and WFSTs (which could also be used for ASR task, but this is the only similarity to ASR systems) and no third party code or ASR tool was used during the development.

The result of our effort is described in the beginning of the paper with a general overview of the system followed by detailed description of its parts. Finally, at the end of the paper, the results of the detection accuracy are presented along with the development description.

2 EAR-TUKE overview

Our effort, as it is stated in the introduction section, was to create an acoustic event detection system that is not overcomplicated, easy to use and lightweight. This effort was rewarded by the design of EAR-TUKE system. In the design process we kept in mind the following requirements:

- Processing of long-term uninterrupted audio stream,
- Continuous operation of the system,
- Easy extensibility of the system,
- Low memory profile,
- Low computational requirements,
- Fast response,
- Independence from environmental changes (such as SNR - Signal to Noise Ratio changing).

The most important requirement that had to be taken into account was the uninterrupted processing of the input audio stream and of course the corresponding continuous operation of the whole system. This system is designed to be used for long-term monitoring of large urban areas for suspicious acoustic events. This also implies the implementation of solutions for saving memory and computational resources. These requirements have to be combined with fast response of the system. The system should be able to provide results of its detection immediately when the acoustic event happens. The system has little time for computation and confirmation of results as well as real-time processing of the input audio without any significant delay.

The system composition is similar to other systems used in a speech recognition task, but it was built from scratch (no third party code used) and without typical grammar rules. The whole system can be divided into two parts (Fig. 1). The first part is responsible for audio preprocessing and feature extraction. The second part is the decoder, which uses a search network (composed from all available knowledge sources) to decode, find, detect and classify desired audio events. The knowledge sources are simpler in this case when compared

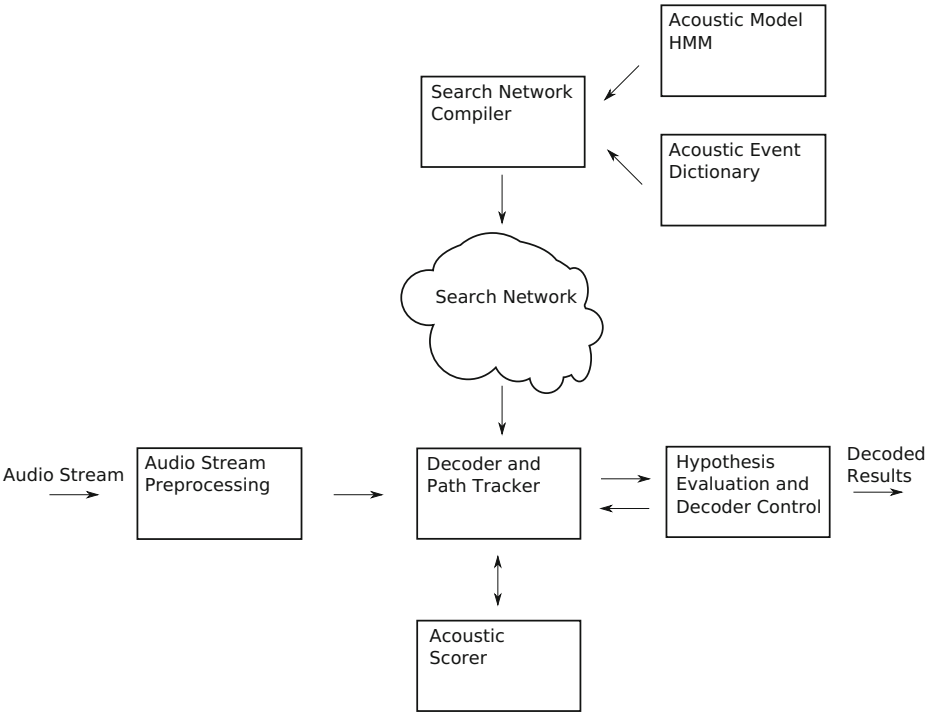


Fig. 1 Internal structure of the EAR-TUKE system

to speech recognition as none or little information about the sequence of acoustic events is involved. The main part of the knowledge sources is based on the acoustic information about the events. This information is statistically described using an acoustic model. The audio preprocessing and decoding, as mentioned before, should be adapted for continuous and uninterrupted processing of input audio signal. As we describe later, we were able to achieve such goal.

The EAR-TUKE system is constructed from individual blocks meaning that each block can be separately used without a present definition of another block of the system. The implementation of each block is self-sufficient and tightly designed, which means that some of the functions of the same block cannot be altered or changed without significant impact on the block itself. Some of the functions are implemented and hard-coded directly into the main functions of the block. This is done in favor of fast response and processing of the audio signal. All these requirements impose limits to the usage of our system, but we have still been able to maintain our desired level of generality.

3 Audio preprocessing and feature extraction

The preprocessing part represents the front-end of our system. It handles audio preprocessing and feature extraction for the next module in the acoustic event detection. The front-end is designed to be modular, supporting easy implementation and addition of new modules.

The basic idea behind the front-end is to process audio stream blindly, meaning that no analysis of the input stream is done by the front-end (such as VAD - Voice Activity Detection, Silence Suppression, etc.), but the feature extraction. This restriction also applies to any kind of activity detection that cuts out only segments where the acoustic event might be present. All input audio is processed by the front-end module and outputted in its entire length.

Currently, the standard Mel-Frequency Cepstral Coefficients (MFCC) can be extracted by our front-end block as well as the intermediate results, the Frequency Bank Coefficient (FBANK) and Mel-Spectral Coefficient (MELSPEC) [33]. Their computation process is described below.

The ear's perception of frequency components in the audio signal does not follow the linear scale, but rather the Mel-frequency. The relation between the frequency and the Mel-frequency is given by the formula [33]:

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

where f is frequency in Hertz. MFCC, FBANK and MELSPEC coefficients are computed according to the Fig. 2.

Normally, signal is filtered using preemphasis filter, and then the 25ms Hamming window with 10ms frame shift is applied on the frames. In our processing preemphasis is omitted, because it is useful mainly for speech signals. The resulting coefficients are transformed to the frequency domain via the discrete Fast Fourier Transform (FFT), and the magnitude spectrum is passed through a bank of N triangular filters. The result of this process are MELSPEC features. The energy output from each filter is then log-compressed

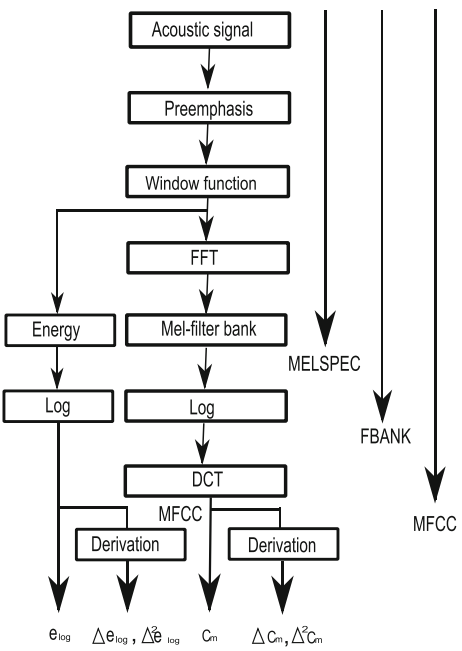


Fig. 2 Principial block scheme of MELPSEC, FBANK and MFCC coefficients

m_j and represents FBANK features. Finally MFCC coefficients (c_i) are obtained after the transformation to the cepstral domain by the Discrete Cosine Transform (DCT) [33]

$$c_i = \sqrt{\frac{2}{N}} \sum_{j=1}^{N} m_j cos\left(\frac{\pi i}{N}(j - 0,5)\right). \tag{2}$$

The performance of recognition systems can be enhanced by adding time derivatives to the basic static features c. The commonly used ones are the first and second regression coefficients referred to as delta (Δ) and acceleration (or delta delta or double delta: $\Delta\Delta$) coefficients. The delta coefficients (d_t) are computed by the formula [33]:

$$d(t) = \frac{\sum_{\theta=1}^{\Theta} \theta(c_{t+\theta} - c_{t-\theta})}{2\sum_{\theta=1}^{\Theta} \theta^2}, \tag{3}$$

where d_t is delta coefficient computed at time t by corresponding static coefficients $c_{t-\theta}$ and $c_{t+\theta}$. The value Θ was set to 2. The same routine is applied on the delta coefficients for computation of acceleration coefficients [33].

Additionally, energetic and/or zero Cepstral coefficient can be included into resulting output feature vector. For each output feature vector, the delta and acceleration coefficients can be computed. At the end of the front-end chain preprocessing modules, the Cepstral Mean Normalization (CMN) module can be attached. The CMN module is capable of online CMN using sliding window of preset length [1]. Although the CMN is supported, we have found that it is unnecessary in the acoustic event detection task as will be described later in this paper. This configuration of the front-end is also based on our previous research in this area, the MFCC and MPEG-7 features investigation [31] and comparison of available feature types in [32].

4 Acoustic modelling

Hidden Markov Models (HMM) are a very popular classification technique for acoustic modeling of time-invariant events such as human speech, animal sounds, environmental sounds, etc. Their good recognition performance has caused the wide-spread use of HMM for various recognition tasks. The goal of acoustic processing is to provide appropriate method to determine the conditional probability $P(O/W)$ that an event/word W will represent an acoustic vector/observation O. Various continuous ergodic HMM-s with PDFs (Probability Density Functions) mixtures from 1 to 1024 were used. The diagonal covariance matrix was used in all experiments. In the training process, the acoustic models with one, two and three emitting states were used for each event class and background sounds.

The model parameter initialization was performed in 15 iteration cycles in average (using maximum likelihood criterion) and the two-fold Baum-Welch training procedure parameter re-estimation (using expectation maximization criterion) was applied in 50 iteration cycles on average. The maximum number of re-estimation cycles was set to 100.

5 Decoder and path tracking algorithm

The decoder is the second standalone block of the EAR-TUKE system. The main function of the decoder is the actual detection and classification of the acoustic events using extracted

feature vectors from the input audio stream. In order to fulfill its function, the search network needs to be provided along with suitable acoustic model definitions. The decoder then cooperates with the acoustic scorer and the block for controlling the decoding mechanism to achieve an efficient detection and classification functions at the same time.

The decoder is designed to be as general as possible and utilizes WFSTs for the search space representation and the interchangeable acoustic scorer representation. This way the decoder can process an arbitrary set of input feature vectors, not only the ones mentioned above. Although we tried to design the decoder as reusable as possible, it contains one inseparable part: the result tracking algorithm. It handles the retrieval of the decoding result from a path that the decoding algorithm took in the search network. This was done in favor of speed increase and for simpler construction of the decoder. In the next subsections the parts participating in decoding process will be described.

5.1 Search network

The search network is represented using WFSTs which is a state machine consisting of states and transitions between them [19]. Each transition has input and output symbols with weights. Each search network (see Fig. 3) contains one initial state (marked bold) and one or more final states (marked with double line). The WFSTs are used because of their ability to translate input sequence of symbols to output sequence of other symbols, in our case from input acoustic observations to names of the detected acoustic events [14].

This representation of the search network has its advantages. The first advantage is that the decoder does not need to know any information provided in the search network as long as the format of the input symbols in the network is coordinated with the acoustic models. This also means that the set of desired events to be detected can be easily adjusted and we can include additional statistical information that needs to be taken into account by the decoding process.

The decoder is based on Hidden Markov Models (HMMs). An HMM consists of two parts. The topology of the model is represented by the transition model and probability density functions (PDFs) [24, 33] represent observations probabilities. The WFSTs are used for the representation of the topology while the transitions contain probabilities of acoustic

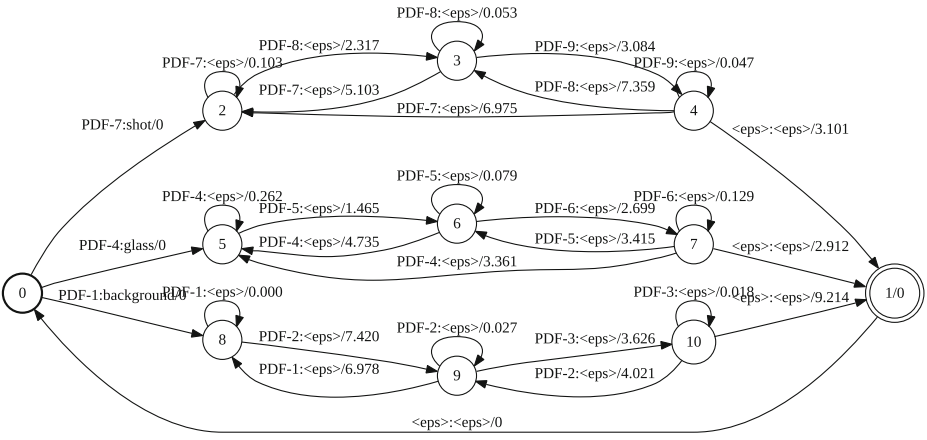


Fig. 3 Weighted Finite-State Transducer Search Network used for breaking glass and gun shot detection (the transition probabilities are in logarithmic scale without the obvious minus sign)

model transitions and input symbols are informing about PDFs that need to be used for scoring the input vectors. The output symbols on the transitions contain names of the acoustic events. The advantage of representing the topology of HMMs by WFSTs is that the search network can support different topology and number of states for different acoustic events without prior knowledge of the decoder.

The search network for our system has to be generated before the process of detection, meaning that the network representation should be static. As the search network for this kind of task tends to be relatively small (a few dozens of states), the static representation does not impose any limitation on the usage of the system even on embedded devices with limited resources. The small size of the network also supports fast rearrangement of the network on demand. Another advantage of the static search network is that decoder does not need to deal with the composition of it on-the-fly and so the decoder can work faster [4].

In order to do the detection and classification of an acoustic event, the search network contains models not only for acoustic events, but also for background sounds. An example of such search network is in Fig. 3. The network contains three events: gunshot, glass breaking and one background model, displayed as the output symbols on the transitions. Symbol "$< eps >$" describes empty output symbol. The input symbols are used as information for scoring process and basically represent the states (Gaussian Mixtures) in HMM acoustic models. The weights on transitions represent transition probabilities between states of the HMM acoustic model. This search network is stored in a binary file that includes the acoustic model information about PDFs allowing faster loading of all required information into the memory.

5.2 Acoustic scorer

As it was mentioned above, the decoder requires a scorer for evaluation of probability of the input feature vector according to the used acoustic model. Our system contains a scorer using PDFs of HMMs and it is based on principle of computing Mahalanobis distance of the feature vector and Gaussian function (4). The scorer uses information about presence of input symbols provided by the search network in order to know which PDFs of the input feature vector should be used for scoring.

$$\begin{aligned} b_n(o_t) &= \sum_{m=1}^{M} c_{nm} \aleph(o_t, \mu_{nm}, C_{nm}) \\ &= \sum_{m=1}^{M} c_{nm} \frac{1}{\sqrt{(2\pi)^D |C_{nm}|}} exp\left(-\frac{1}{2}(o_t - \mu_{nm})^T C_{nm}^{-1}(o_t - \mu_{nm})\right), \end{aligned} \tag{4}$$

The $b_j(o_t)$ is the resulting Mahalanobis distance of the input feature vector o_t in time t with n-th mixture of PDFs $\aleph(o_t, \mu_{jm}, C_{jm})$. As it was mentioned above, the indices n are provided by the search network. Each mixture of PDFs consists of M number of individual PDFs, where each of them is defined by its mean μ_{jm}, variance C_{jm} and its weight c_{jm}. D is dimension of the input feature vector. The scorer is a separate block of the system, so it can be easily replaced with another block that supports another kind of acoustic model or input set of feature vectors.

In order to increase robustness of our system in environment conditions changes, such as changes in signal to noise ratio (SNR), we have included an additional feature. The principle is to remove a set of basic coefficients of the feature vector during scoring process leaving only delta and acceleration coefficients. We have found that this important feature works

in acoustic event detection task better than using CMN as described in [15] (see detailed description in Sections 6.3 and 6.4).

5.3 Decoding algorithm

The task of the decoder is to find the best sequence of the acoustic events based on the input feature vectors extracted from input audio signals. The decoder inspects the search network using feature vectors as guidance in synchronous manner where one step in the search network is performed for one feature vector.

During the search the intermediate scores composed from transition probabilities and acoustic score from PDFs are computed. The result is defined as the most probable path in the search network. The Viterbi decoding criterion is used and implemented in the form of token passing algorithm in a similar way it is done in Hidden Markov Model Toolkit (HTK) [33]. The token is an object that can travel through the search network and remembers all required intermediate results for constructing the final sequence. In the case of our system, the token remembers these four elements during its travel through the search network:

- Accumulated transition score (a_{g_t}),
- Accumulated acoustic score (b_{n_t}),
- Last output symbol found,
- Pointer to a token with last non-empty output symbol.

$$\begin{aligned} P(O|W,\Theta_a) &\approx \max_{Q} P(O,Q|W,\Theta_a) \\ &\approx \max_{Q} \prod_{t=1}^{T} a_{q_{t-1},q_t} \prod_{t=1}^{T} b_{n_t}(o_t) \end{aligned} \tag{5}$$

When the token travels through the search network, the weight on the transitions (a_{q_{t-1},q_t} - transition weight from state q_{t-1} to state q_t) is added to the accumulated transition score. In a similar way, accumulated acoustic score is calculated as Mahalanobis distance of the input feature vector to the PDFs (b_{n_t}) with incoming feature vector according to the input symbol n on the same transition.

These accumulated scores are used together for comparing tokens to each other according to Viterbi decoding criterion (5). The result (sequence of acoustic events W) is then defined as sequence of states Q (path through the search network) with maximum probability $\max_Q P(O,Q|W,\Theta_a)$ when a set of O feature vectors is observed and an acoustic model Θ_a is used.

The next information that needs to be found together with the accumulated score are the output symbols on the transitions. The best output symbol sequence with the highest score gives us the result of the decoding process. The last two items of the token are used exactly for that purpose. Each token remembers the last output symbol that it passed during its travel through the search network along with the pointer to a token with previously passed output symbol. The pointer allows the tokens to chain themselves in order to find the hypotheses of the decoding process and finally the result of the whole decoding.

The complete decoding algorithm can be seen in Fig. 4. The black bullets represent the tokens in the search network, while the black arrows represent the pointer to token with

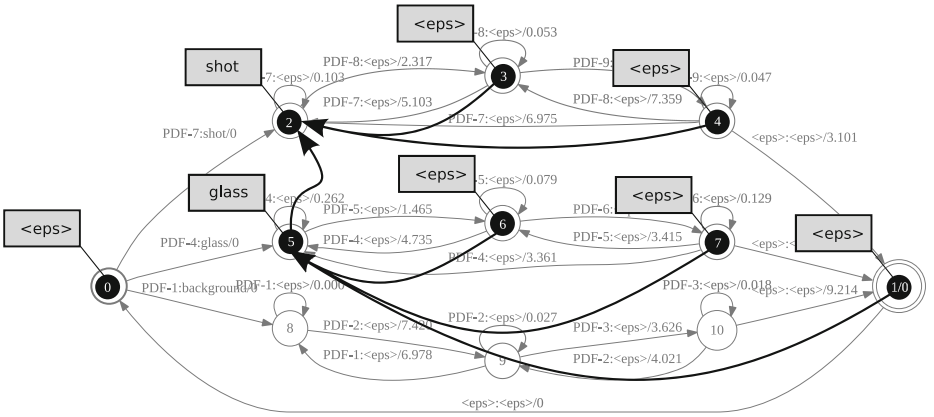


Fig. 4 Visualization of the decoding process on the previous Weighted Finite-State Transducer Search Network example (black bullets represent tokens, black arrows represent pointers to previous tokens with non-empty output symbol, gray color represents the search network with input/output symbols and transition log probabilities without minus sign)

last non-empty output symbol. The probabilities are depicted in logarithmic scale without obvious minus sign. In the beginning only single token exists in the search network and it is placed into the initial state.

Although logical assumption is to place the tokens where the information for decoding is, as it is in case of other systems like HTK, in our case the tokens are placed in states regardless of the information on transitions.

The tokens are traveling from state to state. With an incoming input feature vector the tokens are copying themselves to all consecutive states accumulating the total score on the transitions and remembering the output symbols. Each new copy of a token, before passing through a transition, inherits from previous one its score and pointer to a token with last non-empty output symbol if there has been any (in essence remembering the way back). If the new token passes a non-empty output symbol on its transition to the next state, it remembers it and the subsequent tokens copied from this one will inherit a pointer to this token.

By using the pointer to tokens with the last found non-empty output symbol, the hypotheses are maintained during the decoding process. When the last token in the chain is removed from the decoding process according Viterbi decoding criterion, the whole hypothesis (the whole chain) should be also removed. Any token in the hypothesis chain can be part of another chain, so each token contains a reference counter.

After a token is removed from the search network (does not reside in any state), it is still kept in the memory in case there were any other tokens referencing to it. If the reference counter drops to zero, the token is completely removed from the decoding process. To retrieve partially or fully decoded result, the token from the final state can be taken.

The chain where the final state belongs represents the result of decoding process. This solution does not affect the speed of the decoder and it provides very good memory and stability increase for short acoustic events detection without significant accuracy change as it is described in the next subsection.

5.4 Hypothesis evaluation and decoding control

This last part of the EAR-TUKE system is a small but important block. Thanks to this block, the continuous detection and classification is achieved. This block actually does more than that, it also saves computational resources.

As the preprocessing block does not provide any information to distinguish acoustic events from background sounds, the detection and the following classification needs to be done in a different way to [17] where two consecutive steps of event detection and event classification are used. In the usual approach the decoding results are collected at the end of the recording. Unfortunately, this cannot be done for a continuous audio stream input because the audio stream does not end, and there is no block which cuts some part of the stream (as it is done when an event detection block exists [17]).

If we use our modified decoder and the presented search network for processing and recording, the output will be a sequence of acoustic events detected. We have stated at the beginning of this paper that we want to design the system to continuously monitor the audio stream. To achieve this, a block for controlling the decoder based on the hypothesis evaluation was designed. The function of this block is to look at the end of the current decoding hypothesis (the one represented by a token in the final state) after each input feature vector. When the hypothesis meets preset conditions, the whole decoding process can be quickly reset, meaning that all hypotheses will be removed and all resources released. The new decoding process will start immediately afterwards.

The conditions were chosen so that they take advantage of the background models in the search network. We found that it is safe to reset the decoder when the most probable hypothesis reaches the background model (Fig. 5). To be sure that this hypothesis is not a temporary glitch, there is also a condition concerning the duration of the hypothesis. Empirically, the optimal time was finally set to 100 milliseconds. A safe reset of the decoder means that it does not produce an error or a false hypothesis afterwards. The results of experiments with continuous resetting of the decoder and without it were the same [21].

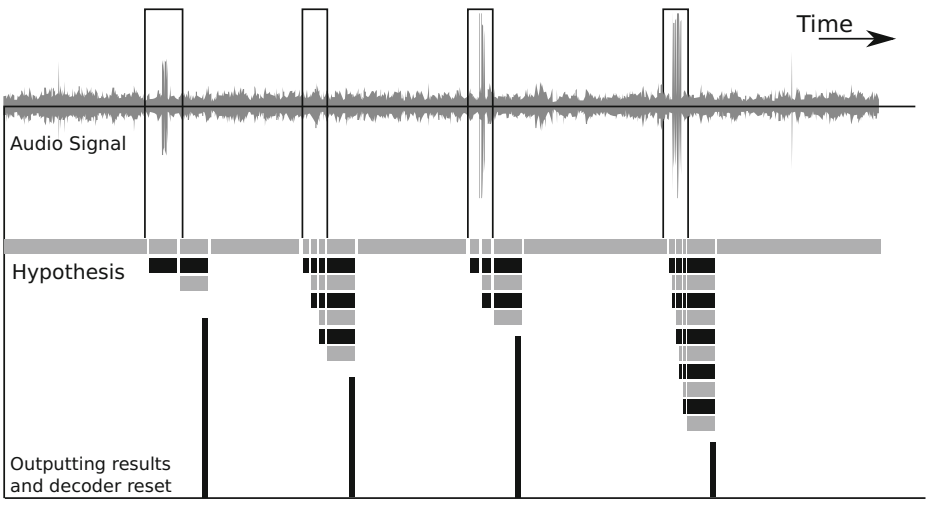


Fig. 5 Example of the hypothesis evaluation (black color represents the acoustic events hypothesis and gray color represents the background hypothesis)

6 Development and evaluation

In this section we will describe important steps in the research of the acoustic event detection and the development of the main system.

We started the development of our system for detection of the acoustic events such as gunshot and breaking glass using freely available engines such as Julius [13], Kaldi [23] and Torch [3]. Soon enough we found out that there were limitations not compatible with our ideas about the system. We were unable to construct a system for continuous processing of the audio stream because of input length limitation imposed in Julius [20]. This limitation has its purpose in saving computational resources (memory) when a too long input stream is going to be processed. The Kaldi and Torch are too complex systems for such a simple task where only fixed grammar (background and foreground event sets) and small number of models are used. We decided to build our own small system with possibility of deployment on embedded devices (using non-dependency source code) instead of using multi-purpose solutions.

One way to overcome such limitation (long input stream) is to use segmentation mechanism. In the case of speech recognition, the voice activity detector is used. This detector can be based on simple energy threshold or more complex solutions such as GMM (Gaussian Mixture Models). The main idea is to cut out segments from the input stream that contain only important data (speech). In our case such segmentation mechanism is not easy to construct because of different characteristics of the glass breaking and gunshot events. The acoustic events are totally different in nature. The glass breaking lasts longer than a gunshot and has lower energy, while gunshots have their energy mostly focused in one burst at the beginning (see Fig. 6).

A different situation arises when a simple energy detector is used and the background sounds in the monitored area are too loud. In this case the segmentation mechanism is unable to identify any segments from the input stream and will flood the subsequent classification module resulting in system crash or no detection at all. All these problems mean that the segmentation mechanism is important and essential part of the system for continuous monitoring. Simple segmentation such as energy based one is not suitable for this purpose. More complex solutions tend to be complex enough to not only do the segmentation but they are already capable of classification of the acoustic events. At this point we decided to take advantage of this fact and construct a system with simultaneous detection and classification. The original idea was supported by the fact that more complex models for classification of the acoustic events are logically capable of their detection. This way we were able to construct the system for acoustic event detection that is capable of continuous processing of the input stream.

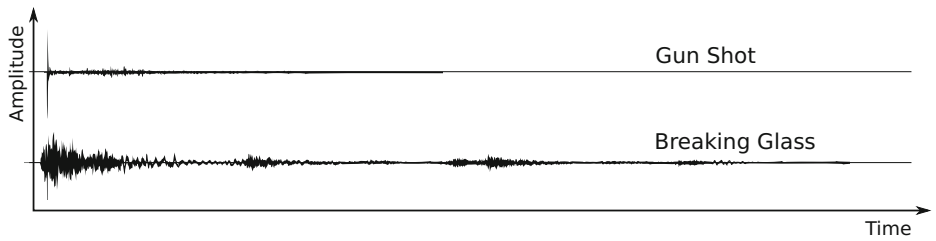


Fig. 6 Gun shot and glass breaking depicted as waveforms in time-domain

In the next subsections we will look in more detail at the important part of the system allowing us to process continuous audio stream to constantly monitor the audio stream for glass breaking and gunshots. Then we will look at the system refinement to achieve more robust (to different SNR conditions, microphone sensitivity/distance - different levels of noise and events) and more accurate detection.

6.1 Experimental setup

In this part of the paper the acoustic event database and the methodology of evaluation is described. The investigation of a noise impact on the detection results follow after the description.

At first, the recognition in presence of noise is reported, then CMN as the standard process for noise reduction was applied for full and reduced feature sets. The corresponding tables contain the average accuracy across all tested SNR scenarios (using mixed SNR test set described in detail in Section 6.3). Our approach to solving SNR problem was also partially inspired by the feature selection strategy. The type of feature (MFCC, FBANK, MELSPEC) and its temporal evolution plays an important role in this issue. The results of best models are presented with respect to changing SNR. A live presentation of our system is described at the end of this section.

6.1.1 Acoustic event database

During the development of our system we created a database for training and testing the system [22]. The JDAE-TUKE (Joint Database of Acoustic Events and background sounds from Technical University of Kosice) contains 150 realizations of glass breaking, 463 realizations of gun shots with present background (mostly traffic, birds, etc. but also events that could be confused with detected events such as trash can slam, car door slam, siren, horn, applause, fireworks, strong wind, strong music, etc.). The database also contains separate pure background recording (traffic) in length of 53 minutes. The database is divided into non-overlapping testing and training part. The initial testing part involves 46 gunshot and 13 glass breaking realizations with already present background sound in the same length. The rest of the database belongs to the training part.

6.1.2 Evaluation metric

Accuracy was used as the main evaluation metric in our case. The accuracy is defined as ratio according to (6) [33].

$$ACC(Accuracy) = \frac{N - D - S - I}{N} \times 100. \tag{6}$$

The N is total number of reference events, while S, D and I are errors that can occur in the detection and classification process. The foreground acoustic event can be either detected correctly or incorrectly – meaning a substitution error (S) can occur. The event can be missed resulting in deletion error (missed detection - D) or more foreground events can be detected in addition to the reference events resulting in insertion error (false alarms - I). In this paper all accuracy results are displayed in percentage (%).

We have also defined conditions when the acoustic events were detected and when they are treated as false alarms or missed detection. The foreground event (breaking glass or gun shot) is correctly detected if there exists at least one acoustic event in the system output

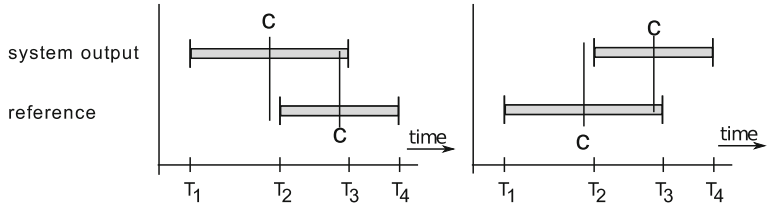


Fig. 7 Evaluation of the system output and reference (c - temporal center of the acoustic event). The event is correctly detected if there exists at least one acoustic event in the system output result, of which temporal center is in between timestamps of the reference event (left), or another way around, the temporal center of the reference event is in between the timestamps of the event in the system output result (right), and the labels are the same

result whose temporal center is between timestamps of the reference event, or another way around, the temporal center of the reference event is between the timestamps of the event in the system output result, and the labels are the same (see Fig. 7). If the labels are different then the error is substitution. We also allow background events to be detected under reference foreground without error. Any foreground event under background reference is a false alarm and any foreground reference without corresponding detected foreground event from the system output is a missed detection.

6.2 Hypothesis evaluation and decoding control

In the beginning of our development we compared our version of the acoustic event detection with the HTK toolkit to see if we are on the best path to be comparable with other systems [21].

The 2-state HMM model from the training part of the database JDAE-TUKE with 4 PDFs on a state using MFCC parametrization including zero, delta and acceleration coefficients was trained on and transformed into WFST search network. The training was done using HTK toolkit.

The detection results were the same compared to HTK based system. Both systems detected 58 from 59 reference acoustic events, so there was 1 missed detection. Both systems also included 4 false alarms. This was done by comparing results at the end of the recording, meaning no system for segmentation of the input stream was used.

In a case of continuous input stream, no results would be available after acoustic events' occurrence, but only at the end of the stream (see Table 1 for plain decoding results and for corresponding confusion matrix see Table 2).

For verification of the chosen setup, we have realized the 10-fold cross-validation where the whole database was divided in 10 equivalent sets and each tenth was chosen for testing and the rest for training in a loop (Leave-one-out). We obtained an average cross-validated accuracy of 94.99 % with a minimum of 90.90 % and maximum of 98.85 %. It could be concluded that the chosen test set was one of the most challenging and

Table 1 Accuracy (%) of EAR-TUKE and HTK comparison

System	Plain decoding	Added hypothesis evaluation and decoder control
EAR-TUKE	91.52	91.52
HTK	91.52	–

Table 2 Confusion Matrix of the test results (BG - Background, omitted in evaluation)

		Hypothesis			
		BG	Glass	Shot	Missed
Ref.	BG	78	0	0	1
	Glass	0	*11*	*2*	0
	Shot	0	0	*45*	*1*
	False Alarm	0	0	*4*	

there is more space for improvement of the results using the algorithms described in the paper.

We have achieved the same result by applying the hypothesis evaluation and decoder control block to continually reset the decoder and output results (see Table 1). This means that we were able to create a system for continuous monitoring without segmentation mechanism before classification. The duration condition for the actual background hypothesis was set to 100 milliseconds.

We have also conducted tests with shorter condition time settings of about the duration of the last background hypothesis, but with no better results. This means that any time setting above 100 milliseconds is stable.

6.3 Robustness in changing signal to noise ratio (SNR)

We have tested our system for robustness against noise (SNR is computed as acoustic event to background noise rate) change in an environment where the long-term monitoring should be realized. For testing purposes we recorded additional 30 minute long background recordings to the testing part of the database and mixed them with arbitrarily placed acoustic events with different SNRs. We have created mixed SNR test set recordings with -3dB, 0dB, 3dB, 6dB, 8dB, 11dB, 14dB, 17dB, and 20dB SNR. We tested all combinations of the models trained. The models had 1 to 3 states HMM with 1 to 1024 PDFs on a state. The background recordings contained a lot of sounds which could confuse the system and be detected as gunshot or breaking glass (trash can slam, car door slam, siren, horn, etc.).

We were able to increase the accuracy of our system by changing the acoustic scorer implementation and removing the basic coefficient of the feature vector. The idea was to remove the correlation of the background noise or microphone sensitivity (which is mainly in basic coefficients vector) instead of using common CMN approach. The delta and acceleration (delta delta) coefficients should describe the speed and level difference of the foreground events. In this case the feature vector from the original 39 coefficients was reduced to 26. The results are displayed in Table 3 as average accuracy along all SNR test recordings, and we can see that there is accuracy increase in all reduced HMM models. The advantage of this approach is also the simplicity of the implementation, where no special feature transformation is needed and also the HMM model for the full 39 feature vector could be used, but only the first 13 PDF's could be ignored in the decoder using only a simple switch. Using this approach we have been able to achieve 93.108 % accuracy in mixed SNR conditions test set.

From the results, the decrease in performance could be noticed for higher number of PDFs, which is most probably caused by overtraining phenomena when a small training data set is not sufficient for such detailed space description estimation.

Table 3 Accuracy (%) of EAR-TUKE system comparing common 39 feature vector to the first 13 features removal (reduced) in mixed SNR conditions test set using 20dB to -3dB acoustic events in the presence of noisy background

PDFs	1 – State	1 – State Reduced	2 – State	2 – State Reduced	3 – State	3 – State Reduced
1	*85.795*	87.482	*83.122*	88.326	*76.371*	*86.498*
2	83.826	86.357	82.982	89.170	72.152	85.232
4	80.731	86.639	75.246	*92.124*	64.838	79.044
8	68.354	89.311	66.807	88.186	61.181	78.481
16	68.917	*93.108*	62.869	76.934	62.729	69.761
32	63.150	86.920	62.447	64.557	64.276	63.572
64	61.885	78.200	63.994	64.276	64.557	65.260
128	62.588	76.512	64.838	65.823	64.135	63.994
256	61.885	75.387	64.698	65.401	63.572	64.698
512	64.557	69.620	63.713	65.823	63.291	65.682
1024	64.979	65.541	63.713	71.589	63.291	71.027

6.4 Influence of CMN in mixed SNR test set

Next, the influence of CMN on the system in the same mixed SNR test set as in the previous paragraph was investigated. From the training part of the database the same models were trained on, but in this case the CMN was enabled. Thus the models were trained on original MFCC vectors with CMN applied on them. Then the system was tested on the SNR mixed recordings from which the reduced feature vectors were extracted with and without CMN enabled.

In the training process simple CMN was used. The mean value was estimated from the entire recording and then subtracted from each feature vector. In the testing phase, where we needed to simulate live application, the CMN were computed using floating window.

Table 4 Accuracy (%) of EAR-TUKE decoder in mixed SNR test set and the influence of using CMN on reduced input feature vectors (suppressing first 13 coefficients)

PDFs	1 – State CMN	1 – State	2 – State CMN	2 – State	3 – State CMN	3 – State
1	87.904	87.904	*89.592*	*89.592*	65.963	66.245
2	87.764	87.764	89.311	89.311	*72.574*	*72.855*
4	87.623	87.623	82.700	82.982	58.368	58.368
8	*90.717*	*90.999*	71.589	71.871	58.650	58.650
16	90.014	89.873	65.682	65.823	59.634	59.634
32	79.606	79.466	62.729	63.010	59.775	59.775
64	76.512	76.512	63.854	63.572	61.463	61.463
128	70.745	70.745	64.838	64.838	62.729	62.869
256	68.776	68.776	67.089	67.370	64.838	64.979
512	69.058	68.917	68.354	68.917	66.526	66.948
1024	62.588	62.447	69.620	70.042	67.932	67.792

For feature vector $X(n)$, the mean value was estimated from a defined number of $n + \frac{M}{2}$ following and $n - \frac{M}{2}$ preceding feature vectors depending on the chosen window with length of M according to (7). At the beginning and the end of the recording, the mean value was estimated based only according to future or past feature vectors respectively. In either case the CMN vector was computed from the full feature vector (39 coefficients: basic, delta and acceleration coefficients) and then the normalization procedure was applied in the last stage of the current input vector computation. If there was a reduction of the feature vector applied in the decoder (described in the previous Section 6.3), only the scoring value of the first 13 (basic) coefficients was ignored, so no modification of the input, pre-processing or HMM model was required.

$$X_{CMN}(n) = X(n) - \frac{1}{M} \sum_{m=n-\frac{M}{2}}^{n+\frac{M}{2}} X(m) \tag{7}$$

The results are in Table 4 where the accuracy increase is not significant. In cases of 3 and 1 state models there is slight increase of the accuracy while for 2 state models it stays the same. Over all it means that when using acoustic models with CMN enabled, the reduced feature vectors do not need the CMN applied during testing phase. This gives us an advantage because the application of a CMN on live audio stream is problematic. Usually the CMN is applied on the whole recording, but in case of live audio stream, a floating window that may not be accurate is used [1]. In our case the floating window implementation of CMN was used for testing while for training the CMN was applied on the whole recording.

Comparing the results with previous section we can conclude that no CMN is required at all. Using reduced feature vector the highest accuracy of the detection can be achieved completely without CMN applied for both model in training and on reduced feature vectors in testing phase (Fig. 8).

6.5 Acoustic feature selection

Various types of features can be extracted from acoustic signal. Some of them are effective for describing the nature of analyzed sound and on the other side there are also features that have low information content. We investigated different selection criteria, based on the mutual information criterion. From the group of 13 selection algorithms [2] (MRMR MIBASE, MRMR MID, MRMR MIQ, BETAGAMMA, CONDRED, JMI, MIFS,

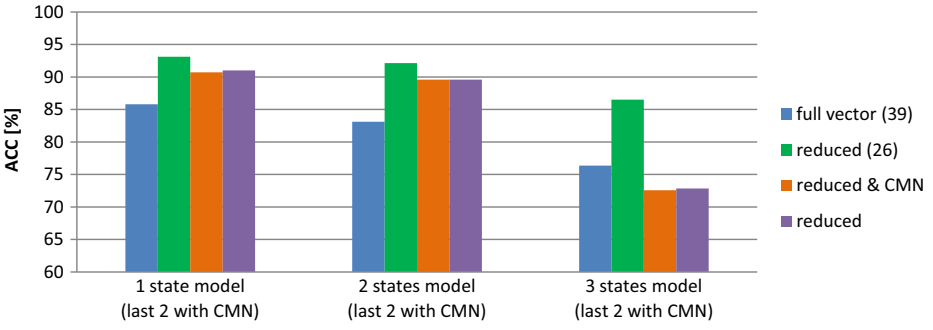


Fig. 8 Accuracy comparison for different HMM model used (different number of states and one trained without CMN and second trained with CMN enabled) compared to feature vector used during testing phase

CMI, CONDRED, CMIM, ICAP, DISR, and CIFE), the best features were chosen by the BetaGamma algorithm. All mentioned algorithms are compared in the work [10].

BetaGamma selection criterion is inspired by the Conditional Mutual Information (CMI) criterion [2]. BetaGamma is defined as follows:

$$J_{BetaGamma} = I(x_k; y) - \beta \sum_{j \in S} I(x_j; x_k) + \gamma \sum_{j \in S} I(x_j; x_k | y); \tag{8}$$

where S denotes feature set with individual features x and y denotes a target class. The expression $I(x_k; y)$ represents the relevance of a feature to the target class, $I(x_j; x_k)$ is a feature dependency and $I(x_j; x_k | y)$ is a conditional dependency of features to the target class y. β and γ are the penalization factors, where β is the inter-feature dependency penalization factor and γ is the class-conditional penalization factor, both are from the range $< 0, 1 >$.

We analyzed the relevance of common Mel-Frequency Cepstral Coefficients (MFCC), Mel-Spectral Coefficients (MELSPEC) and Frequency Bank Coefficients (FBANK) and also MPEG-7: Audio Spectrum Envelope (ASE), Audio Spectrum Flatness (ASF), Audio Spectrum Spread (ASS), Audio Spectrum Centroid (ASC), Audio Waveform (AW), Spectral Flux, Spectral Roll-off, Skewness, Kurtosis, Zero Crossing Rate (ZCR) and energy (E) features [10, 11]. For all features, delta and double delta coefficients (acceleration) were computed. Supervectors (dim 303) included all mentioned features (101 static + 101 Δ + 101 $\Delta\Delta$).

During the selection process, features were ranked and finally new feature order according to the feature relevance was obtained. In the first positions, features with high relevance were placed and the end of this sequence belonged to the less relevant ones. The best 60 features selected according to the BetaGamma are depicted in the Fig. 9.

The core (80 %) of this effective feature vector included mainly MFCC, FBANK and MELSPEC coefficients. They were created primarily for speech related tasks but they are very effective also in non-speech tasks such as music retrieval, acoustic event recognition, etc. [6, 18].

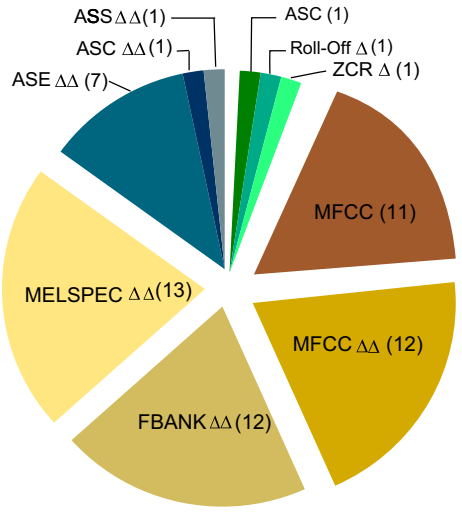


Fig. 9 The best 60 features according to BetaGamma selection, where the feature types, their temporal status and number of coefficients are introduced (feature groups are distinguished by colors)

Table 5 Accuracy(%) of EAR-TUKE system using MFCC in mixed SNR test set

PDFs	1 – State	1 – State Reduced	2 – State	2 – State Reduced	3 – State	3 – State Reduced
1	71.11	75.56	64.44	60.00	*67.78*	60.00
2	72.22	76.67	*68.89*	60.00	*67.78*	60.00
4	74.44	73.33	*68.89*	60.00	66.67	60.00
8	*78.89*	72.22	*68.89*	60.00	*67.78*	60.00
16	68.89	66.67	*68.89*	60.00	*67.78*	60.00
32	61.11	60.00	67.78	60.00	*67.78*	60.00
64	57.78	60.00	*68.89*	60.00	66.67	60.00
128	67.78	60.00	66.67	60.00	63.33	60.00
256	71.11	*80.00*	64.44	60.00	60.00	60.00
512	58.89	72.22	63.33	60.00	60.00	*63.33*
1024	37.78	74.44	60.00	*62.22*	60.00	*63.33*

Very interesting is the number of static and dynamic features, where the majority of selected features were the second temporal derivation of static coefficients called acceleration or $\Delta\Delta$ coefficients. This also confirms the relevance of the feature reduction to improve accuracy.

We had known this fact before so our front-end processing was based on the MFCC and features were extracted during their computation i.e. FBANK and MELSPEC. A special attention was devoted also to the investigation of delta and acceleration coefficients.

6.6 Testing available feature types

In the previous section we have found out that the majority of selected features are delta and acceleration coefficients, which was also confirmed by our empirical test with the feature reduction (removal of basic coefficients). In this section, three features types (MELSPEC,

Table 6 Accuracy(%) of EAR-TUKE system using FBANK in mixed SNR test set

PDFs	1 – State	1 – State Reduced	2 – State	2 – State Reduced	3 – State	3 – State Reduced
1	50.00	62.22	50.00	*90.00*	58.89	62.22
2	40.00	56.67	50.00	86.67	57.78	60.00
4	50.00	90.00	60.00	77.78	44.44	60.00
8	55.56	*94.44*	47.78	66.67	60.00	60.00
16	*65.56*	85.56	61.11	61.10	*61.11*	60.00
32	60.00	76.67	56.67	60.00	60.00	60.00
64	57.78	80.00	61.11	62.22	57.78	60.00
128	55.56	55.56	56.67	62.22	60.00	60.00
256	60.00	81.11	*62.22*	63.33	60.00	63.33
512	56.67	76.67	57.78	65.56	58.89	*66.67*
1024	61.11	78.89	54.44	66.67	54.44	65.56

Table 7 Accuracy(%) of EAR-TUKE system using MELSPEC in mixed SNR test set

PDFs	1 – State	1 – State Reduced	2 – State	2 – State Reduced	3 – State	3 – State Reduced
1	56.67	68.89	61.11	88.89	57.78	*84.44*
2	58.89	74.44	60.00	85.56	*62.22*	*84.44*
4	63.33	84.44	*66.67*	*91.11*	47.78	82.22
8	60.00	86.67	57.78	86.67	46.67	50.00
16	58.89	91.11	36.67	64.44	37.78	42.22
32	57.78	93.33	44.44	63.33	43.33	46.67
64	60.00	94.44	44.44	54.44	43.33	48.89
128	64.44	96.67	44.44	53.33	50.00	52.22
256	*65.56*	*98.89*	50.00	54.44	57.78	55.56
512	64.44	93.33	53.33	42.22	53.33	57.78
1024	60.00	90.00	57.78	68.89	53.33	53.33

FBANK and MFCC) will be tested and compared. The models with delta and acceleration coefficients including CMN were trained on our database. Please note that no zero coefficient was used here as opposed to the previous sections, meaning that the feature vector is only 38 dimensional. The models have 1 to 3 states and 1 to 1024 PDFs on each state. The testing set stayed the same, as well as the SNR recordings that were used in previous tests. Table 5 contains the results for MFCC feature set, the Table 6 for FBANK feature set and finally the Table 7 the MELSPEC feature set.

From the results we can see that reduction of the feature vector and using only delta and acceleration coefficients is improving the accuracy of the system (see detailed description in Section 6.3). The exception is for 2 and 3 state MFCC models which on the other hand can be caused by the absence of delta and acceleration of MFCC's zero coefficient. The results of the best models from each feature type and for each state number are depicted in Fig. 10. From all this we can see that MELSPEC feature type is the best choice for detection of two acoustic events, the glass breaking and a gunshot. This is also supported by the stability of the MELSPEC features through changing SNR as it is displayed in Fig. 11.

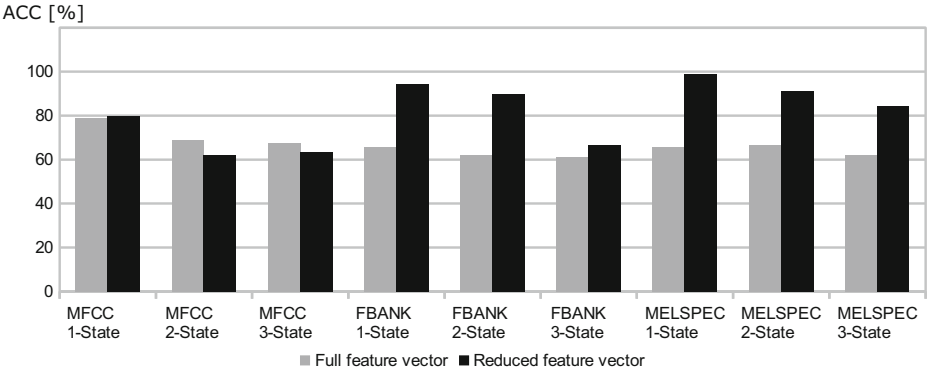


Fig. 10 The best models tested with full and reduced feature vector in changing SNR from all feature types and for all state numbers

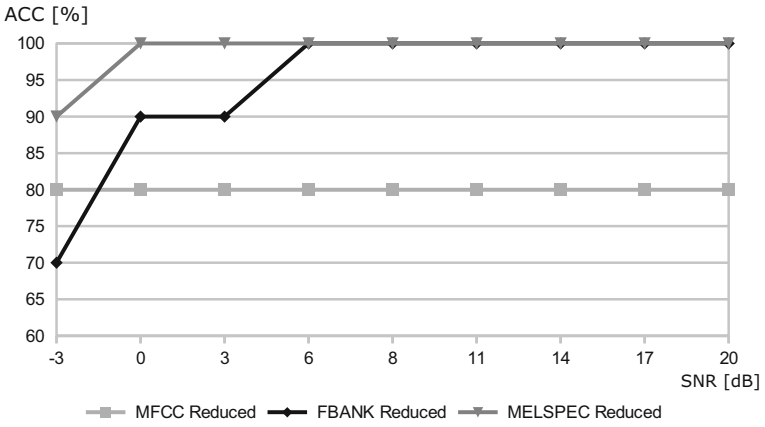


Fig. 11 Comparison of best models for each feature type and SNR

The best result $Acc = 98.89\ \%$ was obtained for MELSPEC reduced feature set for one state HMM with 256 PDFs. For this configuration we depicted the results using Detection Error Tradeoff - DET graph (see Fig. 12), where the insertion penalty was used as discrimination threshold in 20 additional tests.

6.7 Graphical user interface

To allow easy configuration of our system, we included a configuration manager using a configuration file (see Fig. 13). This way many of the parameters for front-end and decoder

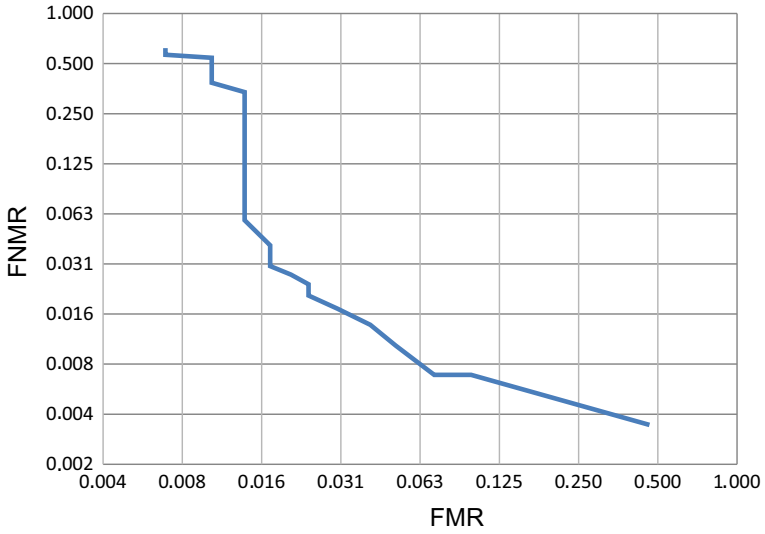


Fig. 12 Detection error tradeoff graph of the best model (MELSPEC reduced, 256 PDFs, 1 state) in logarithmic scale, where insertion penalty was used as discrimination threshold (FNMR - false non-match / missed alarm rate, FMR - false match / false alarm rate)

Fig. 13 Example of a configuration file setting parameter for feature extraction and detection

```
#setting front-end parameters
ACC_WND 2
ZERO_COEF T
ENERGY F
CMN_WND 0
DEL_WND 2
SKIP_OFFSET 13
FRONT_END_TYPE MFCC
MEL_NUM 29
CEP_NUM 12

#settings for decoder
INSERT_PENALTY -500
DURATION 100
INDEX_FILE ../test/1_1_0/index.txt
MODEL_FILE ../test/1_1_0/model.bin
```

can be set. The configuration parameters include important settings for MFCC, MELSPEC and FBANK feature extraction along with CMN, delta, and double delta (acceleration) computation. For decoder options the user is able to set the insertion penalty and decision time for Hypothesis evaluation and decoder control system. The configuration file also has to specify two additional important files, the model binary file and corresponding index file.

Binary model file contains WFST search network and Gaussian PDFs specification. In this file the output symbols on the network's transitions are numbers, with which a name needs to be associated. This is done by using the index file that contains a name of an acoustic event along with associated number in the search network. The same output name can be associated with different numbers, so the output would contain only desired names of events while each acoustic event can have more than one acoustic model.

This file also has one additional function. It contains information about which acoustic events are foreground and which are background for simultaneous classification and detection. Background models are distinguished by their name. They can either be all named "background" or if the user desires, different background models of the output hypothesis of background events can be named differently with a "background" prefix as depicted in Fig. 14.

```
<eps>                0
background           1
background\_street   2
background\_stadion  3
GunShot              4
GunShot              5
GlassBreaking        6
```

Fig. 14 Example of an index file (the $< eps >$ is naming empty transitions in the WFST search network, events with "background" prefix belong to the background models of the monitored area, all other events are considered foreground models)

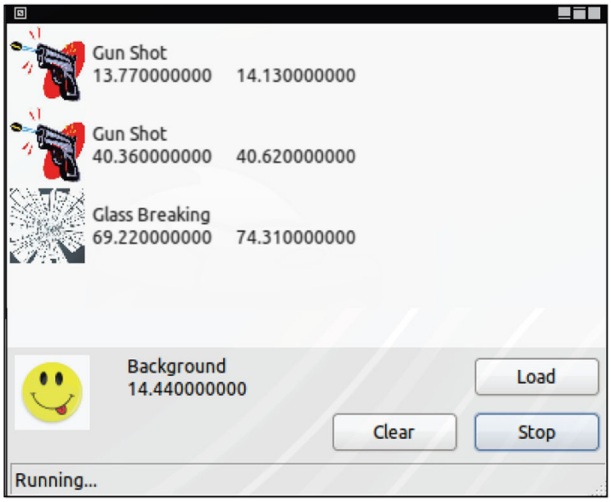


Fig. 15 Graphical user interface of the Acoustic Event Detection System (EAR-TUKE) with detected event and timestamp depicted

To provide an interactive user experience during operation, a small graphical user-interface is also provided. It displays the sequence of detected and classified acoustic events. As can be seen in Fig. 15, each detected acoustic event displays its name and the beginning and ending timestamp from the start of the system.

6.8 Live presentation of EAR-TUKE system

As we wrote about the effectiveness of our solution here are the computational requirements for running the system according to Fig. 16. The computer on the left generated acoustic events were (gunshots and breaking glass) and at the same time mixed in

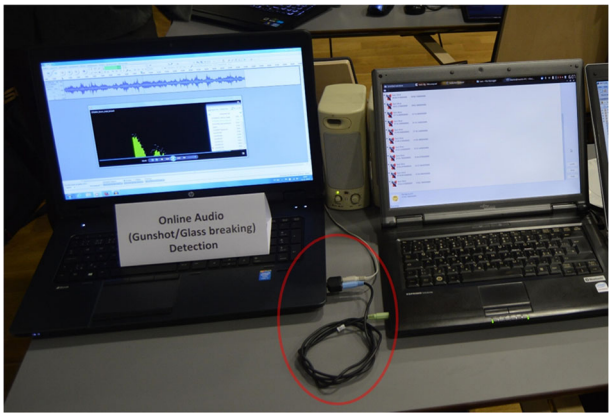


Fig. 16 Live presentation (MCSS 2014 Expo) of the EAR-TUKE system operated on the right-hand-side computer while events mixed with background noise were generated by the PC on the left. The connection of those computers by audio cable is marked by red circle

background noises with adjustable levels. The computer on the right operated the EARTUKE system detecting those generated events. The two computers were connected only by an audio cable from the sound card's output (left computer) to the microphone input (right computer). This connection is marked by the red circle (see video on `speetis.fei.tuke.sk/video/AEDlive.wmv`).

The target computer's configuration was $4GB$ of memory with an Intel Core 2 Duo T5450 processor operating at $1.66GHz$ clock rate. The system with graphical user interface operating was using 10 % of the processor cycles, and $35MB$ of memory with only one core active while detecting the two acoustic events.

7 Conclusion

In this paper we have described our system developed for acoustic event detection, the EARTUKE system built from scratch with no third party code/tool used. The system can provide the ability of long-term monitoring task and robustness in acoustic event detection in different environment SNR values. By description of the system development we have also presented our approach to acoustic event detection based on HMMs and utilizing WFST for supporting the extensibility and universality. We have also presented and implemented a method to increase robustness based on the basic coefficient removal from the input feature vector. The presented method has proven to be a better solution to robustness increase than the CMN in this particular task. For the computational profile enhancement, we have described the resource managing system for saving computational resources by continuous resetting of the system based on current hypotheses monitoring.

We can compare our best results of 98.89 % accuracy for mixed SNR test set (noisy environment, 256 PDFs, single state HMM with reduced feature vector, no CMN used) with other gunshot detection systems, for example one already described by Freie [7] (based on MFCC, LPC & HMM) which achieved 79.8 % for $20dB$ SNR, Gerosa [8] & Valenzise [30] (GMM, MPEG-7 & MFCC feature vector reduced) that achieved 93 % accuracy for 10dB SNR recordings (close to 99 % for clean recordings), and Uzkent [29] reported accuracy of 88.7 % for pitch range (PR) of non-speech sounds and the Autocorrelation Function using Support Vector Machines (SVM) with Gaussian kernel. Recently, Suman [28] also tested MFCC and three layer Artificial Neural Networks (ANN) classifier achieving 95 % for clean and 85 % for noisy recordings. As we can see, our system is comparable with other systems described in the relevant papers using inexpensive and easy to deploy components (common outdoor microphone or existing noise monitoring stations could be used).

In our next research we plan to broaden the database and test the system in real conditions if the planned cooperation with security forces is successful. Nevertheless, we have done cross-validation of our initial tests to be sure that the test set used achieves the most challenging results, so the well chosen recordings were then mixed more times with different SNR levels into background recordings and it was used for all next test setups.

Unfortunately, it was complicated during the development for us to acquire new gunshot recordings from other types of guns (our database contains 4 types of guns recorded in different environments and distances from the microphone, and we were not able to achieve new ones during the INDECT project). In the end, we have done two live demonstrations of the system (during public Expo of the international conferences MCSS 2013 and MCSS 2014 in Krakow) where it obtained positive reactions from the attendees, and the

live test provided a possibility to generate different randomly chosen sounds/events (also using external microphone or recordings such as trash can slam, car door slam, siren, car horn, etc.) with different SNR (using different volumes of background and events).

As we have stated previously, although the system is focused on acoustic event detection, it can be used in other tasks that are very similar, such as continuous monitoring of marine mammals and human related activities from hydrophone data [26]. Using WFSTs in our approach, any additional statistical information can be introduced to the search network along with any type of the HMM topology. This also allows the usage of the system in applications like keyword spotting, where a small search space is also used.

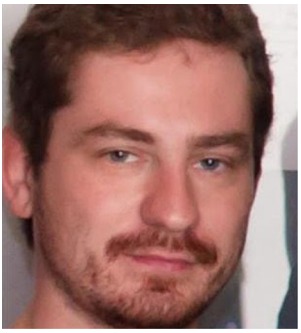
The future work will include more robustness, accuracy and speed improvements along with new feature extraction implementations including unused MPEG-7 descriptors.

Acknowledgements This publication is supported partially (50 %) by the Project implementation: University Science Park TECHNICOM for Innovation Applications Supported by Knowledge Technology, ITMS: 26220220182, supported by the Research & Development Operational Programme funded by the ERDF & partially (40 %) by EU ICT project INDECT (FP7 - 218086) and MINEDU (10 %) project VEGA 1/0075/15.

References

1. Alam MJ, Ouellet P, Kenny P, O'Shaughnessy D (2011) Comparative evaluation of feature normalization techniques for speaker verification. advances in nonlinear speech processing. Springer, Berlin Heidelberg, pp 246–253
2. Brown G, Pocock A, Zhao MJ, Luján M (2012) Conditional likelihood maximisation: a unifying framework for information theoretic feature selection. J Mach Learn Res 13(1):27–66
3. Collobert R, Bengio S (2001) SVMTorch: support vector machines for large-scale regression problems. J Mach Learn Res 1(2):143–160
4. Dixon PR, Hori C, Kashioka H (2012) A comparison of dynamic WFST decoding approaches. In: IEEE international conference on acoustics, speech and signal processing (ICASSP), Kyoto, pp 4209–4212
5. Eyben F, Weninger F, Gross F, Schuller B (2013) Recent developments in openSMILE, the munich open-source multimedia feature extractor. In: Proceedings 21st ACM international conference on multimedia (MM). ACM, Barcelona, pp 835–838
6. Foote JT (1997) Content-based retrieval of music and audio. In: Proceedings SPIE 3229, multimedia storage and archiving systems II, pp 138–147
7. Freire IL, Apolinario JA Jr (2010) Gunshot detection in noisy environments. In: 7th international telecommunications symposium (ITS), Manaus, pp 1–4
8. Gerosa L, Valenzise G, Tagliasacchi M, Antonacci F, Sarti A (2007) Scream and gunshot detection in noisy environments. In: 15-th European signal processing conference (EUSIPCO-07), Sept. 3–7, Poznan, pp 1–5
9. Hladek D, Ondas S, Stas J (2014) Online natural language processing of the Slovak language. In: CogInfoCom 2014: 5th IEEE international conference on cognitive infocommunications, Vietri sul Mare, pp 315–316
10. Kiktova E, Lojka M, Juhar J, Cizmar A (2014) Comparison of feature selection algorithms for acoustic event detection system. In: Proceedings ELMAR - international symposium electronics in marine, Zadar, pp 47–50
11. Kiktova-Vozarikova E, Juhar J, Cizmar A (2015) Feature selection for acoustic events detection. Multimed Tools Appl 74(12):4213–4233
12. Lamere P, Kwok P, Gouvea E, Raj B, Singh R, Walker W, Warmuth M, Wolf P (2003) The CMU SPHINX-4 speech recognition system. In: IEEE international conference on acoustics, speech and signal processing (ICASSP), Hong Kong, pp 2–5
13. Lee A, Kawahara T (2009) Recent development of open-source speech recognition engine Julius. In: Proc. Asia-Pacific signal and information processing association, annual summit and conference. APSIPA ASC, Sapporo, pp 131–137
14. Lojka M, Juhár J (2010) Fast construction of speech recognition network for slovak language. J Electr Electron Eng 3(1):111–114

15. Lojka M, Pleva M, Juhar J, Kiktova E (2013) Modification of widely used feature vectors for real-time acoustic events detection. In: Proceedings ELMAR - international symposium electronics in marine, Zadar, pp 199–202
16. Lopatka K, Kotus J, Czyzewski A (2011) Application of vector sensors to acoustic surveillance of a public interior space. Arch Acoust 36:851–860
17. Lopatka K, Czyzewski A (2014) Acceleration of decision making in sound event recognition employing supercomputing cluster. Inf Sci 285:223–236
18. Mckinney M, Breebaart J (2003) Features for audio and music classification. In: Proceedings international symposium on music information retrieval, Baltimore, pp 151–158
19. Mohri M, Pereira FCN, Riley M (2008) Speech recognition with weighted finite-state transducers. Springer Handbook of Speech Processing:1–31
20. Pleva M, Lojka M, Juhar J (2012) Modified viterbi decoder for long-term audio events monitoring. J Electr Electron Eng 5(1):195–198
21. Pleva M, Lojka M, Juhar J, Vozarikova E (2012) Evaluating the modified viterbi decoder for long-term audio events monitoring task. In: Proceedings ELMAR - international symposium electronics in Marine, Zadar, pp 179–182
22. Pleva M, Vozarikova E, Dobos L, Cizmar A (2011) The joint database of audio events and backgrounds for monitoring of urban areas. J Electr Electron Eng 4(1):185–188
23. Povey D, Ghoshal A, Boulianne G, Burget L, Glembek O, Goel N, Hannemann M, Motlicek P, Qian Y, Schwarz P, Silovsky J, Stemmer G, Vesely K (2011) The Kaldi speech recognition toolkit. In: Proceedings ASRU - IEEE workshop on automatic speech recognition and understanding, Hawaii, pp 1–4
24. Rabiner L (1989) A tutorial on hidden markov models and selected applications in speech recognition. Proc IEEE 77(2):257–286
25. Rusko M et al (2014) Slovak automatic dictation system for judicial domain. In: Human language technology challenges for computer science and linguistics, Springer International Publishing, LNAI 8387, pp 16–27
26. Sattar F, Driessen PF, Page WH (2013) Automatic event detection for noisy hydrophone data using relevance features. In: Proceedings pacific RIM conference on communications, computers, and signal processing, Victoria, pp 383–388
27. Schliep A, Georgi B, Rungsarityotin W, Costa I, Schonhuth A (2004) The general hidden markov model library: analyzing systems with unobservable states. In: Proceedings of the Heinz-billing-price, pp 121–135
28. Suman P, Karan S, Singh V, Maringanti R (2014) Algorithm for gunshot detection using mel-frequency cepstrum coefficients (MFCC). In: Proceedings ninth international conference on wireless communication and sensor networks Allahabad, Editors: Maringanti, R, Tiwari, M, Arora, A, LNEE 299, pp 155–166
29. Uzkent B, Barkana BD, Cevikalp H (2012) Non-speech environmental sound classification using svms with a new set of features. Int J Innov Comput, Inf Control (ICIC) 8(5):3511–3524
30. Valenzise G, Gerosa L, Tagliasacchi M, Antonacci E, Sarti A (2007) Scream and gunshot detection and localization for audio-surveillance systems. In: Proceedings IEEE conference on advanced video and signal based surveillance - AVSS, London, pp 21–26
31. Vozarikova E, Juhar J, Cizmar A (2011) Acoustic events detection using MFCC and MPEG-7 Descriptors. In: Multimedia communications, services and security. Springer, CCIS 149, pp 191–197
32. Vozarikova E, Lojka M, Pleva M, Juhar J, Cizmar A (2013) Comparison of different feature types for acoustic event detection system. In: Multimedia communications, services and security. Springer, CCIS 368, pp 288–297
33. Young S, Kershaw D, Odell J, Ollason D, Valtchev V, Woodland P (2006) The HTK book version 3.4. Cambridge University Press



Martin Lojka was born in Snina, Slovakia in 1984. In 2010 he graduated PhD. in study program Infoelectronics at the Department of Electronics and Multimedia Communications of the Faculty of Electrical Engineering and Informatics at the Technical University of Kosice. He works as a researcher in the field of speech technologies, telecommunications, speech recognition, acoustic modeling, Viterbi decoding, finite state transducers, etc. He has published over 30 technical papers in journals and conference proceedings.

Matúš Pleva was born in Kosice, Slovakia in 1977. In 2010 he graduated PhD. in study program Telecommunications at the Department of Electronics and Multimedia Communications of the Faculty of Electrical Engineering and Informatics at the Technical University of Kosice. He works as a researcher in the field of the acoustic modeling, acoustic event detection, speaker recognition, speech processing, human-machine interaction, security & biometrics, networking, etc. He has published over 70 technical papers in journals and conference proceedings.

Eva Kiktová was born in Liptovsky Mikulas, Slovakia in 1984. In 2009 she graduated M.Sc. (Ing.) at the Department of Electronics and Multimedia Communications of the Faculty of Electrical Engineering and Informatics at the Technical University of Kosice. In 2013 she graduated PhD. at the same department in the study program Telecommunications. She works as a researcher in the field of the acoustic event detection and classification, speaker recognition and digital speech and audio processing.

Jozef Juhár was born in Poproc, Slovakia in 1956. He graduated from the Technical University of Kosice in 1980. He received Ph.D. degree in Radioelectronics from Technical University of Kosice in 1991, where he works as full Professor at the Department of Electronics and Multimedia Communications. He is author and coauthor of more than 200 scientific papers. His research interests include digital speech and audio processing, speech/speaker identification, speech synthesis, development in spoken dialogue and speech recognition systems in telecommunication networks. Prof. Juhár is a member of ISCA, AES and IEEE. He is a member of the editorial boards and reviewer of several international scientific journals.

Anton Čižmár was born in Michalovce, Slovakia in 1956. He graduated from the Slovak Technical University in Bratislava in 1980, at the Department of Telecommunications. He holds a Ph.D. degree in Radioelectronics from the Technical University of Kosice in 1986, where he works as a Full Professor at the Department of Electronics and Multimedia Communications. Now he works as a rector of the Technical University of Kosice. He is author and co-author of more than 170 scientific papers. His scientific research areas are broadband information and telecommunication technologies, multimedia systems, telecommunication networks and services, 4. generation mobile communication systems and localization algorithms.