Graz University of Technology

Masters's Thesis

Acoustic Event Detection of General Sounds

conducted at the
Signal Processing and Speech Communications Laboratory
Graz University of Technology, Austria

by
Michael Peitler, 0321750

Supervisor:
Franz Pernkopf

Graz, October 11, 2016

Affidavit

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources. The text document uploaded to TUGRAZonline is identical to the present master's thesis.

date	(signature)

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommene Stellen als solche kenntlich gemacht habe. Das in TUGRAZonline hochgeladene Textdokument ist mit der vorliegenden Masterarbeit identisch.

Graz, am	(Unterschrift)

Abstract

While CCTV systems are widely used for monitoring public spaces, the information provided by the acoustic domain is often neglected. An acoustic event detection for general sounds can detect security-relevant occurrences like people screaming for help, a gunshot or an explosion by processing the audio signal captured from microphones.
In this thesis, acoustic event detection systems are introduced based on the theory of human auditory perception. The techniques for audio signal pre-processing, feature extraction, machine learning and evaluation of the detection systems are explained. The combination and parametrization of these is analyzed with the goal of high event detection rates and a low number of false alarms. Event detection systems found in literature are introduced. A sound library collection is introduced providing training and test data.
In the last part, a system is proposed to detect security-relevant events in public spaces. Its objective is to distinguish between a human scream, normal human voice, gunshot, explosion, breakage of glass and the background sound. The feature set can be assembled of MFCCs, MP7 features and Teager Energy Operator based features. Frame-wise evaluation is done with suitable classification measures. Precision, recall, accuracy and the F_1-score are computed for each class. These measures are averaged for the whole system and the Acoustic Event Error Rate is computed.
In the course of the experiments, the best system configuration in terms of classification performance was found with the combined MP7+TEO feature set, maximum-margin GMMs with 128 components and a frame length of 200ms. This resulted in an F_1-score of 0.74 and an Acoustic Event Error Rate of 0.24.

Kurzfassung

Während Videokamerasysteme zur Überwachung von öffentlichen Plätzen bereits weit verbreitet im Einsatz sind, wird die akustische Domäne nachwievor selten dafür benutzt. Ein System zur Detektion von allgemeinen Geräuschen kann sicherheitsrelevante Ereignisse wie Hilfeschreie, Schüsse oder Explosionen aus Mikrofonsignalen erkennen.
In dieser Arbeit werden Systeme zur akustischen Detektion von allgemeinen Ereignissen basierend auf der Theorie der auditorischen Wahrnehmung von Menschen erklärt. Techniken zur Vorverarbeitung und Merkmalsextraktion, maschinellem Lernen und Evaluierung solcher Systeme werden vorgestellt. Deren Kombination und Parametrierung mit dem Ziel von hohen Detektionsraten bei möglichst kleiner Anzahl an Fehlalarmen wird analysiert.
Im letzten Teil wird ein System zur Detektion von sicherheitsrelevanten Ereignissen in öffentlichen Bereichen vorgestellt. Es sollen menschliche Schreie, normale menschliche Stimme, Schüsse, Explosionen, Glasbruch und das Hintergrundgeräusch voneinander unterschieden werden. Der Merkmalsvektor kann aus MFCCs, MP7 und Merkmalen basierend auf den Teager Energy Operator aufgebaut werden. Das System wird mit Hilfe von passenden Klassifikationsmaßen Frameweise evaluiert. Precision, Recall, Accuracy und F_1-score werden für jede Klasse berechnet. Diese Maße werden gemittelt und mit der Acoustic Event Error Rate für das gesamte System angegeben.
Während den Experimenten wurde die beste Systemkonfiguration in Bezug auf die Klassifikationsperformance mit dem MP7+TEO Merkmalsvektor, Maximum-Margin GMMs mit 128 Komponenten und einer Framelänge von 200ms gefunden. Das System erreicht einen F_1-score von 0.74 und eine Acoustic Event Error Rate von 0.24.

Acknowledgement

I'd like to take the opportunity to express my gratitude to my supervisor Franz Pernkopf for his continuous support throughout this thesis and previously conducted projects, his patience and straightforward help.

I also want to thank my colleagues, friends and family for their understanding and technical and moral support over the last years.

Contents

1 Introduction

With the increasing need of surveillance and monitoring of human areas, closed circuit television (CCTV) systems are very much accepted regardless of their impact on privacy. They are widely installed and used on a continuous, 24/7 setting requiring humans to constantly monitor the scenes happening. Studies [1] showed that in a typical CCTV installation only about 30% of all relevant incidents have been detected. With the help of automated analysis of the audio and video streams of the cameras, these mainly human issues are addressed and the overall detection rate is expected to rise [2]. The audio stream is usually only used to get more information on a scene detected via video imaging, but acoustic event detection systems can lead to more detections of relevant occurrences, faster response time and relieving the need of human staff.

The key components of an acoustic event detection system are a single or multiple microphones, the subsequent analog-to-digital converter and a digital processing unit like a digital signal processor (DSP), PC or embedded system which processes the audio data captured and produces the detection result, as seen in Figure 1.1.

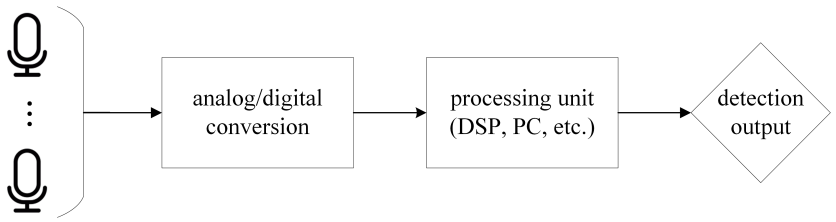

Figure 1.1: Basic acoustic event detection system components.

In the processing unit, digital signal processing and machine learning algorithms are used to analyze the input audio data and process this information to an understandable output. The acoustic detection of general sounds belongs to the field of *computational auditory scene analysis (CASA)*. Wang [3] defines *CASA* as "the study of auditory scene analysis by computational

means". It is derived from *auditory scene analysis* the human's ability to perceptually analyze a specific environment with the auditory system. The goal of *CASA* can be formulated as to design a machine which achieves the same performance as a human in *auditory scene analysis*. There are no restrictions in how to perform this task, e.g. it is not mandatory to fully reconstruct the behavior of the human auditory system.

1.1 Scope of the Thesis

In the course of this work an acoustic event detection system for the detection of general sounds is developed. A framework for acoustic event detection was setup in MATLAB for conducting experiments and evaluations. The desired purpose of the system is to detect security-relevant events in public spaces like streets, town centers, parking lots, train stations, etc.

Five events shall be classified and distinguished from the background class:

- **Human Voice**
  Human activity can be detected by the presence of human voice. The detection system can be used to monitor public spaces and report intrusions in restricted areas.
- **Human Scream**
  In an emergency situation, people will scream and shout with highly agitated voice. Such screams shall be detected, regardless of the actual words spoken to minimize context-dependency and overcome language barriers.
- **Gunshot**
  In case of a shooting, immediate intervention through authorities is required. The event detection system should help to alert the responders and speed up the reaction time.
- **Explosion**
  The rare event of an explosion, caused by mechanical failures or even terrorist attacks, shall be detected to subsequently alert emergency responders.
- **Breakage of Glass**
  It is very likely that a burglary begins with the breakage of glass. This sound can be detected acoustically to trigger an alarm instead of using dedicated vibro-electric glass break detectors.
- **Background**
  The background class models the typical background sound of the system's environment when no event to be detected is present. The system should not trigger an alarm in this case.

The system is designed for monophonic sound event detection (i.e. one event can be detected per time instance) and uses one audio channel. A signal pre-processing chain is implemented and the following feature extraction module supports MFCCs and some variants, MP7 temporal and spectral features and another type of feature based on the Teager Energy Operator. For classification, GMMs or SVMs can be used. Apart from traditional generative GMM parameter learning, a discriminative learning framework for optimizing the conditional-log-likelihood or the multi-class margin of the GMM is used first in this work for acoustic event detection. The trained models are validated using k-fold cross validation. Frame-based evaluation is conducted using class-wise computation of precision, recall, accuracy and the F_1 score. The numbers are also averaged for the whole system and the Acoustic Event Error Rate (AEER) is calculated.

1.2 Outline of the Thesis

In this thesis, the main aspects of *auditory scene analysis* and *computational auditory scene analysis* as well as their practical application and the derivation of acoustic event detection systems are presented in Chapter 2. Commonly used processing components, requirements and applicable techniques for acoustic event detection systems are introduced. Chapter 3 deals with audio signal features relevant for acoustic event detection. They are presented with event examples given. Chapter 4 is about classification, machine learning algorithms and evaluation of classifiers and event detection systems. In Chapter 5 the implementation of the system is presented. A custom sound library collection containing test and training data of the event classes described above is presented. Various experiments are conducted in a MATLAB simulation environment with different feature sets, classifiers, parameters and learning algorithms. The proposed system's performance is evaluated with the metrics introduced.
In the end, conclusive remarks are stated as well as an outlook to further work and improvements.

2 Fundamentals of Acoustic Event Detection

In this chapter the theoretical basics of acoustic event detections are explained. Based on the human's perceptive abilities it is explained which components are required to successfully detect events from audio signals.

2.1 Auditory Scene Analysis

In this section the auditory scene analysis model introduced by Albert Bregman [4] is explained. It describes the process how humans transform a sound mixture to meaningful occurrences. According to the model, this is achieved by auditory grouping in time and frequency domain.

2.1.1 Auditory Streaming

The perceptual grouping process by the auditory system is called streaming. According to Noorden [5] streaming depends on the difference between the frequencies of the individual tones and the time between their onsets. If the time between the onsets and the frequency difference is small, the occurrence is perceived as a single stream, otherwise the listener receives multiple streams. It is also possible to switch between these two possible forms of organization. Figure 2.1 shows alternating tone sequences with small (A) and large (B) frequency gaps between the high and low tones and the resulting perceptual organization in one or two streams, which are indicated by the dashed lines.

Auditory masking effects also have to be considered. The loudness threshold of the audibility of a certain sound may rise higher when another sound is present.

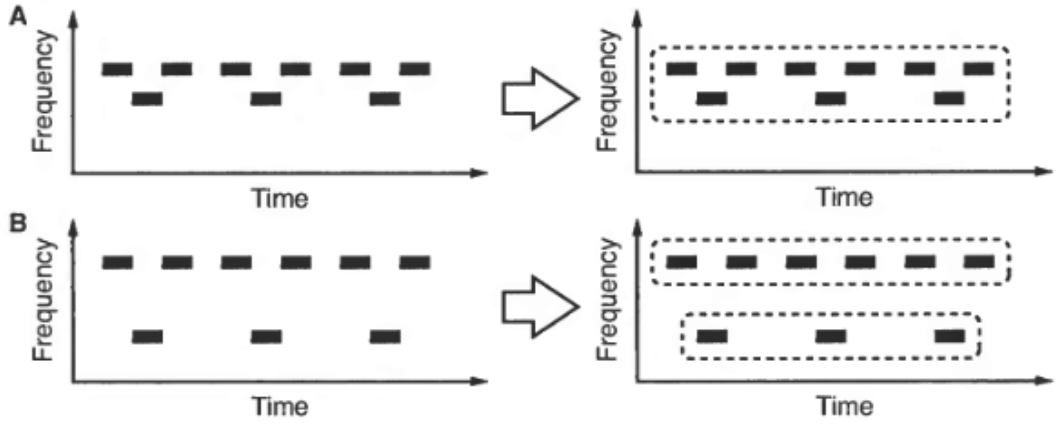

Figure 2.1: Perceptual organization of alternating tone sequences [3].

2.1.2 Perceptual Organization and Mechanisms

According to Wang et al. [3], Bregman [4] and Shamma et al. [6] the human hearing perception is organized by the following cues:

- **Proximity in Frequency and Time**
  Perceptual grouping occurs when several acoustic components are in close neighborhood in terms of frequency or time of occurrence. When the gap between their fundamental frequencies or the time between the occurrences is small then the grouping into the same stream is likely to occur.

- **Periodicity**
  If the individual sounds are harmonically related, they will be perceived as a single stream. For example, a musical chord is generally heard as one instance although it consists of multiple tones. This can be seen in Figure 2.2(a) and (b).

- **Continuous or Smooth Transition**
  When different and consecutive sound components form a continuous change of frequency over time, it is likely that they are grouped together in one stream. This is also the case if the changes are discontinuous but smooth. If they happen abrupt, the components will appear in different streams because this variant of change is an indication for another source. This kind of perceptual organization is also applicable for changes of pitch, intensity, spatial location and spectral shape.

- **Onset and Offset**
  The onset describes the beginning of a particular sound. An increase in energy, changes in the spectral distribution of the energy or in the case of musical notes a change in pitch can be the reason for an onset. The offset refers to the ending of a sound, the onset's properties described can be applied vice-versa. Grouping occurs when onset or, to a lesser extent, offset times of components are similar, i.e. they start and respectively stop at the same time.

- **Amplitude and Frequency Modulation**
  If frequency components experience the same temporal modulation, they will be perceptually grouped. This is applicable for amplitude as well as for frequency modulation.

- **Rhythm**
  Tones will also be grouped into one stream if they are rhythmically related. That means,

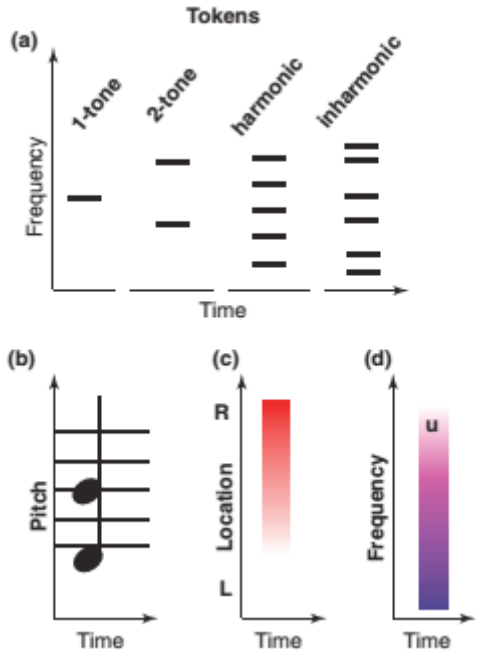

Figure 2.2: Principles and examples of auditory streaming [6].

if a rhythmic figure becomes repeated, the streaming is a kind of a sequential organization cue.

- **Common Spatial Location**
  If simultaneous sounds originate from the same location, the grouping is based on this commonality. Due to the fact that humans can separate sound sources even from a monaural mix there is a tendency to disregard this form of grouping. Figure 2.2(c) shows a sound with a right-centered source.

- **Complex Sound Attributes**
  A sound can also have more complex attributes, like the sound of the vowel 'u' in Figure 2.2(d). It incorporates the typical spectral structure of a vocal sound. Other tokens may be the high diffusivity of sound in a large and reverberant hall or the sound of a large choir singing in unison.

Apart from these low-level grouping cues the human auditory system also relies on high-level grouping cues too [7]. These are basically the experiences a listener has got. Depending on the language and culture amongst others the perception is likely to be different, leading to e.g. perceptive restoration of missing phonemes.

2.2 Computational Auditory Scene Analysis

The goal of *computational auditory scene analysis* (CASA) [3] can be formulated as to design a machine which achieves the same performance as a human in *auditory scene analysis*. There are no restrictions in how to perform this task, it is not mandatory to reconstruct the behavior of the human auditory system. Usually, one or more microphones are in use to record the scene.

Figure 2.3 shows the basic architecture of a CASA system. The first step is to peripherally analyze the input signal resulting in a time-frequency representation of the time-domain signal.

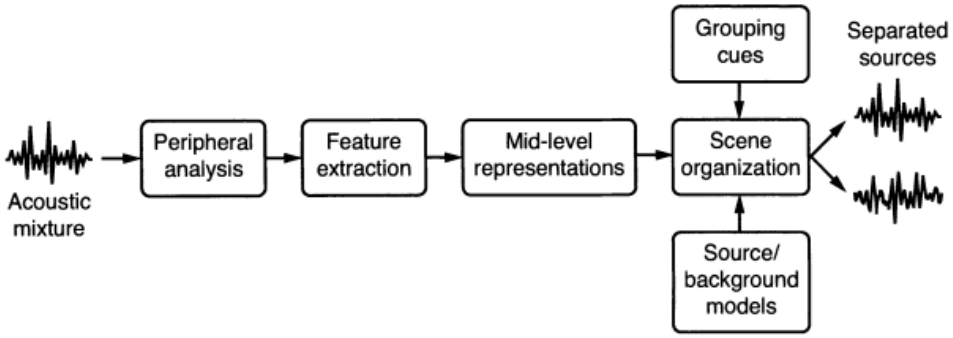

Figure 2.3: CASA system architecture [3].

Commonly the cochleagram is used, a representation of the auditory periphery which models the characteristics of the auditory periphery. In contrast many CASA systems tend to perform a time-frequency transform without auditory modeling using the short-time Fourier transform and use auditory properties where applicable while the feature extraction process. There, e.g. a Mel-scaled filter bank and/or logarithmic frequency spacing is used. Depending on the task specific acoustic features are extracted from the input signal. The next step is to create mid-level representations like segments, before attempting to organize the scene using grouping cues and trained models of the source and background to result in separated source streams.

2.2.1 Applications of CASA

Apart from acoustic event detection systems, the CASA theory is applicable to a wide range of speech- and sound-related research topics [3]:

- **Automatic Speech and Speaker Recognition (ASR)**
  Many speech and speaker recognition systems suffer from the problem of degrading performance when background noise or other interferring sounds in addition to the desired speaker are present. From the CASA point of view, the relevant voice is only one sound in a mixture which has to be dealt with. So far, such systems use source separation techniques to preprocess the input signal and ideally receive only the speaker's voice at the recognition system's input.

- **Hearing Aids**
  The typical perceptual consequence of hearing impairment is the loss of sensitivity, which results in an increased threshold level for understandability and a reduced dynamic range. Other problems are reduced frequency selectivity, increased susceptibility to background noise and impaired binaural capabilities [8]. Similar to an ASR system humans with hearing difficulties have the problem of degraded understandability especially in noisy conditions. Basic hearing aids only offer a fixed or manually tunable amount of amplification but this is not enough for loud environments, where many speakers and other sound sources are present.

- **Music Information Retrieval (MIR)**
  MIR is an interdisciplinary science with the purpose of retrieving information from music. The research in music recommender systems was driven by businesses to help users of digital music services to easily find music similar to the tracks in their individual collections [9], [10]. Usually, automatic categorization of the collected tracks is performed before.

Another research field of MIR is automatic music transcription [11], where special emphasis is taken on polyphonic transcription of different, simultaneously played instruments [12].

- **Audio Information Retrieval**
  To simplify the time-consuming process of data labeling of large audio collections, algorithms can be used to automate this task. Recorded sounds from real-life scenes usually consist of multiple, mixed sources, so the first step is to separate them. The actual information retrieval is then executed individually for each source stream. The result is a polyphonic annotation of the scene captured by the recording.

- **Auditory Scene Reconstruction**
  Based on the received sound mixture the underlying acoustic scene can be reconstructed. The individual sources need to be separated and they have to be located in space. Systems can be created to present the scene to the listener with the options of enabling or disabling a specific source or displacing it.

2.3 Acoustic Event Detection Systems

In this section, acoustic event detection systems are introduced, which are derived from the CASA system architecture presented in Section 2.2. The basic topology, components and techniques are explained, as well as the requirements on audio data.

2.4 System Overview

Schuller [13] presents the typical architecture of an intelligent audio analysis system, which is used for the acoustic detection and classification of general events. It is shown in Figure 2.4 and can operate in the training/adaption and the normal operation mode. The operations indicated by the dashed lines are only in use during the system's training/adaption phase. The input of the system is the time-domain audio signal originating from an audio capture device (microphone) in the operational phase or a selected instance from the audio database during the training phase. The audio signal is subsequently preprocessed, apart from signal conditioning these step might include de-noising, de-reverberation or source separation. After that the low level descriptors (LLDs), also called the signal features (see Chapter 3) are extracted to obtain a low-dimensional feature vector describing the current input signal with segment sizes usually between 10 and 250ms. Due to the typical duration of an event individual frames are combined into longer chunks. In many acoustic event detection systems the signal features are hierarchically extracted, e.g. emotion detection systems at first verify the presence of human voice with a specific feature set and only then extract special speech emotion features. Then the feature vector can be reduced with e.g. principal components analysis (PCA) or linear discriminant analysis (LDA) to reduce model complexity. In the classification step a label is assigned to the signal instance. For improved accuracy the result can be fused with information from other sources like a video event detection system and finally the result can be encoded in a data format which is suited for integration of the detection result.

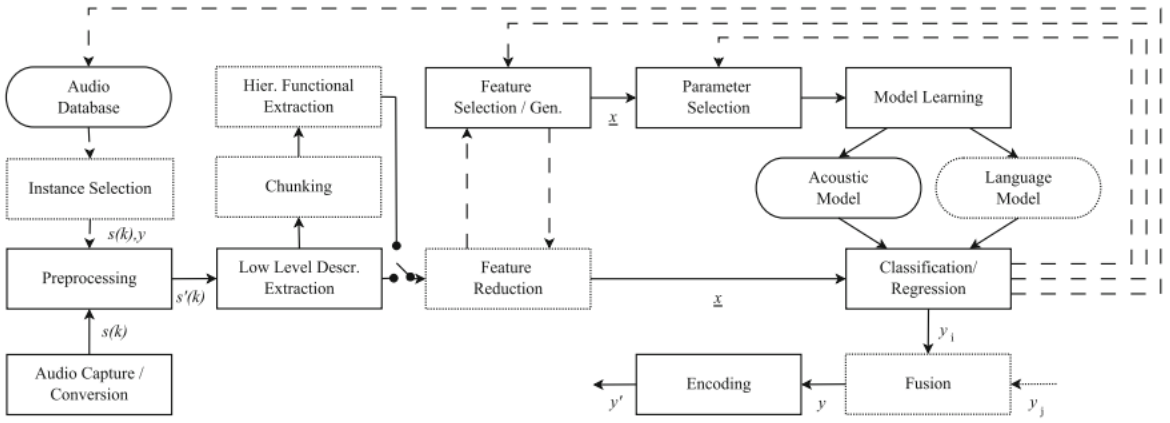

Figure 2.4: Intelligent audio analysis system overview [13].

2.4.1 Audio Database and Live Audio Capture Requirements

The availability of adequately prepared audio data is crucial for training an acoustic event detection system. There are also requirements on live audio capture during the normal operation of the system.

Technical Requirements on Audio Data

Generally, a sample rate between 16kHz (used by Ntalampiras et al. [14] and [15] or Chan et al. [16]) and CD quality with 44.1kHz (Clavel et al. [17]) is used, sharing a common 16bit resolution. The usage of a high-quality microphone is also recommended. Its acoustic overload point should be considered to be sufficient for the event detection system to avoid clipping. The noise floor level should be as low as possible to be able to capture a high dynamic range and a minimum amount of non-linearity should be introduced, i.e. a low THD value is favored. The microphone preamplifier and analog-digital converter, as well as all parts of the microphone input chain are required to comply with this demands. Recently, e.g. Giannoulis et al. [18] used multi-microphone arrays for audio capture which can be useful for event localization, leading to the requirement of multichannel capture and processing equipment with channels being identical, synchronized and drift-free.

Requirements on Audio Databases

In [13] the following criteria for sound libraries to be used for an audio analysis task are presented:

- **Quality**
  The audio data is expected to be realistic and adequate for the task. The capture conditions should be ideal and intended as well as known data degradation can be helpful to result in a robust system.

- **Quantity**
  As many sound samples as possible are required to successfully train, test and verify the event detection system. To achieve a high degree of generalization, sounds from many different sources with high diversity are necessary. The event classes should be realistically balanced and distributed.

- **Modelling**
  The data categorization must be reasonable for the task given and well-defined between different models.

- **Labelling**
  It should be unique, done by many labellers and come close to the ground truth (resulting in the 'gold standard' [19]).

- **Release**
  For the release of the sound library, easy accessibility is useful. The documentation of the recording and side conditions is mandatory.

2.4.2 Preprocessing

Apart from signal conditioning basic operations like level adaption or normalizations are usually applied. Typical preprocessing steps like denoising and dereverberation are commonly not applied in acoustic event detection systems. Heittola et al. [20] use non-negative matrix factorization (NMF) for separating individual sources in a multisource environment to deal with simultaneously occurring events individually. Figure 2.5 illustrates the process of dividing a mixture signal including several sources in separate signals, each of the resulting signals corresponding to one source.

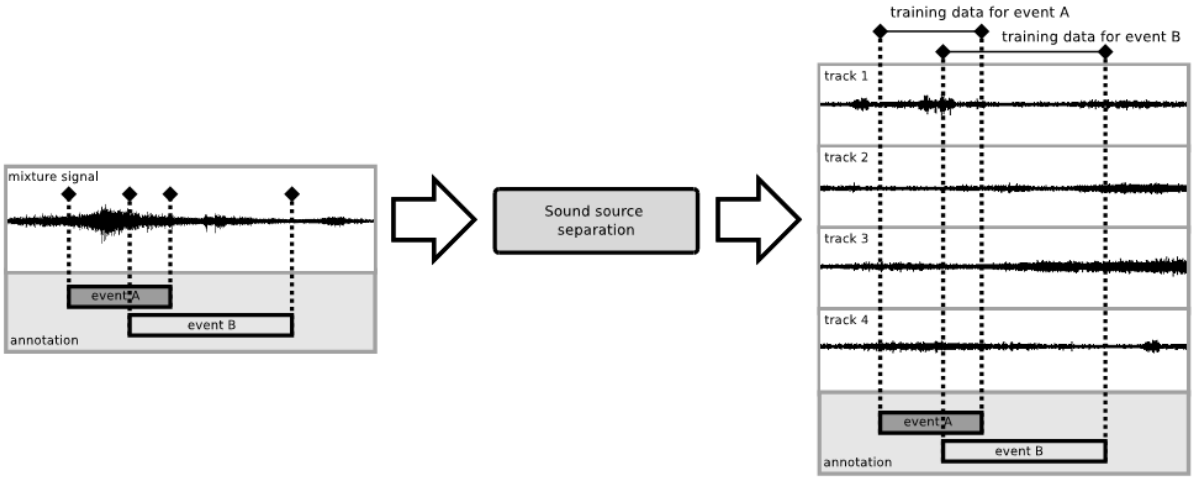

Figure 2.5: Source separation in multisource environments [20].

2.4.3 Feature Extraction, Selection and Reduction

The selection of a suitable feature set for the event to be detected has a huge impact on the classification performance of the event detection system. Experiments with different feature sets have been conducted and can be found in Section 5.7.1.

Especially in speech recognition Mel-frequency cepstral coefficients are widely used, e.g. recently in [21]. Many acoustic event detection systems also use them, mainly based on the success in speech recognition [22], [23], [24]. Selecting an appropriate feature set is crucial for high accuracy of the event classification. It depends on the events to detect and their context, so there is no optimal feature set for each detection task. Zhuang [25] found out that the extraction from features apart from MFCCs results in 30% more accuracy in the event classification. In contrast,

a comparison between MFCC and MPEG-7 feature sets [26] showed that MFCCs deliver better results in their experiments. Other signal features like the linear prediction coefficients (LPC) non-negative matrix factorization (NMF) based descriptors can also be used. For the task of feature reduction popular techniques like the PCA, Independent Component Analysis (ICA) or the LDA can be used [27], [28]. Lu [29] utilizes the PCA for dimensionality reduction in his acoustic event detection problem.

2.4.4 Supervised Learning and Classification

Acoustic event detection systems which have to be trained initially are based on supervised learning algorithms. Labeled training data as described above is required to make the system able to correctly assign detected events to their corresponding classes. This is desirable for scene analysis and reconstruction, where the specific event that has happened needs to be known.

As described earlier, an event detection system based on the supervised learning approach operates in two different phases: the learning, training or adaption phase and the operational or testing phase. At first the system's model will be trained and adapted according a given set of training and test data from the sound library as shown in Figure 2.6.

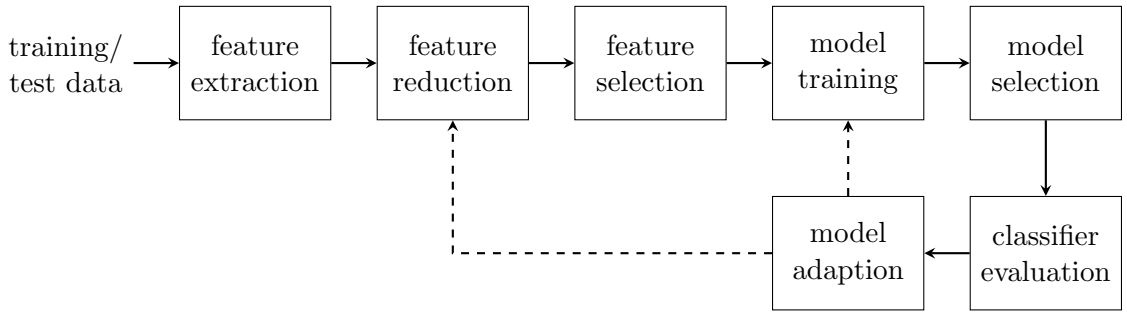

Figure 2.6: Training phase.

After successful selection of the trained system it can be used in practice to classify the sound events. As shown in Figure 2.7, the same feature extraction and classification methods are used as in the training phase. This mode is also used for model selection and the testing of the classifier.

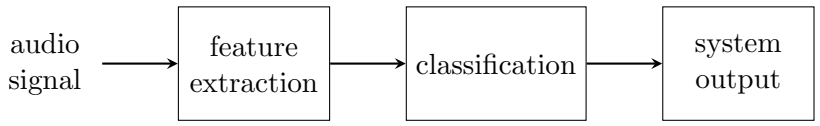

Figure 2.7: Operational phase.

Vuegen et al. [30] use a Gaussian Mixture Model (GMM, explained in Section 4.1) per event category for classification, the same approach is followed by Vozarikova et al. [31]. To overcome the limitations of generative model learning, SVMs can be used, e.g. Temko et al. [32]. In this

thesis a discriminative learning approach for GMMs is investigated in Section 4.1.4 and Section 5.7.2. Ellis [33] uses a multilayer perceptron neural network trained with back-propagation for the detection of alarm sounds. Because audio events can have a specific temporal structure, Hidden Markov Models (HMM) [34], [35] are used to exploit the temporal dependencies. Mesaros et al. [22] use three-state left-to-right HMMs with 16 Gaussians per state for event detection in selected real-world environments like office, street, restaurant, etc. Heittola et al. [23], [24] put special emphasis on the modeling of the background scenery and developed a context-dependent event detection system, where GMMs are used for context recognition and HMMs for the event detection itself.

2.4.5 Unsupervised Learning and Classification

The unsupervised approach does not require labeled training data, but in contrast to classification the objective of such a system is to group events and assign them to clusters. Which event exactly happened will not be determined, but e.g. the rate of occurrence of events can be determined to detect uncommon or unusual, seldomly happening events.

When using unsupervised learning algorithms, unlabeled data is used for training. The task is to group the data and exploit any structure in the data, i.e. the algorithm has to find the occurring classes by itself without prior knowledge. Schmalenstroeer et al. [36] compare two approaches for unsupervised learning of acoustic events. A sequential dynamic time warping algorithm is used versus k-means++ [37] clustering to classify audio events. Siddiqi et al. [38] and Ramois et al. [39] developed an algorithm to automatically discover the states required for training an HMM. It is shown that this unsupervised method has advantages over traditional Baum-Welch parameter estimation. Zhou et al. [40] use the Bayesian information criterion (BIC), Chua et al. [41] use compression and edit distance for unsupervised segmentation and clustering of audio streams. Giannoulis et al. [42] developed an NMF-based acoustic event detection baseline system where NMF decomposition is used to split unlabeled audio data in classes.

2.4.6 Model Validation

The goal of model validation is to predict how a model will generalize on an unknown, independent data set used in real-world application of the system. Due to the generally limited amount of data available, special techniques are required to optimize the model [27]. Giannoulis et al. [42] use the popular k-fold cross validation for model validation in their acoustic event detection system. In practical, cross validation is widely used in the scientific community. Clavel et al. [17] use the Bayesian Information Criterion (BIC) [27] for selecting the optimal model parameters in addition to cross validation.

2.5 Extended System Topologies

Widely implemented basic acoustic event detection system usually use one layer of classification, i.e. all classifiers are evaluated in parallel to obtain the most likely event. Ntalampiras et al. [14] showed that a multi-stage detection topology increases the number of correctly detected events as well as it reduces the false alarm probability. The system, shown in Fig. 2.8, distinguishes between sounds produced by humans or other sound sources. If the sound is human, it will be checked by a suitable voice activity detector if it is vocal. In this case, the vocal sound is distinguished between screamed or normal voice. Atypical sounds will be treated separately by GMMs.

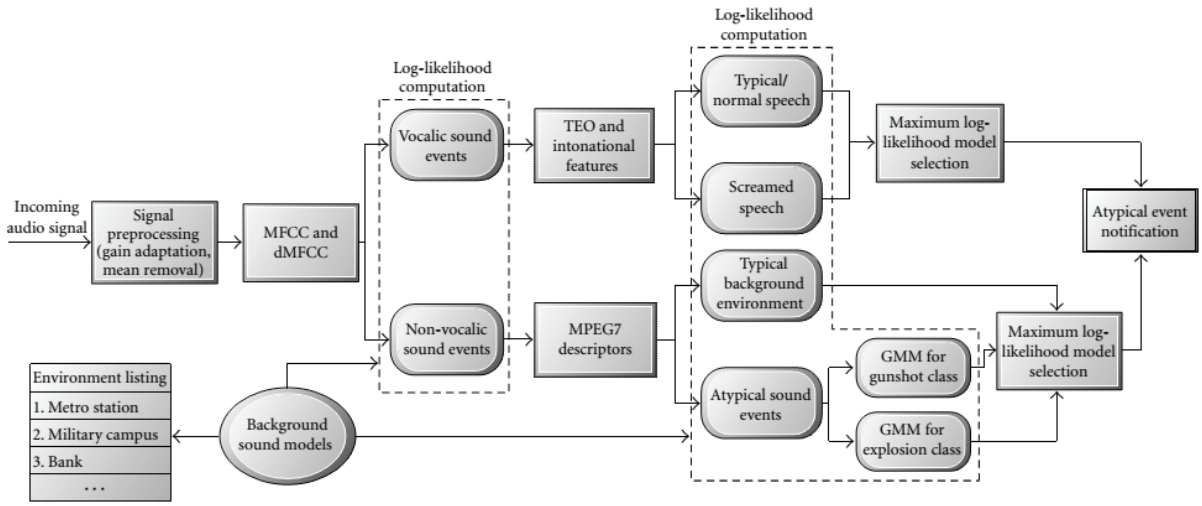

Figure 2.8: Multistage acoustic event detection system [14].

2.6 Fusion

The fusion of the information can be done after the final decision of the acoustic event detection system, e.g. an audio event can be fused with a video event. E.g. a scream detected from the audio signal will only be reported if human activity can be detected from the video stream to avoid false alarms. Systems can also be fused at an earlier stage. Sadlier et al. [43] present an SVM-based event detection system for field sports fusing at feature level with features extracted from audio and video streams. Ye et al. [44] introduce bi-modal codewords for representing combined audio-video features.

2.7 System Evaluation

The commonly known classification performance metrics like *precision*, *recall*, *accuracy* and the F_β score can be applied to the evaluation of an acoustic event detection system. For the evaluation of the *CLEAR 2007* challenge results [45], special measures for such systems have been introduced an therefore been used in other challenges, like the 2013 IEEE AASP Challenge on *Detection and Classification of Acoustic Scenes and Events (DCASE)*. These measures are explained in detail in Section 4.4.1. Based on them, metrics for polyphonic sound event detection were defined [46] for the recent *DCASE 2016* challenge.

2.8 Notation

In this section the basic notation in time and frequency domain is explained.

The Symbol $s(n)$ denotes the digital audio signal at the time instant n with the corresponding sampling frequency f_s. After segmentation into frames the signal vector at frame index m is written as $\mathbf{s}_m$ consisting of N_m in samples. In total, M frames are available. The time difference in samples between two successive frames is the hop size N_{hop}. Generally, bold letters denote matrices or vectors.

In frequency domain, corresponding uppercase letters are used for the spectrum. k is used as the bin index and N_{FT} is the length of the Fourier transform.

Some features are based on the short-term power spectral density estimation of overlapping time frames. The first step is to multiply the samples of each overlapping time frame with a window function $w(n)$. Applying the DFT leads to the complex Fourier spectrum of the m^{th} frame

$$S_m(k) = \sum_{n=0}^{N_{FT}-1} s(n + mN_{hop})w(n)\mathrm{e}^{-j\frac{2\pi nk}{N_{FT}}} \qquad 0 \leq m \leq M-1;\ 0 \leq k \leq N_{FT}-1. \tag{2.1}$$

According to Parseval's theorem, the energy of a time frame can be expressed by the magnitude-squared Fourier spectrum, i.e.

$$\hat{P}_m = \frac{1}{E_w} \sum_{n=0}^{N_m-1} \left|s(n + mN_{hop})w(n)\right|^2 = \frac{1}{E_w N_{FT}} \sum_{k=0}^{N_{FT}-1} \left|S_m(k)\right|^2, \tag{2.2}$$

scaled by the window's energy,

$$E_w = \sum_{n=0}^{N_m-1} \left|w(n)\right|^2. \tag{2.3}$$

Due to the fact that the Fourier spectrum is symmetric around the sampling frequency f_s, it is sufficient to only take the first half of it into account. Therefore, the power spectrum $P_m(k)$ of the m^{th} frame is then defined as

$$P_m(k) = \begin{cases} \frac{1}{E_w N_{FT}} \left|S_m(k)\right|^2 & \text{for } k = 0 \text{ and } k = \frac{N_{FT}}{2} \\ \frac{2}{E_w N_{FT}} \left|S_m(k)\right|^2 & \text{for } 0 < k < \frac{N_{FT}}{2} \end{cases}. \tag{2.4}$$

The frequency resolution, i.e. the distance between two bins is given as

$$\Delta f = \frac{f_s}{N_{FT}}. \tag{2.5}$$

Each bin index corresponds to a center frequency, this relation can be expressed by

$$f(k) = k \cdot \Delta f \qquad 0 \leq k \leq \frac{N_{FT}}{2}. \tag{2.6}$$

The bin index which contains a frequency f is

$$k(f) = \left[\frac{f}{\Delta f}\right] \quad 0 \leq f \leq \frac{f_s}{2}, \tag{2.7}$$

where $[\cdot]$ denotes the round-to-nearest integer operator.

3 Audio Signal Features

This chapter presents audio signal features suitable for an acoustic event detection system. The first section deals with selected features defined by the MPEG-7 standard. The most of them were already known before standardization. For distinguishing vocal emotions the Teager Energy Operator (TEO) is introduced, as well as the popular Mel-frequency Cepstral Coefficients (MFCCs) including some variants to compensate the effects of unwanted noise.

3.1 MPEG-7 Audio Protocol Descriptors

MPEG-7 standardizes a lot of descriptive audio features. In [47] the MPEG-7 audio features are presented, as well as [48] contains a complete documentation of MPEG-7 Audio and it also deals with audio content indexing and retrieval aspects. The equations seen in this section are from [47], unless marked differently. In the following sections, fundamental descriptors of MPEG-7 in time and spectral domain, as well as selected harmonicity features, are presented including exemplary applications of the features on the test signals of popular event classes.

This section deals with these MPEG-7 audio features:

- Basic Descriptors
  - Audio Waveform (AW)
  - Audio Power (AP)
- Basic Spectral Descriptors
  - Audio Spectrum Centroid (ASC)
  - Audio Spectrum Spread (ASS)
  - Audio Spectrum Flatness (ASF)
- Audio Harmonicity Descriptors
  - Harmonic Ratio (HR)

Upper Limit of Harmonicity (ULH)

Audio Fundamental Frequency (AFF)

3.1.1 Basic Descriptors

Audio Waveform (AW)

The AW feature is used to describe the waveform of the audio signal. It simply consists of two values: the minimum and maximum value of one frame of the time signal $s(n)$. The input signal is therefore splitted in consecutive, non-overlapping frames, i.e. $N_m = N_{hop}$. Consider a vector

$$\mathbf{s}_m = \begin{bmatrix} s(mN_{hop}) & s(1+mN_{hop}) & \cdots & s(N_{hop}-1+mN_{hop}) \end{bmatrix}^T$$

corresponding to the content of the m-th non-overlapping time frame. Then the audio waveform can be defined as a two-element vector

$$\mathbf{AW}_m = \begin{bmatrix} \min(\mathbf{s}_m) \\ \max(\mathbf{s}_m) \end{bmatrix}. \tag{3.1}$$

Examples Figure 3.1 shows the AW feature of a scream, gunshot, glass break and explosion sound. The input files have not been normalized. The amplitude of the scream is constantly decreasing after the onset, while the others show a more or less distinctive peak.

Audio Power (AP)

AP represents the signal's temporally smoothed energy of its non-overlapping frames, i.e.

$$AP_m = \frac{1}{N_{hop}} \sum_{n=0}^{N_{hop}-1} \left| s(n+mN_{hop}) \right|^2 \qquad 0 \le m \le M-1. \tag{3.2}$$

Examples Figure 3.2 shows AP examples of all classes. The scream in Figure 3.2(a) shows a pretty much constant audio power with an onset, similar to the AW feature. The other classes have one or more notable peaks.

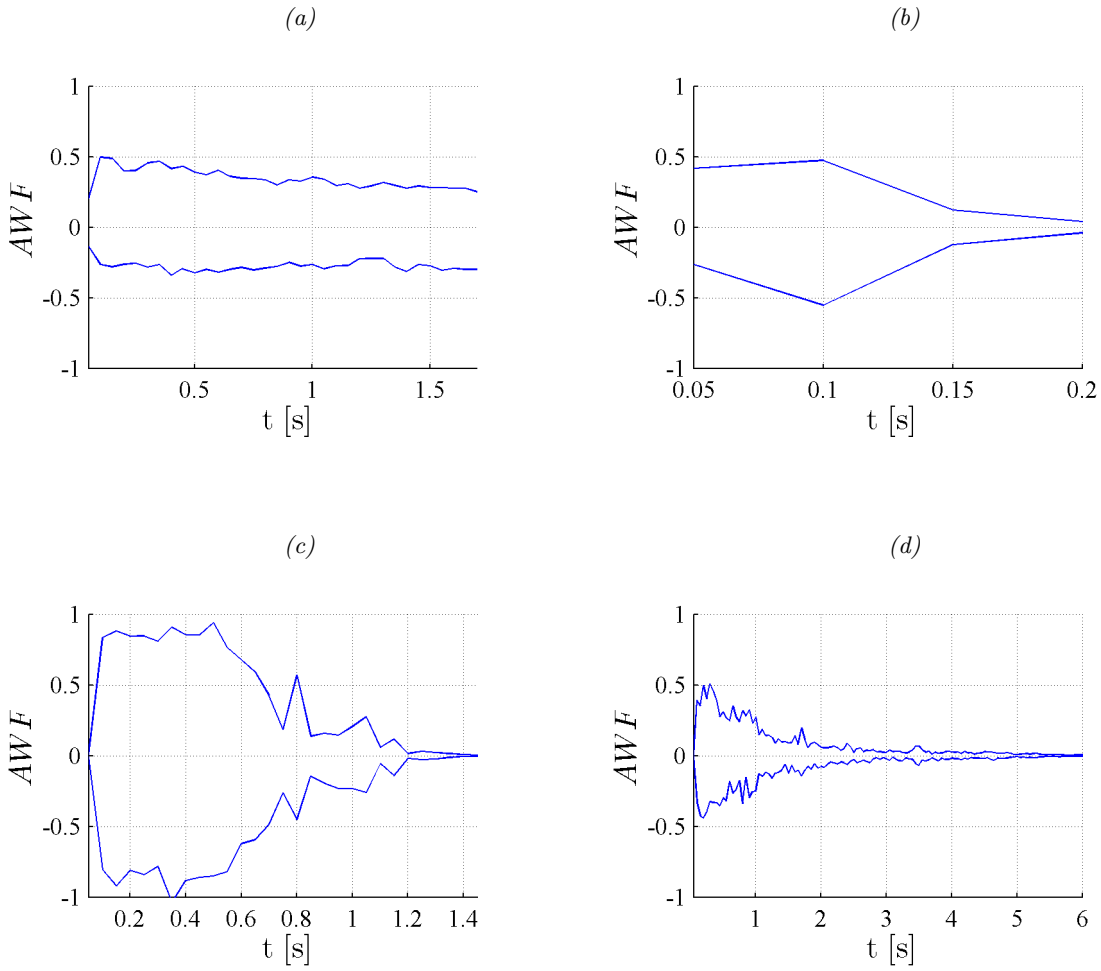

Figure 3.1: Audio waveform feature examples: (a) scream, (b) gunshot, (c) glass break, (d) explosion.

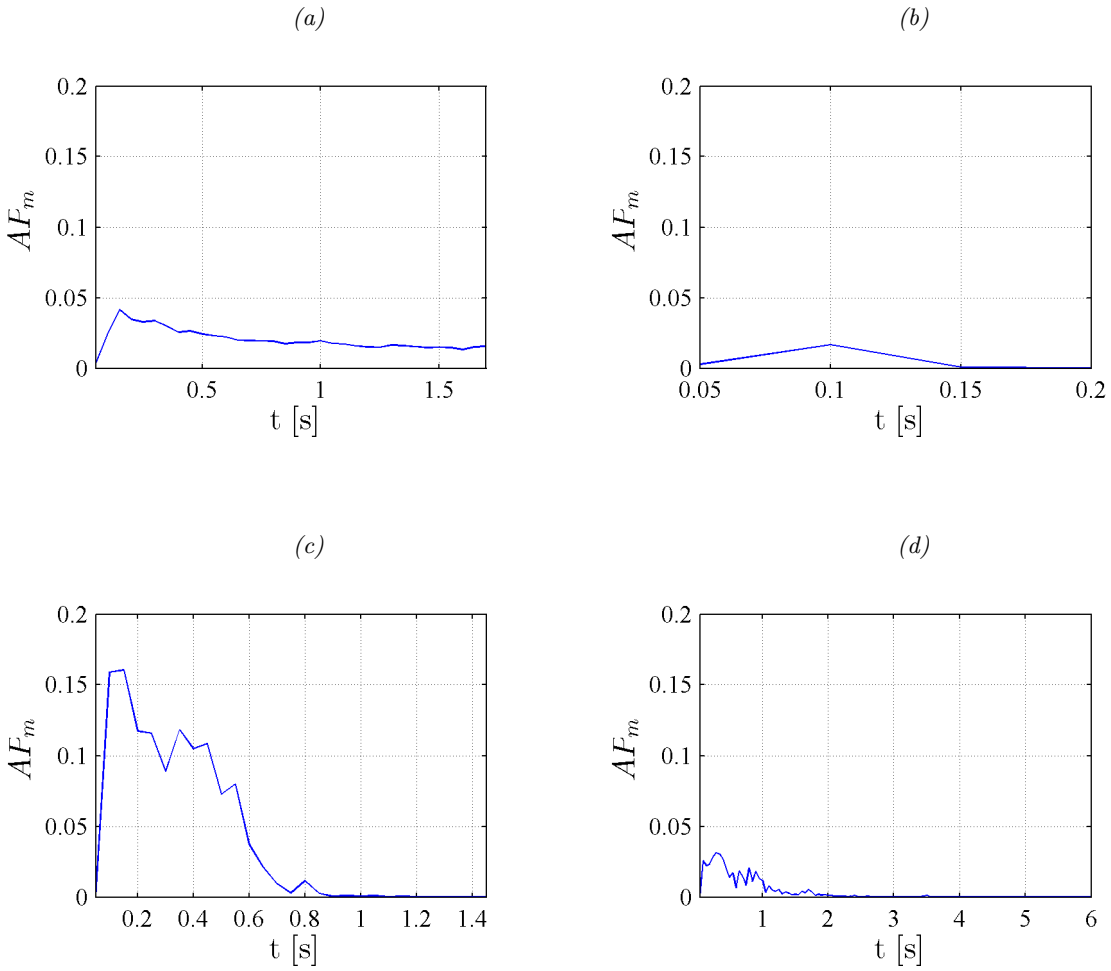

Figure 3.2: Audio power feature examples: (a) scream, (b) gunshot, (c) glass break, (d) explosion.

3.1.2 Basic Spectral Descriptors

Audio Spectrum Centroid (ASC)

The ASC is the center of gravity of the log-frequency weighted power spectrum of the input signal $\mathbf{s}_m$. For reducing the weight of very low frequencies and/or preventing a zero DC component, all power coefficients below 62.5Hz are summed into a single coefficient. That means, all bins with an index less or equal than

$$K_{low} = \left\lfloor \frac{62.5}{\Delta f} \right\rfloor, \tag{3.3}$$

where $\lfloor \cdot \rfloor$ denotes the floor operator.

According to this rule a new power spectrum $P'_m(k')$ is introduced,

$$P'_m(k') = \begin{cases} \sum_{k=0}^{K_{low}} P_m(k) & \text{if } k' = 0 \\ P_m(k' + K_{low}) & \text{for } 1 \leq k' \leq N_{FT}/2 - K_{low} \end{cases}. \tag{3.4}$$

The new center frequencies of the bins k' are defined as

$$f'(k') = \begin{cases} 32.25 & \text{for } k' = 0 \\ f(k' + K_{low}) & \text{for } 1 \leq k' \leq N_{FT}/2 - K_{low} \end{cases}. \tag{3.5}$$

The ASC of the m^{th} frame is defined as

$$ASC_m = \frac{\sum_{k'=0}^{N_{FT}/2-K_{low}} \left[\mathrm{ld}\left(\frac{f'(k')}{1000} \right) P'_m(k') \right]}{\sum_{k'=0}^{N_{FT}/2-K_{low}} P'_m(k')}, \tag{3.6}$$

where $\mathrm{ld}(\cdot)$ denotes the logarithm based on 2, i.e. $\log_2(\cdot)$.

The ASC indicates the domination of low or high frequencies in the audio signal. This information is related to the sharpness. To better approximate the human perception, the frequency scale used is logarithmically spaced. In literature, various other definitions of the spectral centroid can be found [49].

Examples According to the ASC the scream in Figure 3.3(a) and the gunshot are dominated by frequencies around 2 kHz leading to ASC values between zero and two. The sound of glass breaking varies in its centroid in Figure 3.3(c) and the explosion in Figure 3.3(d) is dominated by low frequencies as expected.

Audio Spectrum Spread (ASS)

The ASS is defined as the second spectral moment of the log-frequency power spectrum. In other words, this is the root-mean-square deviation of the log-frequency power spectrum from

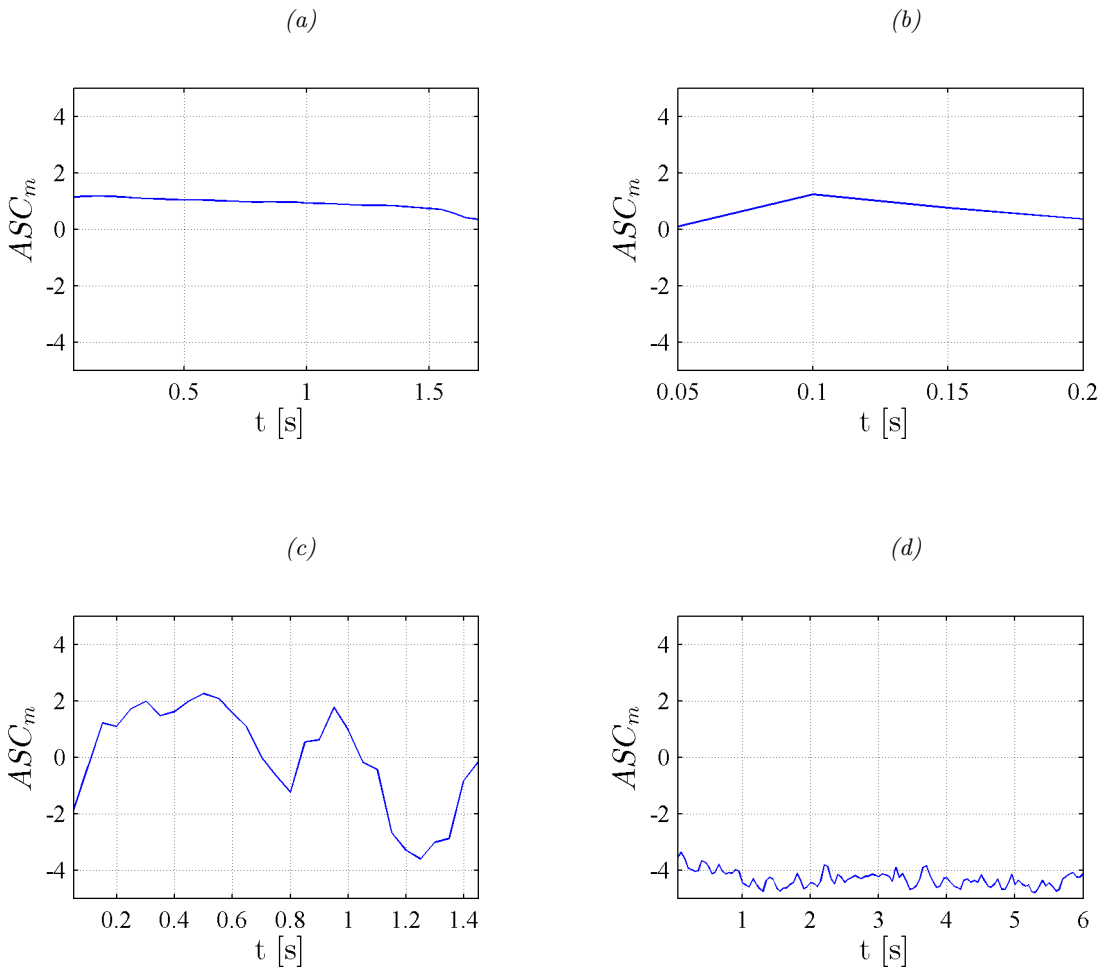

Figure 3.3: Audio spectrum centroid feature (ASC) examples: (a) scream, (b) gunshot, (c) glass break, (d) explosion.

its spectral centroid (ASC)

$$ASS_m = \sqrt{\frac{\sum_{k'=0}^{N_{FT}/2-K_{low}} \left[\mathrm{ld}\left(\frac{f'(k')}{1000}\right) - ASC_m\right]^2 P'_m(k')}{\sum_{k'=0}^{N_{FT}/2-K_{low}} P'_m(k')}}. \tag{3.7}$$

The ASS provides information about the spectrum's spread from its centroid. It can be used to distinguish tonal from non-tonal, noise-like sounds.

Examples The scream in Figure 3.4(a) results in very low values of the ASS. It is indeed a narrow-band sound, similar to the explosion's values in Figure 3.4(d) which is dominated by low frequencies, although they indicate a little more broadband spectrum. The ASS of the breakage of glass in Figure 3.4(c) is constantly varying and the highest ASS value is reached by the gunshot in Figure 3.4(b) because it is a broadband sound event.

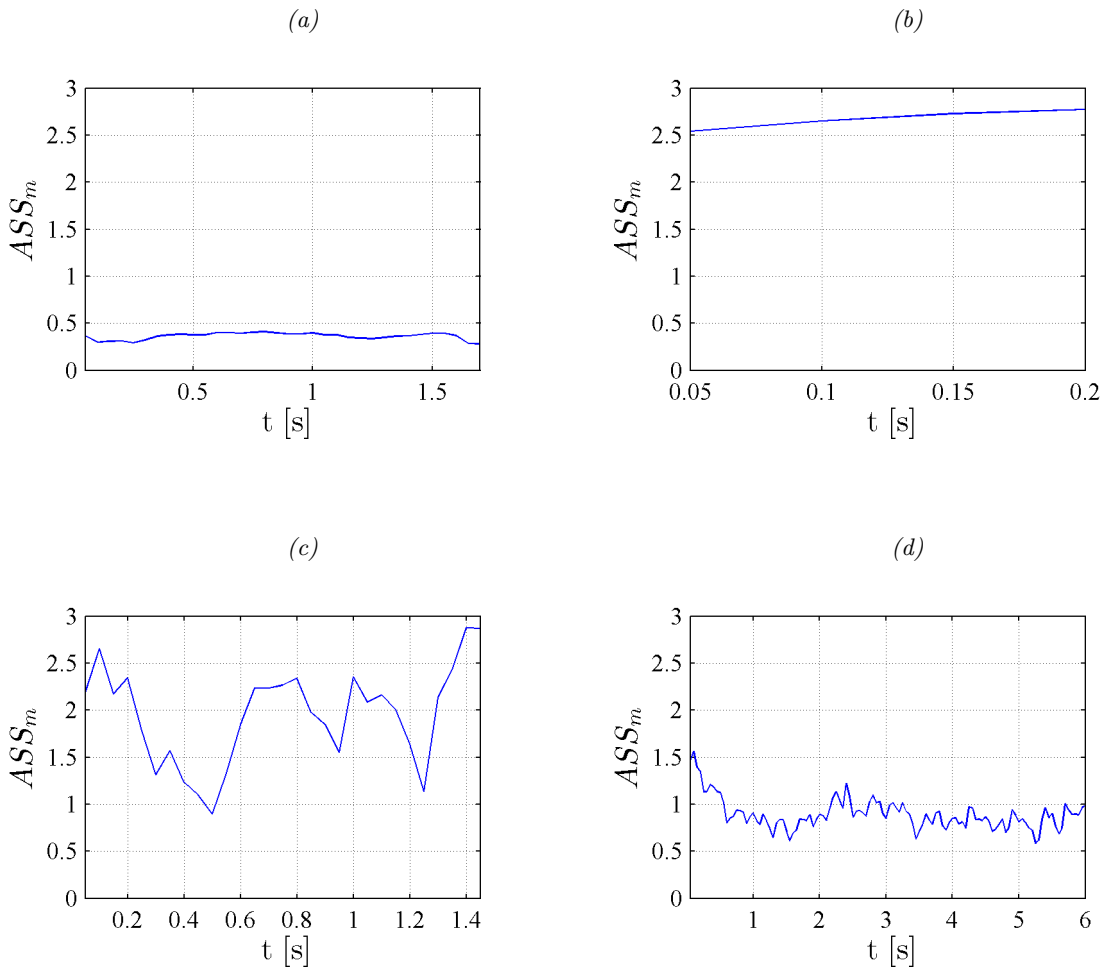

Figure 3.4: Audio spectrum spread feature examples: (a) scream, (b) gunshot, (c) glass break, (d) explosion.

Audio Spectrum Flatness (ASF)

The ASF quantifies the difference between the signal's spectrum and a completely flat spectrum. In other words, this is a measure for the similarity to white noise or the correlation of the signal. The calculation of the ASF involves the following steps:

1. Calculation of the power spectrum $P_m(k)$ of each non-overlapping 30ms long signal frame.

2. Partitioning of the spectrum in 1/4-octave-spaced log-frequency bands within the $[f_{lo}; f_{hi}]$ frequency interval, i.e. $f_{lo} = 2^{\frac{1}{4}\eta} \cdot 1\,\text{kHz}$ and $f_{hi} = 2^{\frac{1}{4}B} \cdot f_{lo}$.
   η is related to the lower band edge frequency, the recommend minimum is $\eta = -8$, which leads to 250 Hz. B equals the number of frequency bands and should be chosen in a way that f_{hi} is about the bandwidth of the signal and definitely not higher than $f_s/2$.

3. Because the ASF features can be sensitive to sampling frequency variations, the bands used in the calculation are made larger by 10 % so that they overlap by 5 % on the low border $f_{b,lo} = 0.95 \cdot f_{lo} \cdot 2^{\frac{1}{4}(b-1)}$, $1 \leq b \leq B$, and high border $f_{b,hi} = 1.05 \cdot f_{hi} \cdot 2^{\frac{1}{4}b}$, $1 \leq b \leq B$.
   The corresponding bin indices are $K_{b,lo} = \text{round}\big(f_{b,lo}/\Delta f\big)$ and $K_{b,hi} = \text{round}\big(f_{b,hi}/\Delta f\big)$.

4. For reducing computational costs and getting log-frequency spacing of the bands, a grouping procedure of $P_m(k)$ in $P_{m,g}(k')$ is defined:

   - Bands with $f_{hi} < 1\,\text{kHz}$ are taken as they are.
   - Bands with f_{hi} in the interval $[1\,\text{kHz}; 2\,\text{kHz}]$ are grouped in pairs, two successive values of $P_m(k)$ are replaced by their average.
   - Each group of bands between $\left[2^{\eta}\,\text{kHz}; 2^{\eta+1}\,\text{kHz}\right]$ with $\eta \geq 1$ are replaced by their arithmetic mean value.
   - At the last group at the upper frequency bound: if not more than 50% of the coefficients required are available, the last group is simply ignored.

   The new bands' edge indices of this grouped power spectrum are $K'_{b,lo}$ and $K'_{b,hi}$.

5. Finally, the ASF is calculated as

$$ASF_m(b) = \frac{\sqrt[K'_{b,hi}-K'_{b,lo}+1]{\prod_{k'=K'_{b,lo}}^{K'_{b,hi}} P_{m,g}(k')}}{\frac{1}{K'_{b,hi}-K'_{b,lo}+1} \sum_{k'=K'_{b,lo}}^{K'_{b,hi}} P_{m,g}(k')}. \tag{3.8}$$

6. To increase numeric precision, the logarithm can be applied to avoid very small numbers. Then the ASF is calculated as

$$ASF_m(b) = \frac{\exp\left\{\frac{1}{K'_{b,hi}-K'_{b,lo}+1} \sum_{k'=K'_{b,lo}}^{K'_{b,hi}} \ln\big(P_{m,g}(k')\big)\right\}}{\frac{1}{K'_{b,hi}-K'_{b,lo}+1} \sum_{k'=K'_{b,lo}}^{K'_{b,hi}} P_{m,g}(k')}. \tag{3.9}$$

An ASF of one means that the signal is equal to white noise and zero is equal to a fully correlated, sinusoidal signal.

Examples One can observe the typical harmonic structure of a human scream even in the ASF feature in Figure 3.5(a). The frequency bands dominated by the human voice show values close to zero, that means that they the signals are correlated, i.e. sinusoidal. The breakage of glass in Figure 3.5(c) also shows a lot of correlation in contrary to the gunshot in Figure 3.5(b) and the explosion in Figure 3.5(d). Table 3.1 shows the lower and upper frequency of each band.

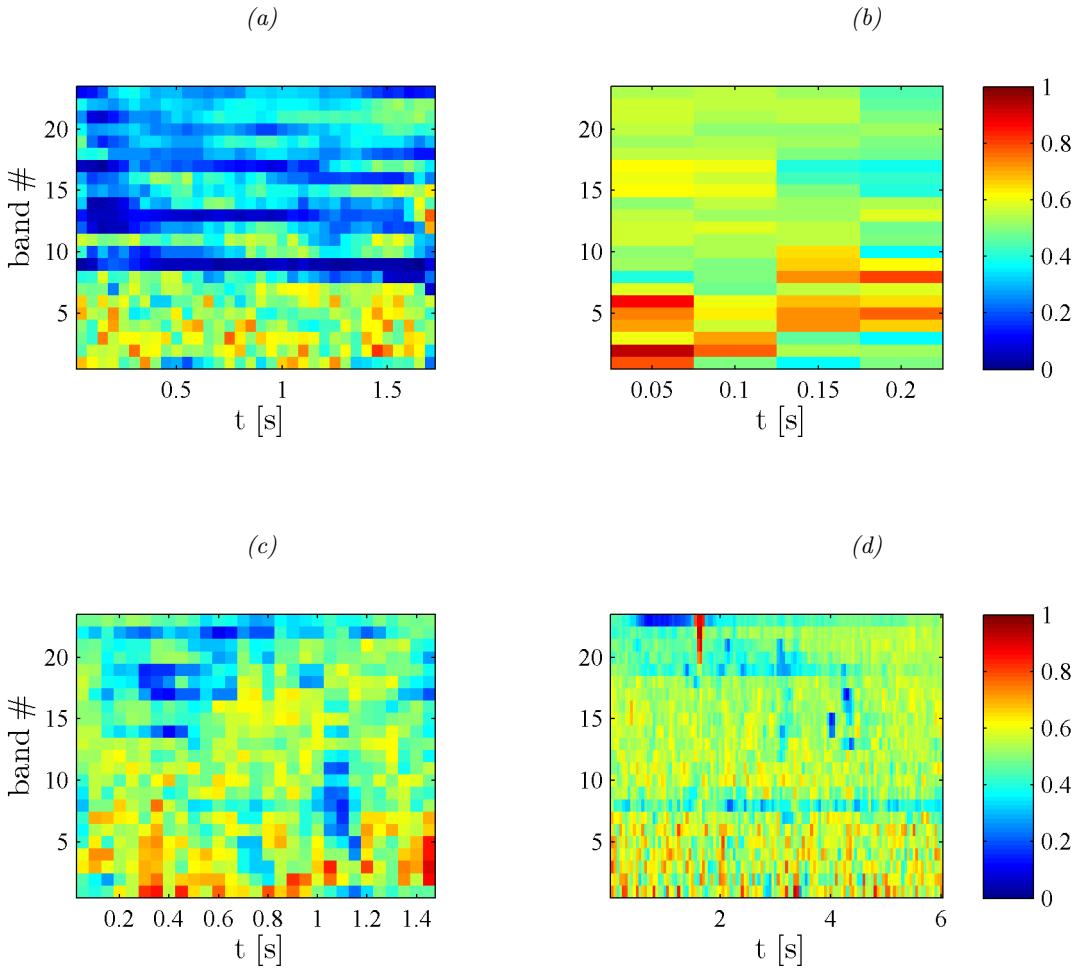

Figure 3.5: Audio spectrum flatness feature examples: (a) scream, (b) gunshot, (c) glass break, (d) explosion.

band	f_{lo} **[Hz]**	f_{hi} **[Hz]**
1	238	312
2	282	371
3	336	441
4	399	525
5	475	624
6	565	742
7	672	883
8	799	1050
9	950	1249
10	1130	1485
11	1344	1766
12	1598	2100
13	1900	2497
14	2259	2970
15	2687	3532
16	3195	4200
17	3800	4995
18	4519	5940
19	5374	7064
20	6391	8400
21	7600	9989
22	9038	11879
23	10748	14127

Table 3.1: Frequency ranges of the ASF bands.

3.1.3 Audio Harmonicity

Harmonic Ratio (HR)

The basic principle of the HR feature is the estimation of the normalized autocorrelation function $\Gamma_m(l)$ of the m^{th} frame of the input signal

$$\Gamma_m(l) = \frac{\sum_{n=0}^{N_m-1} s_m(n)s_m(n-l)}{\sqrt{\sum_{n=0}^{N_m-1} s_m(n)^2 \sum_{n=0}^{N_m-1} s_m(n-l)^2}} \qquad m \leq l \leq L;\ 0 \leq m \leq M-1. \tag{3.10}$$

The maximum lag L is proportional to the maximum fundamental period $T_{0,max}$ or, equivalently, the minimum fundamental frequency $f_{0,min}$, i.e.

$$L = T_{0,max} \cdot f_s = \frac{f_s}{f_{0,min}}. \tag{3.11}$$

In the MPEG-7 standard the HR is defined as

$$HR = \max_{m \leq l \leq N_{hop}} \Gamma_m(l), \tag{3.12}$$

where the upper border is limited by N_{hop}, the time interval between two frames. The maximum lag will be found at the fundamental frequency of the input signal. A drawback of the definition in Equation 3.12 is the fact that the zero-lag peak is often detected and will lead to a HR close to one. The value of the HR basically ranges between zero at white noise input and one for fully periodic signals.

Examples The scream in Figure 3.6(a) shows a HR close to one as expected from a vocalic sound. The explosion also leads to high HR values in Figure 3.6(d). The breakage of glass and the gunshot contain correlation and result in medium HR values shown in Figures 3.6(b) and 3.6(c).

Upper Limit of Harmonicity (ULH)

The ULH is the estimated frequency, above which the input signal no longer has any harmonic structure. Basically it is the relation of the output and input power of a time-domain comb filter tuned on the fundamental frequency of the signal [50]. The calculation requires the following steps:

1. The signal is filtered by the optimal-gain comb filter

$$\tilde{s}_m(n) = s_m(n) - G_m s_m(n - \hat{m}) \qquad 0 \leq n \leq N_w - 1. \tag{3.13}$$

$\hat{m}$ is the lag which maximizes the autocorrelation function, i.e.

$$T_0 = \arg\max_l \Gamma_m(l) \tag{3.14}$$

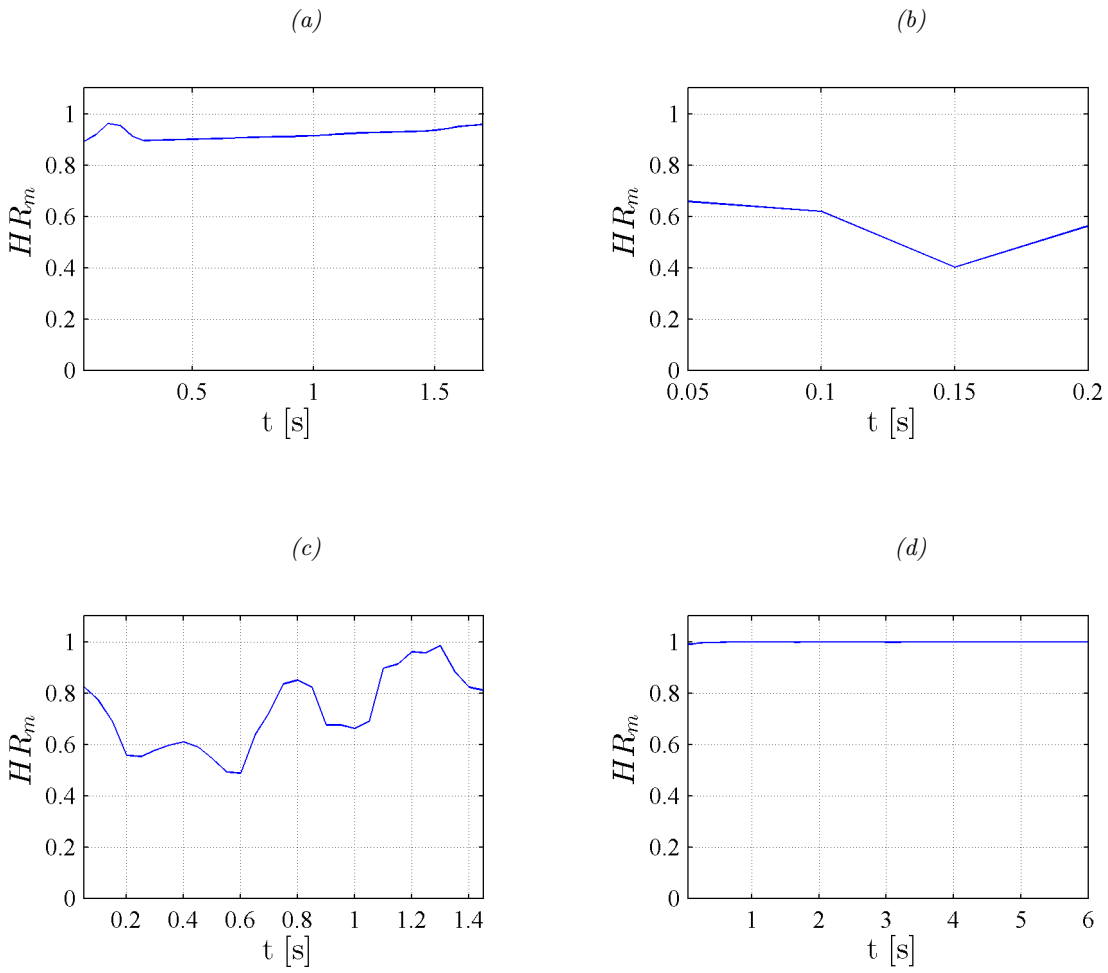

Figure 3.6: Harmonic Ratio feature examples: (a) scream, (b) gunshot, (c) glass break, (d) explosion.

and optimal comb-filter gain is calculated as

$$G_m = \frac{\sum_{n=0}^{N_m-1} s_m(n)s_m(n-\hat{m})}{\sum_{n=0}^{N_m-1} s_m(n)s_m(n-\hat{m})^2}. \tag{3.15}$$

2. The power spectrum $P'_m(k')$ of the input signal and the power spectrum of $P'_{m,c}(k')$ of the filter output are calculated according Equation 3.4.

3. The power ratio

$$R_m(k_{lim}) = \frac{\sum_{k'=k_{lim}}^{N_{FT}/2-K_{low}} P'_{m,c}(k')}{\sum_{k'=k_{lim}}^{N_{FT}/2-K_{low}} P'_m(k')} \tag{3.16}$$

is repeatedly calculated, while decrementing k_{lim} from k_{max} to the first frequency bin, until the R_m falls below 0.5. This bin index is then named k_{ulh} and the corresponding center frequency f_{ulh} of the bin indexed by k_{ulh} can be determined via Equation 3.5.

4. At last the ULH per frame can be calculated as

$$ULH_m = \mathrm{ld}\left(\frac{f_{ulh}}{1000}\right). \tag{3.17}$$

Similar as the HR, the ULH can be used to distinguish harmonic from non-harmonic sound, like voiced and unvoiced speech.

Examples The female scream in Figure 3.7(a) results in ULH values above 1, corresponding to a frequency of 2 kHz, which is realistic for a female voice. Similar to the HR feature, the ULH also shows similar values for the explosion in Figure 3.7(d), in comparison to the scream. The gunshot shows no harmonic structure at all according to Figure 3.7(b), while the breakage of glass in Figure 3.7(c) is partly harmonic when the glass splinters are jingling after the impact sound.

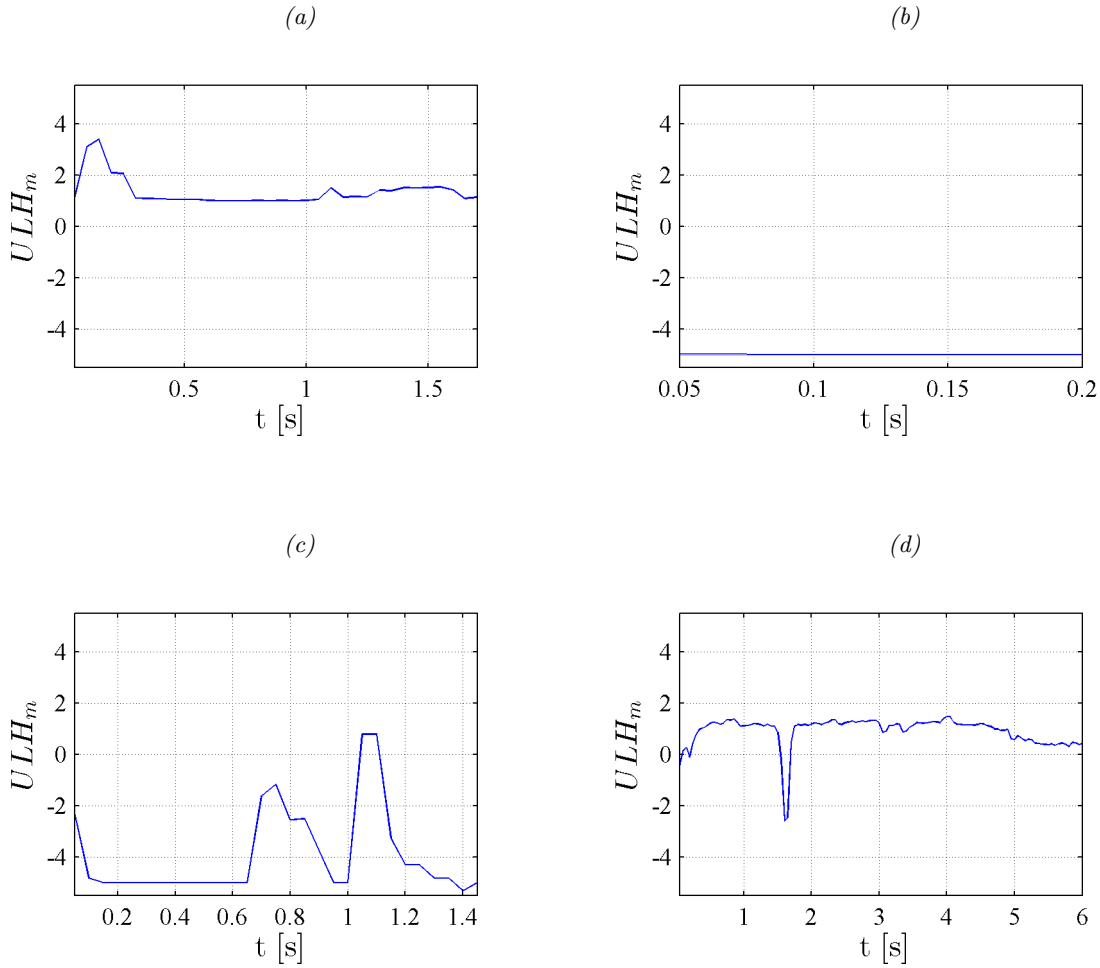

Figure 3.7: Upper limit of harmonicity examples: (a) scream, (b) gunshot, (c) glass break, (d) explosion.

Audio Fundamental Frequency (AFF)

In the MPEG-7 standard a spectro-temporal autocorrelation (STA) approach is defined for the estimation of the AFF [51]. The STA function is given as

$$\Gamma_{STA}(l) = \beta\Gamma_{TA}(l) + (1-\beta)\Gamma_{SA}(l), \tag{3.18}$$

where the temporal autocorrelation $\Gamma_{TA}(l)$ is given as

$$\Gamma_{TA}(l) = \frac{\sum_{n=0}^{N_w-1} s(n)s(n-l)}{\sqrt{\sum_{n=0}^{N_w-1} s(n)^2 \sum_{n=0}^{N_w-1} s(n-l)^2}}, \tag{3.19}$$

omitting the frame index m. Using solely the temporal autocorrelation can result in detecting the maximum lag at an integer multiple of a pitch candidate lag, e.g. $2l$ or $3l$. The spectral autocorrelation $\Gamma_{SA}(\cdot)$ introduced in [52] is not susceptible to this multiplications, i.e.

$$\Gamma_{SA}(l) = \Gamma_{SA}(k_l) = \frac{\sum_{k=0}^{K_{max}-1} S(k)S(k-k_l)}{\sqrt{\sum_{k=0}^{K_{max}-1} S(k)^2 \sum_{k=0}^{K_{max}-1} S(k-k_l)^2}}. \tag{3.20}$$

The time lag l is converted to the corresponding bin index using

$$k_l = \left[\frac{1}{l \cdot \Delta f}\right], \qquad 0 \le k \le K_{max} - 1, \tag{3.21}$$

$[\cdot]$ denotes the round-to-nearest operator. The fundamental period is estimated from the spectro-temporal autocorrelation function in Equation 3.18 as

$$T_0 = \arg\max_l \left[\Gamma_{STA}(l)\right], \tag{3.22}$$

leading to the fundamental frequency

$$AFF = f_0 = \frac{1}{T_0}. \tag{3.23}$$

The MPEG-7 standard requires additional parameters to complete the AFF descriptor, e.g. frequency limits and detection confidence measures. These are not used in this project.

Examples As expected, a reasonable fundamental frequency is detected for the scream in Figure 3.8(a). There is also some tonality in the breakage of glass and meaningful frequencies are detected in Figure 3.8(c). An explosion usually leads to a low fundamental frequency, the algorithm unfortunately detected much higher frequencies as expected in Figure 3.8(d). A gunshot has no fundamental frequency although the algorithm detected one for the duration of two frames in Figure 3.8(b).

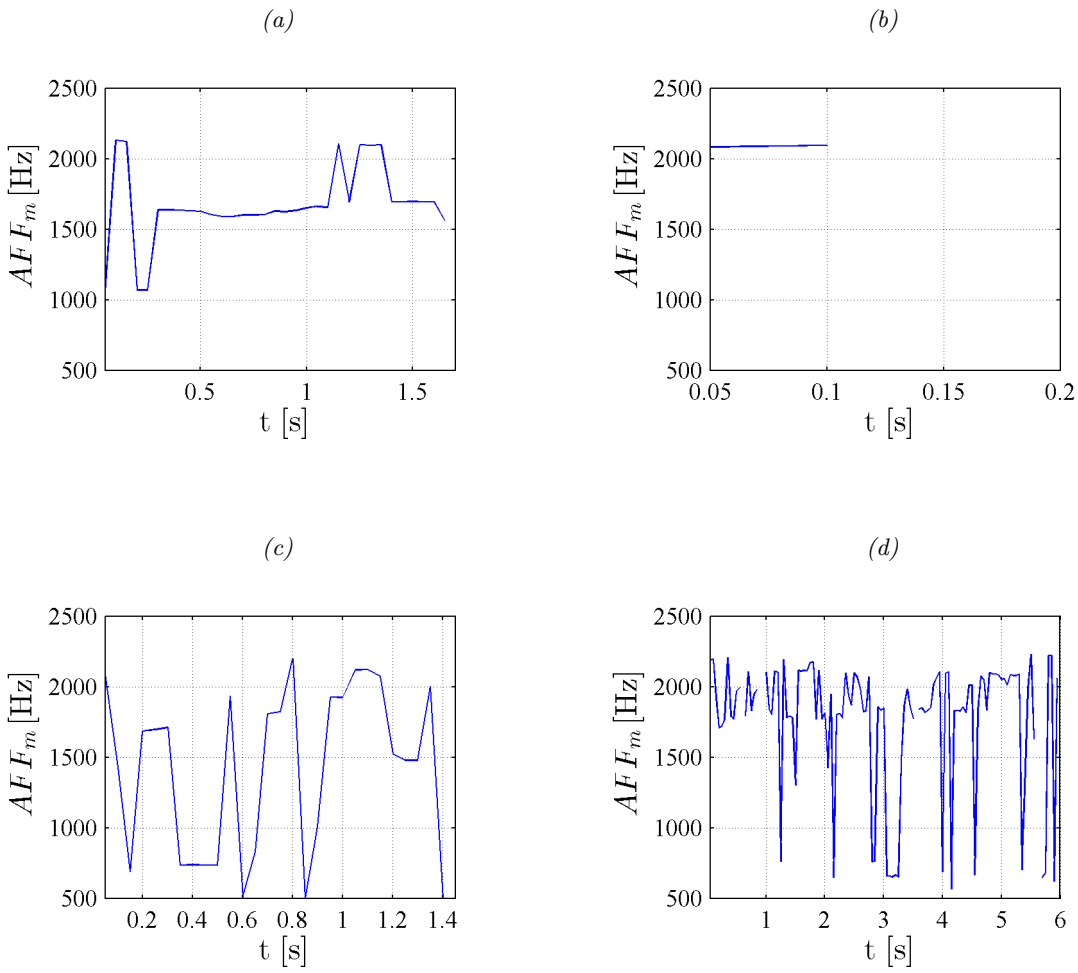

Figure 3.8: Audio fundamental frequency examples: (a) scream, (b) gunshot, (c) glass break, (d) explosion.

3.2 Teager Energy Operator

Speech emotion changes are the result of a change in the vocal tract parameters. To exploit those differences in a signal feature, the Teager Energy Operator (TEO) is used. Kaiser introduced the common form of this nonlinear operator in [53] as

$$\Psi_c\{s(t)\} = \left[\frac{\mathrm{d}}{\mathrm{d}t}s(t)\right]^2 - s(t)\left[\frac{\mathrm{d}^2}{\mathrm{d}t^2}\right]s(t) = \left[\dot{s}(t)\right]^2 - s(t)\ddot{s}(t). \tag{3.24}$$

This equation can be explained as the difference between the squared first derivative of the input signal and the input signal times the second derivative of the input signal.
To distinguish between normal speech and speech under stress, like screamed or highly agitated speech, the critical band (CB) based TEO autocorrelation envelope (TEO-CB-Auto-Env) feature by Zhou [54] is used. The processing blocks are shown in Figure 3.9.

The TEO-CB-Auto-Env feature is calculated as follows:

1. The input signal is filtered in 16 critical bands using a Gabor bandpass filter bank. The low and high frequency (f_{lo} and f_{hi}) as well as the center frequency f_c and the width of

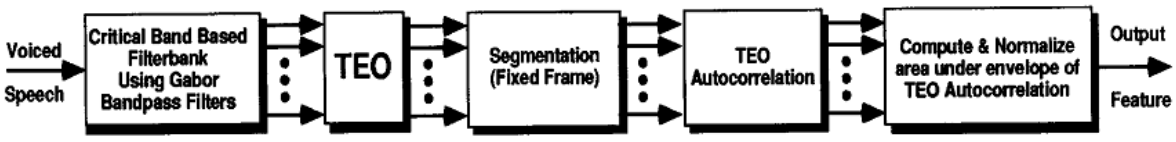

Figure 3.9: Overview of the TEO-CB-Auto-Env calculation [54].

each band are shown in Table 3.2.

band	f_{lo}	f_c	f_{hi}	bandwidth
#		Hz		
1	100	150	200	100
2	200	250	300	100
3	300	350	400	100
4	400	450	510	110
5	510	570	630	120
6	630	700	770	140
7	770	840	920	150
8	920	1000	1080	160
9	1080	1170	1270	190
10	1270	1370	1480	210
11	1480	1600	1720	240
12	1720	1850	2000	280
13	2000	2150	2320	320
14	2320	2500	2700	380
15	2700	2900	3150	450
16	3150	3400	3700	550

Table 3.2: Critical bands of the Gabor bandpass filter bank.

The Gabor filter's impulse and frequency responses are defined as [55]

$$h(t) = \mathrm{e}^{-a^2t^2} \cdot \cos(2\pi f_c t) \tag{3.25}$$

and

$$H(f) = \frac{\sqrt{\pi}}{2a}\left[\mathrm{e}^{-\frac{\pi^2(f-f_c)^2}{a^2}} + \mathrm{e}^{-\frac{\pi^2(f+f_c)^2}{a^2}}\right], \tag{3.26}$$

where a is the bandwidth and f_c the center frequency of the corresponding band. The discrete form of the Gabor filter can be derived by sampling its continous impulse response version, with $T = 1/f_s$,

$$h(n) = \mathrm{e}^{-a^2(nT)^2} \cos\left(\frac{2\pi f_c}{f_s} n\right). \tag{3.27}$$

The filter is tuned on the specific critical band by simply choosing a and f_c according

Table 3.2. All following operations are then performed individually for each band.

2. In the next step the Teager-Energy Operator is applied. Its discrete form introduced in [56] can be written as

$$\Psi\{s(n)\} = s(n)^2 - s(n-1)s(n+1). \tag{3.28}$$

3. If the input signal is not already arriving in frames of adequate size, segmentation in frames of 25ms with 50% overlap has to be applied.

4. The normalized autocorrelation of the energy-estimated signal is calculated.

5. The area under the autocorrelation envelope is calculated and normalized by $N/2$.

6. Finally, the TEO-CB-Auto-Env feature is available as a vector with 16 entries.

Examples Figure 3.10(a) shows the TEO-CB-Auto-Env feature for a female scream. Its value is continuously high in value as expected for a scream, especially at the higher frequency bands. All other classes show a very different TEO-CB-Auto-Env feature. It's much lower in value as to be seen in Figures 3.10(b), 3.10(c) and 3.10(d).

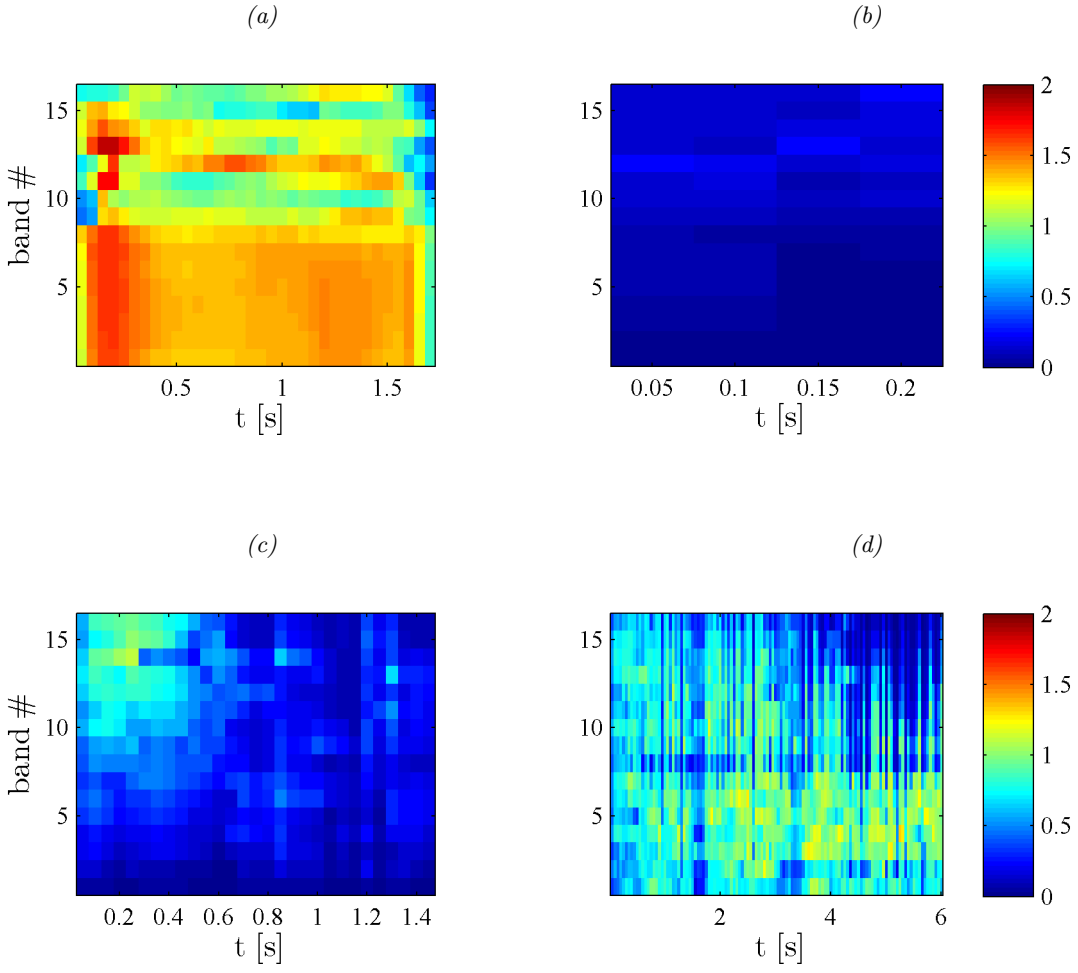

Figure 3.10: TEO-CB-Auto-Env feature examples: (a) scream, (b) gunshot, (c) glass break, (d) explosion.

3.3 Mel-frequency cepstral coefficients

Mel-frequency cepstral coefficients (MFCCs) are widely used in speech recognition. They represent the information contained in speech signals very well. By speaker-independently modeling the vocal tract parameters, especially the formants are reflected very well. The calculation and application of the MFCCs is widely documented [57], [58]. Figure 3.11 shows the steps involved.

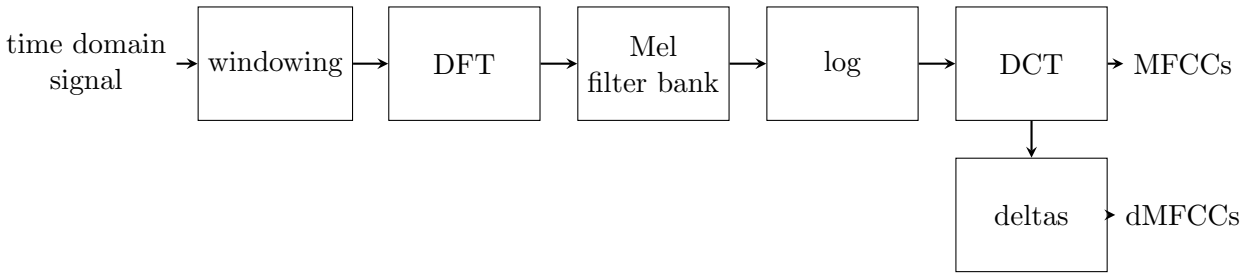

Figure 3.11: MFCC extraction process.

The MFCCs are calculated as follows:

1. Windowing and DFT transform of the time domain signal into the spectral domain.

2. Based on the m^{th} spectrum of the input signal, the mel-spectrum is calculated as

$$MF_m(r) = \frac{1}{A_r} \sum_{k=L_r}^{U_r} \left| V_r(k) S_m(k) \right|^2 \tag{3.29}$$

with the normalization factor

$$A_r = \sum_{k=L_r}^{U_r} \left| V_r(k) \right|^2 \tag{3.30}$$

and $V_r(k)$ being the triangular-shaped frequency response of the r^{th} mel filter for the DFT indices $[L_r; U_r]$.

3. The i^{th} MFCC coefficient is calculated by application of the Discrete Cosine Transform (DCT) on the logarithm of MF_m, i.e.

$$c_m(i) = \frac{1}{R} \sum_{r=1}^{R} \log(MF_m) \cos\left[\frac{2\pi}{R} \left(r + \frac{1}{2} \right) i \right] \tag{3.31}$$

with the typical values of $R = 24$ and the coefficient index i running from 1 to 13.

Examples The scream in Figure 3.12(a) shows the typical MFCC structure of a vocalic sound. Each of the other sounds can be differentiated via the 13 coefficients. The other examples are dominated by the lower coefficients. The explosion contains all frequency components and results in a high-valued first coefficient as seen in Figure 3.12(d).

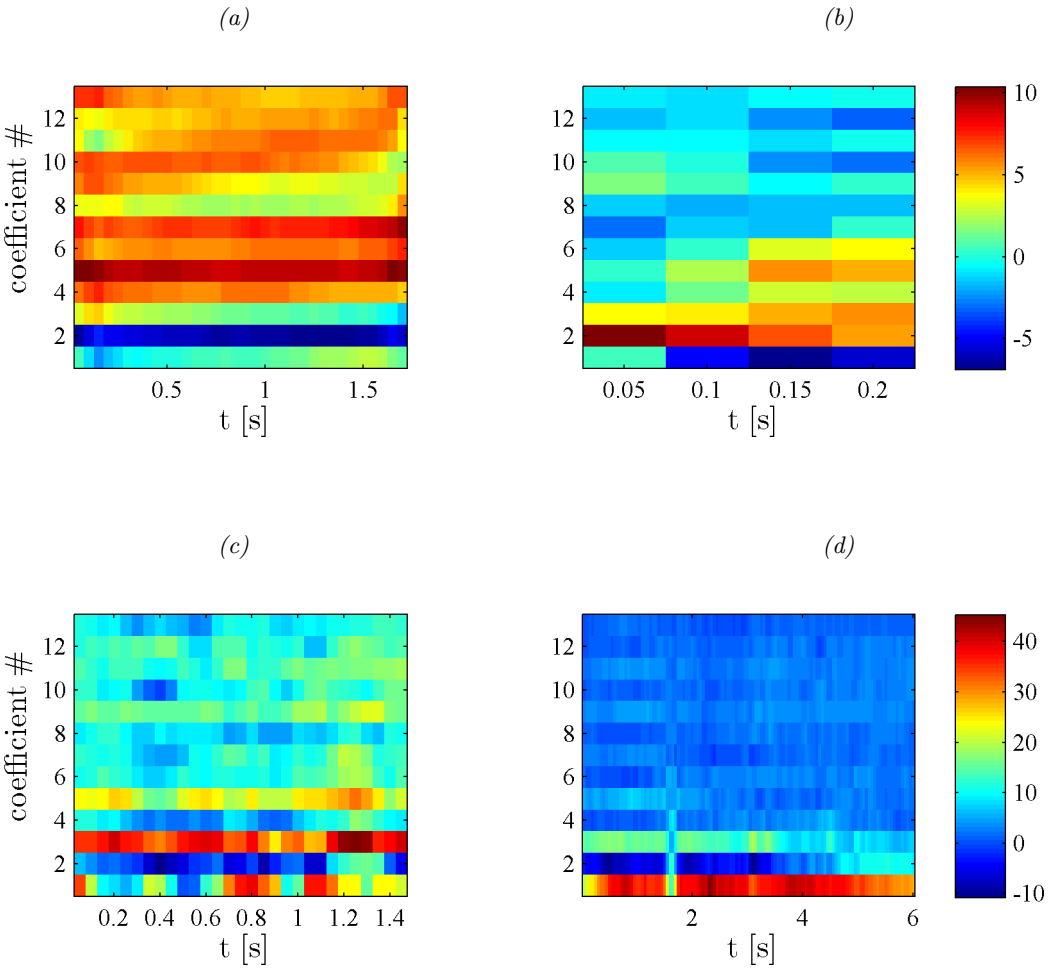

Figure 3.12: MFCC feature examples: (a) scream, (b) gunshot, (c) glass break, (d) explosion.

3.3.1 Delta- and Delta-Delta-MFCCs

To take into account the temporal changes of the signal, the delta and delta-delta MFCCs are commonly added to the feature vector. According to [48] they are defined as

$$\Delta c_m(i) = -c_{m-2}(i) - \frac{1}{2}c_{m-1}(i) + \frac{1}{2}c_{m+1}(i) + c_{m+2}(i) \tag{3.32}$$

and

$$\Delta\Delta c_m(i) = c_{m-2}(i) - \frac{1}{2}c_{m-1}(i) - c_m(i) - \frac{1}{2}c_{m+1}(i) + c_{m+2}(i). \tag{3.33}$$

Examples The $\Delta MFCCs$ of the scream show the typical attack-sustain-decay (ADSR) envelope of a vocalic utterance. One can observe from Figure 3.13(a) that during the sustain phase almost all of the coefficients remain more or less constant in value, while they clearly change at the beginning and the end of the event. The Δ-coefficients of the gunshot, Figure 3.13(b), and the breakage of glass, Figure 3.13(c), show that a smaller amount of them changes at the beginning and the end, so most of them evolve over time. The $\Delta MFCCs$ of the explosion in Figure 3.13(d) show that during the impact the lower numbered coefficients increase and immediately decrease in value.

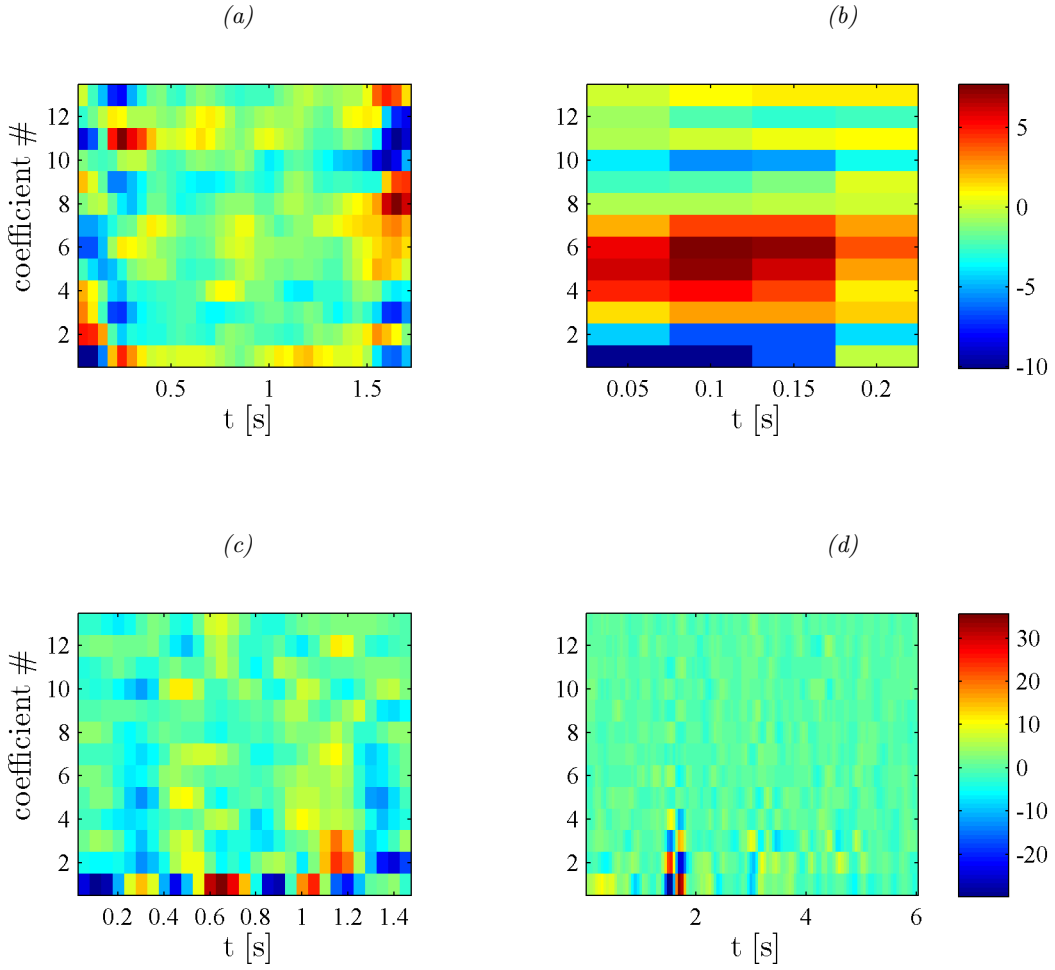

Figure 3.13: Δ MFCC feature examples: (a) scream, (b) gunshot, (c) glass break, (d) explosion.

3.3.2 Root Cepstral Coefficients

To reduce the influence of noise, the Root Cepstral Coefficients (RCCs) [59] can be used. They have been successfully applied in speech recognition. The calculation is very similar to the traditional MFCCs, instead of Equation 3.31 the following one is applied

$$c_m(i)^{RCC} = \frac{1}{R}\sum_{r=1}^{R}|MF_m|^{root} \cdot \cos\left[\frac{2\pi}{R}\left(r+\frac{1}{2}\right)i\right], \tag{3.34}$$

where $0.2 \leq root \leq 0.25$ is used to calculate the RCCs.

Examples Because of the fact that the sound event samples are noise-free, the $RCCs$ deliver the same results as the $MFCCs$, disregarding different scaling factors. Noise-like sounds like the gunshot in Fig 3.14(b) show a tendency to get more zero-coefficients, like the explosion in Figure 3.14(d).

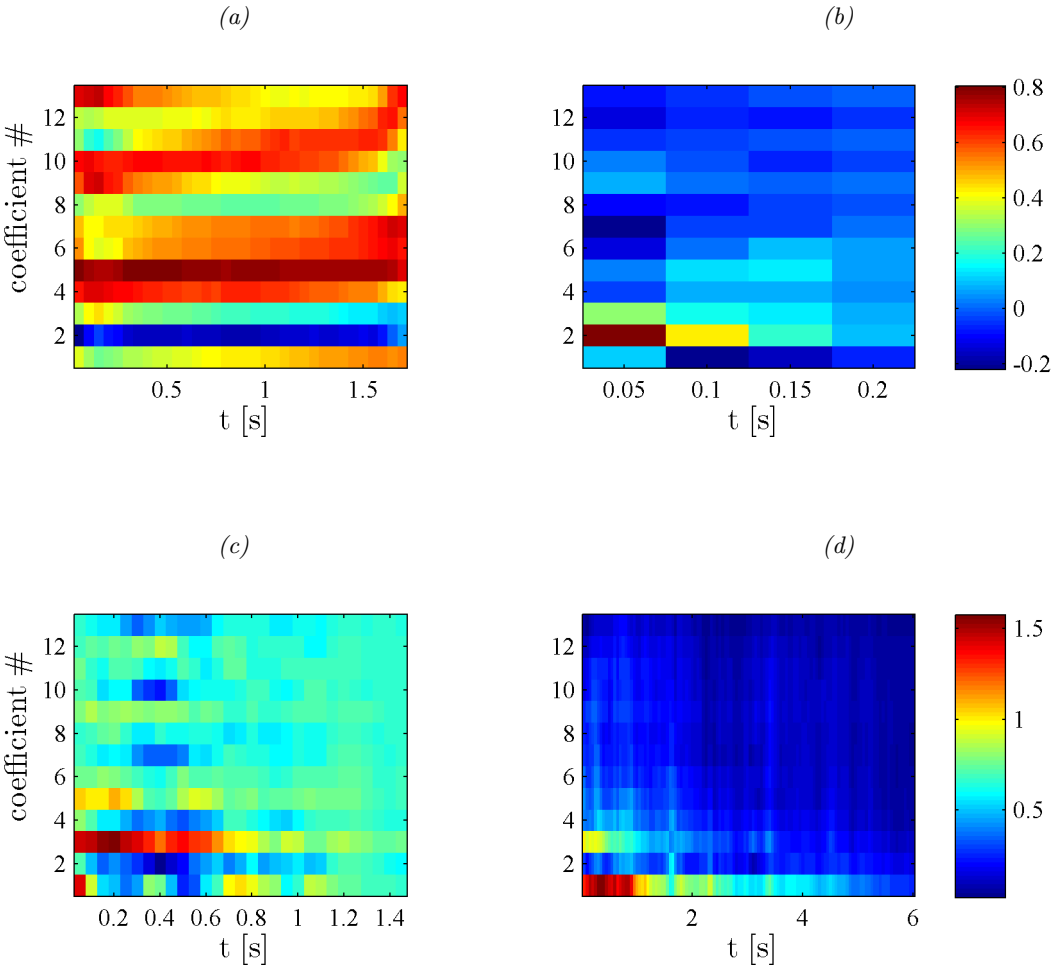

Figure 3.14: RCC feature examples: (a) scream, (b) gunshot, (c) glass break, (d) explosion.

3.3.3 Cepstral Mean Normalization

Another way of compensating noise in the MFCCs is the method of cepstral mean normalization (CMN) [60], [61]. The mean of all coefficients of an event will be subtracted from each coefficient, i.e.

$$\widetilde{c}_m(i) = c_m(i) - \frac{1}{M} \sum_{m'=1}^{M} c_{m'}(i). \tag{3.35}$$

Examples Due to the fact that the normalization has been applied within one audio file only, the resulting cepstral-mean normalized MFCCs in Figures 3.15(a) to 3.15(d) resemble the Δ-MFCCs very much. The application of the normalization is more useful over a longer period of time, covering more utterances.

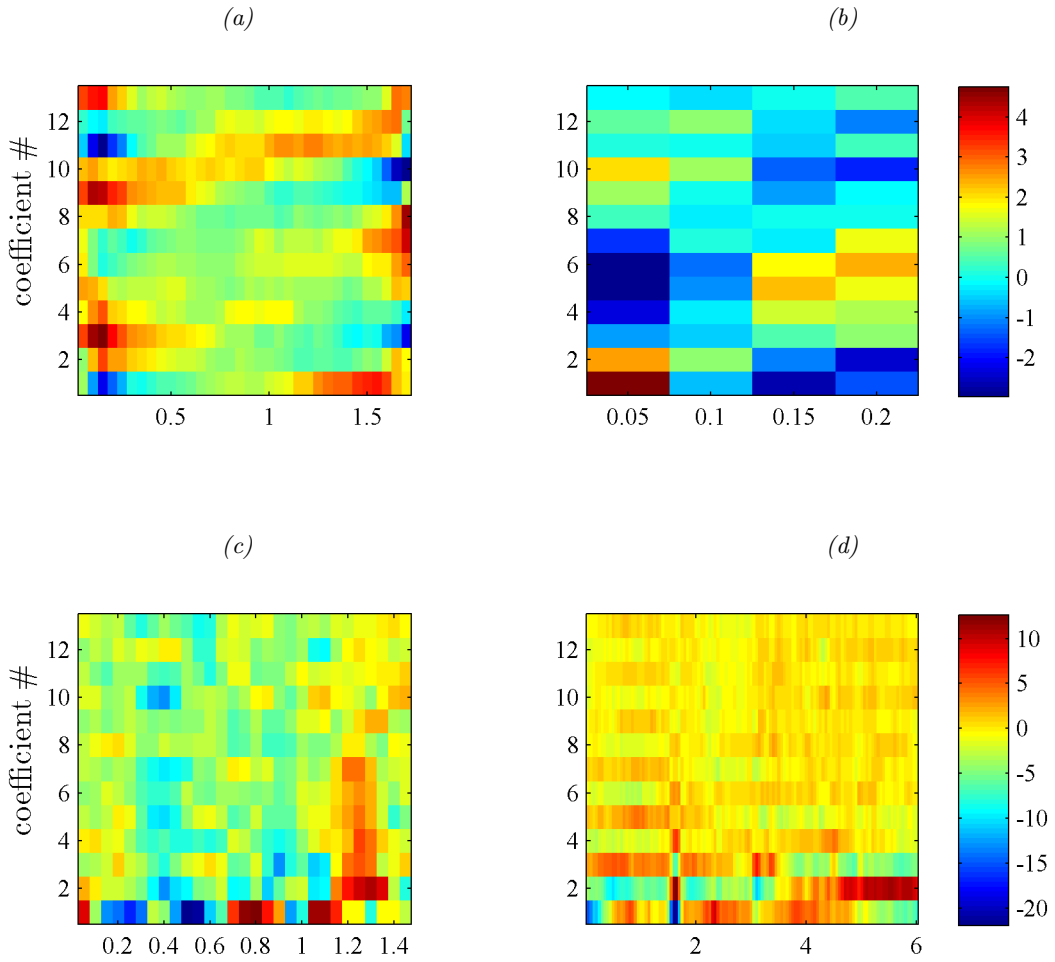

Figure 3.15: Cepstral-mean normalized MFCC feature examples: (a) scream, (b) gunshot, (c) glass break, (d) explosion.

4 Machine Learning

For the classification tasks several classifiers will be used. The training phase is considered as *supervised learning*, the system is in knowledge of the class labels of each sample. In this chapter, Gaussian Mixture Models (GMMs) and apart from the traditional generative parameter learning paradigm, two discriminative learning algorithms are presented. As an alternative to the GMMs, Support Vector Machines (SVMs) are introduced. The chapter concludes with model selection techniques and classifier evaluation methods.

4.1 Gaussian Mixture Models

A Gaussian Mixture Model (GMM) [27] is defined as a superposition of K Gaussian densities

$$p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k). \tag{4.1}$$

Each k$^{\text{th}}$ multivariate Gaussian density $\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ has its own mean vector and covariance matrix. This densities are called *components* of the GMM. π_k is called the *mixing coefficient* or the *responsibility* of the k^{th} component and

$$\sum_{k=1}^{K} \pi_k = 1, \quad 0 \leq \pi_k \leq 1. \tag{4.2}$$

To obtain the likelihood that an input vector $\mathbf{x}$ is generated by the GMM, the log-likelihood

function

$$\ln\big(p(\mathbf{x}|\boldsymbol{\pi},\boldsymbol{\mu},\boldsymbol{\Sigma})\big) = \ln\left(\sum_{k=1}^{K}\pi_k\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k)\right), \tag{4.3}$$

can be used. The log-likelihood of an input data set $\mathbf{X}$ of N samples $\{\mathbf{x}_1,...,\mathbf{x}_N\}$ is

$$\ln\big(p(\mathbf{X}|\boldsymbol{\pi},\boldsymbol{\mu},\boldsymbol{\Sigma})\big) = \sum_{n=1}^{N}\ln\left(\sum_{k=1}^{K}\pi_k\mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k)\right). \tag{4.4}$$

There is no closed form solution available to determine the the maximum likelihood parameters $\boldsymbol{\pi}$, $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$. To set up the parameters (i.e. training) $\boldsymbol{\pi}$, $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ of the GMM, the EM-Algorithm is usually used. The covariance matrix $\boldsymbol{\Sigma}_k$ is often assumed to be a diagonal matrix. This reduces the computational costs and the GMM is still able to model the covariances, if enough components are available [62].

The k-means algorithm is often used to initialize the EM algorithm for learning GMMs. For this reason it is introduced below.

4.1.1 k-means Algorithm

In general, the k-means Algorithm [27] is used to cluster a data set given. It can be used to initialize the centers ($\boldsymbol{\mu}_k$) of the components of a Gaussian Mixture Model. An objective function[1] is defined as

$$J = \sum_{n=1}^{N}\sum_{k=1}^{K} r_{nk}\|\mathbf{x}_n - \boldsymbol{\mu}_k\|^2, \tag{4.5}$$

where $\|\mathbf{x}_n - \boldsymbol{\mu}_k\|^2$ is the euclidean distance between $\mathbf{x}_n$ and $\boldsymbol{\mu}_k$.
The indicator function

$$r_{nk} = \begin{cases} 1 & \text{if } k = \arg\min_j \big\|\mathbf{x}_n - \boldsymbol{\mu}_j\big\|^2 \\ 0 & \text{otherwise} \end{cases}, \tag{4.6}$$

assigns a certain data point $\mathbf{x}_n$ to the cluster k. By setting the derivative of the objective function with respect to $\boldsymbol{\mu}_k$ to zero, the solution for $\boldsymbol{\mu}_k$ is

$$\boldsymbol{\mu}_k = \frac{\sum_{n=1}^{N} r_{nk}\mathbf{x}_n}{\sum_{n=1}^{N} r_{nk}}. \tag{4.7}$$

The k-means algorithm stops to iterate if the cumulative distance J between the data points and the assigned centers has converged or the pre-defined maximum number of iterations has exceeded.

[1] it may also be called cost function

4.1.2 Expectation-Maximization Algorithm

The expectation-maximization (EM) [27] algorithm is an iterative procedure for the purpose of finding a maximum likelihood of a statistical model, where no closed-form solution is available to obtain its parameters. The algorithm consists of the initialization, the E- and M-step:

1. Initialize the means $\boldsymbol{\mu}$ (possibly done by the k-means algorithm), covariances $\boldsymbol{\Sigma}$ and mixing coefficients $\boldsymbol{\pi}$.

2. Evaluate the initial log-likelihood using equation 4.4.

3. *E-step:* Compute $N \times K$ responsibilities $\gamma(z_{nk})$, i.e.

$$\gamma(z_{nk}) = \frac{\pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}. \tag{4.8}$$

4. *M-step:* Re-estimation of the parameters using the current responsibilities $\gamma(z_{nk})$ leads to the new values

$$\boldsymbol{\mu}_k^{\text{new}} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) \mathbf{x_n}, \tag{4.9}$$

$$\boldsymbol{\Sigma}_k^{\text{new}} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk})(\mathbf{x_n} - \boldsymbol{\mu}_k^{\text{new}})(\mathbf{x_n} - \boldsymbol{\mu}_k^{\text{new}})^T, \tag{4.10}$$

and

$$\pi_k^{\text{new}} = \frac{N_k}{N}, \tag{4.11}$$

where

$$N_k = \sum_{n=1}^{N} \gamma(z_{nk}). \tag{4.12}$$

5. Evaluate the log-likelihood according Equation 4.4 and check for convergence until the convergence criterion is met. If not, start again from the *E-step* assigning the new parameter estimates, i.e. $\pi_k = \pi_k^{\text{new}}$, $\boldsymbol{\mu}_k = \boldsymbol{\mu}_k^{\text{new}}$ and $\boldsymbol{\Sigma}_k = \boldsymbol{\Sigma}_k^{\text{new}}$ for $k = 1, ..., K$.

The algorithm finds only local optima. Therefore, many runs can be necessary until a good parameter estimate is found.

4.1.3 Bayesian Classifier

Based on Bayes' rule the Bayesian classifier is able to determine the class posterior probability

$$p(c|\mathbf{x}) = \frac{p(\mathbf{x}|c)p(c)}{\sum_{c'=1}^{N_c} p(\mathbf{x}|c')p(c')}, \tag{4.13}$$

of a data sample $\mathbf{x}$ [27], [63], where c is the random variable of the class. It is a measure of the probability that $\mathbf{x}$ originates from $c \in \{1, ..., N_c\}$ where N_c is the number of classes in the classification task. The class prior $p(c)$ is an estimate of the probability that any instance from randomly sampled data originates from c. The denominator is usually omitted because it only scales the result. The most likely class label $\hat{c}$ can be predicted using the maximum-aposteriori (MAP) estimate using

$$\hat{c} = \arg\max_{1 \leq c \leq C} p(c|\mathbf{x}) = \arg\max_{1 \leq c \leq C} p(\mathbf{x}|c)p(c). \tag{4.14}$$

In this case, the term $p(\mathbf{x}|c)$ is modeled by the GMM for a particular class c, i.e. $p(\mathbf{x}|c) = p(\mathbf{x}|\pi_c, \boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c)$.

4.1.4 Discriminative Learning Algorithms for GMMs

After the generative training methods for GMMs two algorithms for discriminative parameter learning are presented [63]. It is proposed to either optimize the conditional likelihood or maximize the margin. Both methods are based on the Extended Baum-Welch (EBW) algorithm, which executes an iterative parameter update similar to the EM algorithm.

Conditional-Likelihood GMMs

The goal of this learning algorithm is to to optimize the conditional likelihood, because a good conditional likelihood leads to good classification performance. As the objective function of the GMM the conditional log likelihood [63] is used

$$CLL(\mathbf{X}|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \log \left(\prod_{n=1}^{N} p(c_n|\mathbf{x}_n) \right). \tag{4.15}$$

The Extended Baum-Welch (EBW) algorithm is used and with the help of partial derivatives of the CLL and a discrete approximation of the Gaussian distribution (because the EBW has been formulated for discrete probability distributions), re-estimation formulas for the means, diagonal covariances and component weights can be stated.

Maximum-Margin GMMs

Furthermore, GMMs optimizing the multi-class margin of a sample n have been introduced. The multi-class margin [64] is given as

$$d_n(\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \min_{c \neq c_n} \frac{p(c_n, \mathbf{x}_n|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma})}{p(c, \mathbf{x}_n|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma})} = \frac{p(c_n, \mathbf{x}_n|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma})}{\max_{c \neq c_n} p(c, \mathbf{x}_n|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma})}. \tag{4.16}$$

The sample n is correctly classified if $d_n(\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) > 1$. With approximations and relaxing the constraints to a soft margin problem, similar as in Section 4.1.4 the EBW is used to iteratively re-estimate the parameters of the GMMs.

4.2 Support Vector Machines

Considering a binary classification problem, an observation/pattern instance x is defined to be either in the class with the label $y = 1$ or not in the class labeled with $y = -1$. This assignment can be formalized as [65]

$$(x_1, y_1), ..., (x_N, y_N) \in \mathbf{X} \times \{\pm 1\}, \tag{4.17}$$

where $\mathbf{X}$ is a non-empty set containing all observations. This assignment is done to simplify the mathematical handling of the problem. To be able to separate the positive and negative observation, a hyperplane in a dot product space $\mathcal{H}$ is defined by a normal vector $\mathbf{w}$ and a scalar bias b as

$$\langle \mathbf{w}, \mathbf{x} \rangle + b = 0, \mathbf{w} \in \mathcal{H}, \mathbf{x} \in \mathbb{R}, \tag{4.18}$$

with the decision function

$$f(\mathbf{x}) = \operatorname{sgn}(\langle \mathbf{w}, \mathbf{x} \rangle + b). \tag{4.19}$$

If the learning problem is linearly separable, a unique optimal hyperplane exists. The margin between any data point and the hyperplane is maximized by solving the objective function

$$\underset{\mathbf{w} \in \mathcal{H}, \mathbf{b} \in \mathbb{R}}{\arg\min}\ \tau(\mathbf{w}) = \frac{1}{2} \|\mathbf{w}\|^2, \tag{4.20}$$

under the inequality constraints

$$y_n(\langle \mathbf{w}, \mathbf{x_n} \rangle + b) \geq 1, \quad n = 1, ..., N. \tag{4.21}$$

Commonly, the data is not linearly separable and the hyperplane can not be constructed without violating the constraints in Equation 4.21. Using slack variables $\xi_n \geq 0, \quad n = 1, ..., N$ the constraints relax to

$$y_n(\langle \mathbf{w}, \mathbf{x_n} \rangle + b) \geq 1 - \zeta_n, \quad n = 1, ..., N. \tag{4.22}$$

By executing

$$\underset{\mathbf{w} \in \mathcal{H}, \mathbf{b} \in \mathbb{R}}{\arg\min}\ \tau(\mathbf{w}) = \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{n=1}^{N} \xi_n, \tag{4.23}$$

a soft margin classifier is obtained, where the constant $C > 0$ can be used to setup the trade-off between minimizing the training error and maximizing the classification margin. This is called a

dual optimization problem. With the use of Lagrange multipliers and the Karush-Kuhn-Tucker (KKT) conditions the *dual problem* can be stated as

$$\arg\max_{\alpha\in\mathbb{R}^\eta} \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} \alpha_n \alpha_m y_n y_m \langle \mathbf{x_n}, \mathbf{x_m} \rangle, \tag{4.24}$$

resulting in the hyperplane decision function (or classification rule)

$$f(\mathbf{x}) = \operatorname{sgn}\left(\sum_{n=1}^{N} y_n \alpha_n \langle \mathbf{x}, \mathbf{x_n} \rangle + b \right). \tag{4.25}$$

As seen in Figure 4.1 the classification problem cannot be solved linearly in the input space, although transforming it into a higher dimensional feature space using the transformation function $\Phi(\mathbf{x})$ enables linear solving of the classification problem.

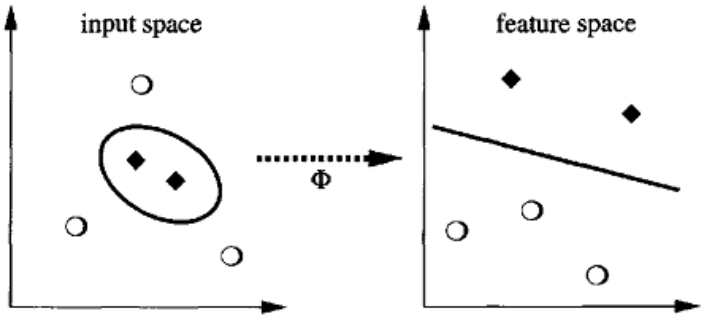

Figure 4.1: Principle of SVM [65].

Because the data points are only included via their dot product, the dot product $\langle \mathbf{x_n}, \mathbf{x_{n'}} \rangle$ in the input space can simply be replaced by the dot product in the higher-dimensional feature space $\langle \Phi(\mathbf{x_n}), \Phi(\mathbf{x_{n'}}) \rangle$. Instead of performing computationally exhaustive transformations the so-called *kernel-trick* can be applied. A kernel function is defined as

$$\mathcal{K}(\mathbf{x_n}, \mathbf{x_{n'}}) = \langle \Phi(\mathbf{x_n}), \Phi(\mathbf{x_{n'}}) \rangle. \tag{4.26}$$

The optimal kernel function is data-dependent and can only be found empirically [13]. The most commonly used kernel functions are:

- Polynomial kernel function with the polynomial order p:

$$\mathcal{K}_p(\mathbf{x_n}, \mathbf{x_{n'}}) = (\langle \mathbf{x_n}, \mathbf{x_{n'}} \rangle + 1)^p. \tag{4.27}$$

- Gaussian kernel i.e. radial basis function (RBF) with the standard deviation σ of the Gaussian:

$$\mathcal{K}_\sigma(\mathbf{x_n}, \mathbf{x_{n'}}) = \mathrm{e}^{\frac{\left\| \mathbf{x_n} - \mathbf{x_{n'}} \right\|^2}{2\sigma^2}}. \tag{4.28}$$

- Sigmoid kernel function with k being the amplification and θ the offset:

$$\mathcal{K}_{(k,\theta)}(\mathbf{x_n}, \mathbf{x_{n'}}) = \tanh\left(k \langle \mathbf{x_n}, \mathbf{x_{n'}} \rangle + \theta\right). \tag{4.29}$$

Further information on pattern recognition with SVMs can be found in e.g. Duda [66].

4.3 Model Selection with k-fold Cross Validation

Especially when using the maximum-likelihood approach, relying solely on the classification performance of the training data set can be dangerous because of overfitting [27]. If a lot of data is available during system development, an extra set of data apart from training and test sets should be used. With the help of this so-called validation set, the performance of the model can be evaluated with data independent from the training phase to optimize the model parameters. Because this optimization process is usually iterative, overfitting can still be an issue. So, the final evaluation of the model should happen with the test data set to get the most accurate results.

In many cases, including acoustic event detection, the available data is rare. With the help of the *cross-validation* technique the amount of data to be used can be maximized, while having a reliable estimate on the classification performance. The dataset is divided into k subsets, where $k-1$ sets are used for training and a single one for validation. This process is repeated k times, resulting in having each of the k subsets used exactly once for validation. This process is shown in Figure 4.2 for $k = 4$. In each iteration (or so-called *fold*) $k-1$ subsets are used for training and one (shown in red) for validation.

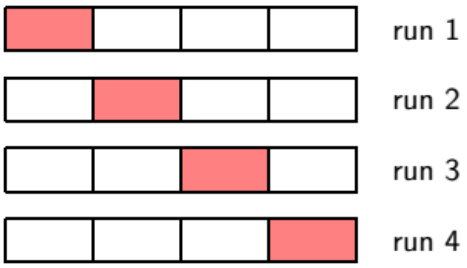

Figure 4.2: k-fold cross validation technique [27].

The major drawback of k-fold cross validation is the increased number of training runs, in the example shown above five runs are now required instead of one. This could dramatically slow down the model training and selection process.

4.4 Classifier Evaluation

After classification has been done, there are four different types of results possible for binary classification problems. These cases can be seen in the so-called confusion matrix in Figure 4.3. On the horizontal axis the result of the classifier is entered (positive or negative). The vertical axis is marked with the actual result, based on the ground truth or the gold standard reference.

This delivers four possible results:

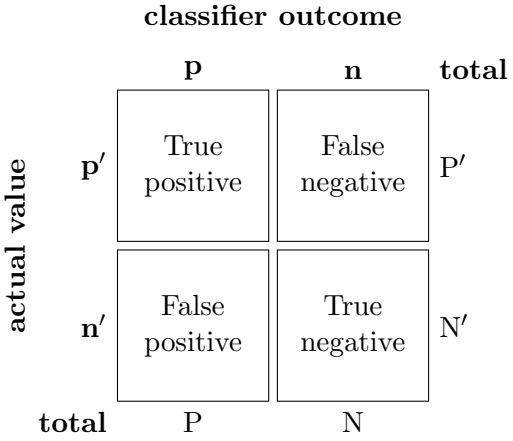

Figure 4.3: Confusion matrix, based on [67].

- **True Positive (TP)**
  The classifier correctly identifies the event in accordance with the reference.
- **True Negative (TN)**
  The classifier correctly rejects the sample, the reference does not indicate an event either.
- **False Positive (FP)**
  The classifier identifies the sample as a relevant event but the reference does not. This is called a false alarm.
- **False Negative (FN)**
  The classifier rejects the sample, although the reference indicates an event. This is called a miss.

4.4.1 Classification Performance Measures

Much better conclusions can be drawn from the following performance metrics used to evaluate the classifiers [68]:

Precision

The *precision* reaches its highest value of 1 if no false alarms occur, it decreases in case of many false positive detections, i.e.

$$Precision = \frac{TP}{TP + FP}. \tag{4.30}$$

Recall

Recall is an indication for missed detections, it gets low when many test candidates have been left out and reaches its maximum when all events are detected, i.e.

$$Recall = \frac{TP}{TP + FN}. \tag{4.31}$$

Accuracy

The accuracy combines both the false alarm rate and the missed detection rate and is therefore a metric for the overall performance of the classifier, i.e.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}. \tag{4.32}$$

F-score

An alternative measure of the accuracy is the *F-score*. Its general form is given as

$$F_\beta = \frac{(1 + \beta^2) \cdot Precision \cdot Recall}{\beta^2 \cdot Precision + Recall}, \forall \beta \in \mathbb{R} : \beta > 0. \tag{4.33}$$

With Equation 4.30 and 4.31 the F-score can be stated for the different classifier outcomes

$$F_\beta = \frac{(1 + \beta^2) \cdot TP}{(1 + \beta^2) \cdot TP + \beta^2 \cdot FN + FP}. \tag{4.34}$$

β is used to set the importance of recall in relation to precision, leading to the fact that the F-score weights the accuracy so that recall is β times more important than precision. Commonly the F_1 score is used, with $\beta = 1$, which results in the harmonic mean of precision and recall

$$F_1 = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall}. \tag{4.35}$$

Acoustic Event Error Rate

In coarse of the *CLEAR* challenge [45] the Acoustic Event Error Rate (AEER) was defined as an evaluation criteria specifically for an acoustic event detection system. Three types of errors are introduced:

1. **Deletion**
   An event was not detected at all (missed event).
2. **Insertion**
   An event was detected although it didn't really occur (extra event).
3. **Substitution**
   An event was detected but not the correct one.

The AEER is defined as

$$AEER = \frac{D + I + S}{N_r}, \tag{4.36}$$

where N_r is the total number of events in the reference and D, I and S the number of deletions, insertions and substitutions. This error rate is similar to the word error rate in ASR and can get larger than 1.

5 Implementation and Results

In this chapter the detection scenario for the acoustic event detection system is explained. The sounds to be detected are introduced and described in time and frequency domain. A suitable sound library collection containing occurrences of all events is presented. The proposed system is introduced and its components are explained. The MATLAB simulation model of the system is presented and the implementation is explained in detail. The experiments conducted are documented, evaluated and interpreted.

5.1 Event Classes and Audio Data Set

In this acoustic event detection system, five classes of audio events are defined which shall be classified and distinguished from the background sound. In this section the characteristics of the sounds are explained.

- **Scream**

  A screamed utterance typically lasts between one and two seconds. After a short attack period the amplitude remains constantly high until the ending of the scream. In the spectrum the typical formant structure of human voice, in this case a female voice, can be seen.

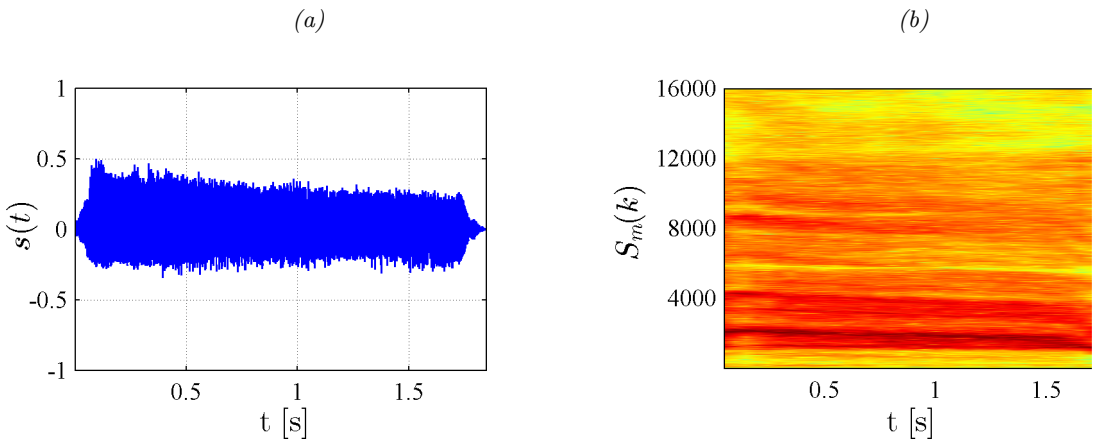

Figure 5.1: Female scream: (a) time domain, (b) frequency domain.

- **Gunshot**

  A gunshot is basically a very short event, lasting only some decades of milliseconds. Due to the high energy of a shot the event is often prolonged by reverberation. The spectrum of a gun shot is tending to be white but will be colored by the influence of the gun in use.

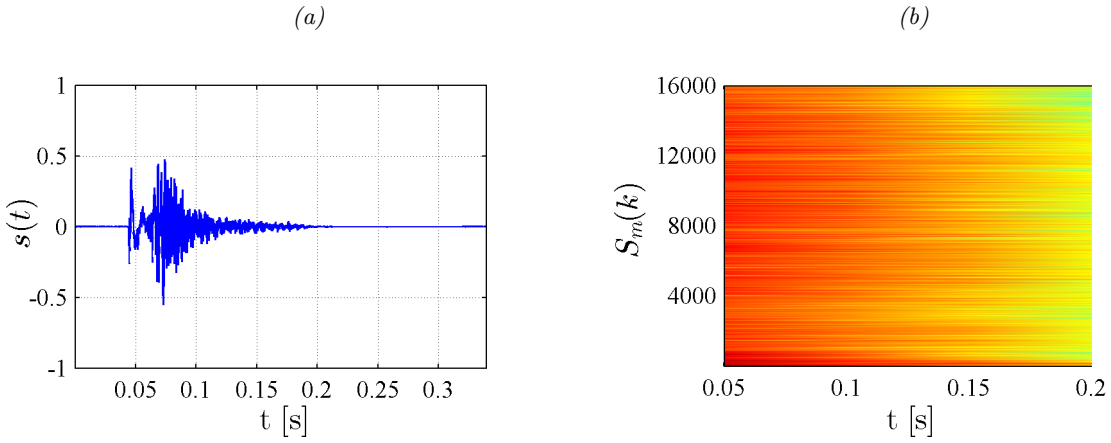

Figure 5.2: Gunshot: (a) time domain, (b) frequency domain.

- **Breakage of glass**

  The breakage of glass usually takes about half a second, depending on the size of the window smashed. During the first 500ms all frequencies in the spectrum show noteable amplitude due to the impact, afterwards only the sound of the broken glass is heard.

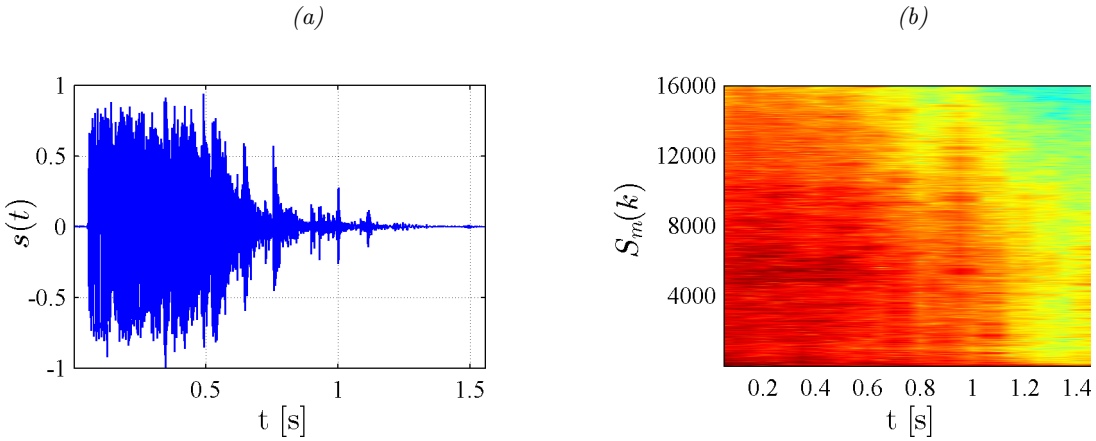

Figure 5.3: Glass breaking: (a) time domain, (b) frequency domain.

- **Explosion**

  An explosion is a similar event to a gunshot. In contrary, it lasts longer and takes at least some seconds. In the spectral domain it is dominated by frequencies around 100 Hz.

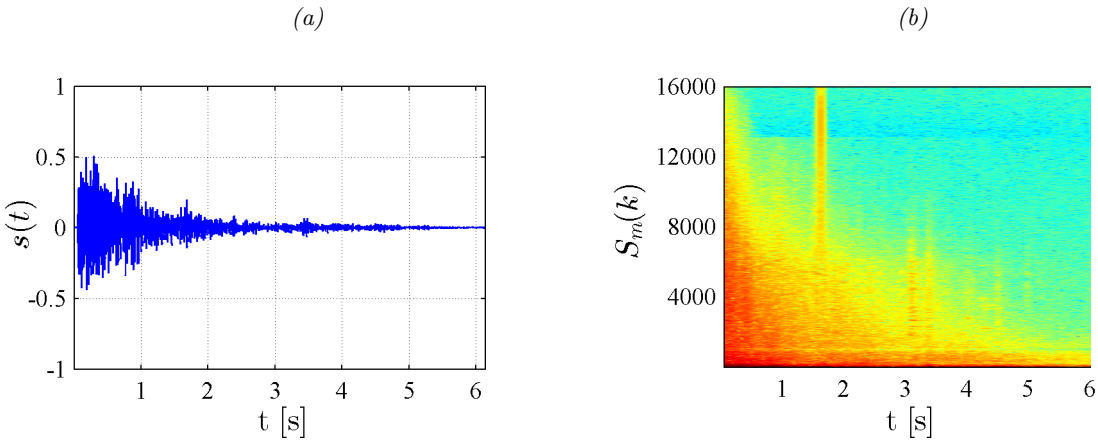

Figure 5.4: Explosion: (a) time domain, (b) frequency domain.

- **Vocal (human voice)**

  Vocal sounds contain all utterances emitted by humans. This can be with (e.g. a word) or without (e.g. a scream) phonetic context. The example in Figure 5.5 shows a woman speaking the words 'oh yes it was'. The four syllables are clearly identifiable in the time domain signal, as well as the typical frequency shaping of the vocal tract can be seen in the frequency domain.

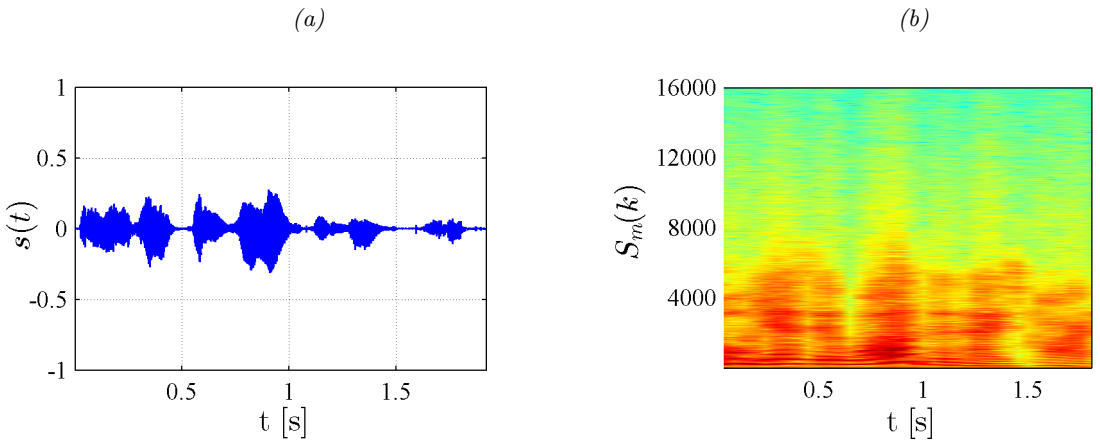

Figure 5.5: Vocal: (a) time domain, (b) frequency domain.

- **Background**

  Background sounds are usually very diverse and differ with the environment. Generally, background sound or noise is broadband, low and constant in amplitude. In Figure 5.6 a typical airport terminal background sound is shown. Footsteps and the sound of people talking in the distance can be heard.

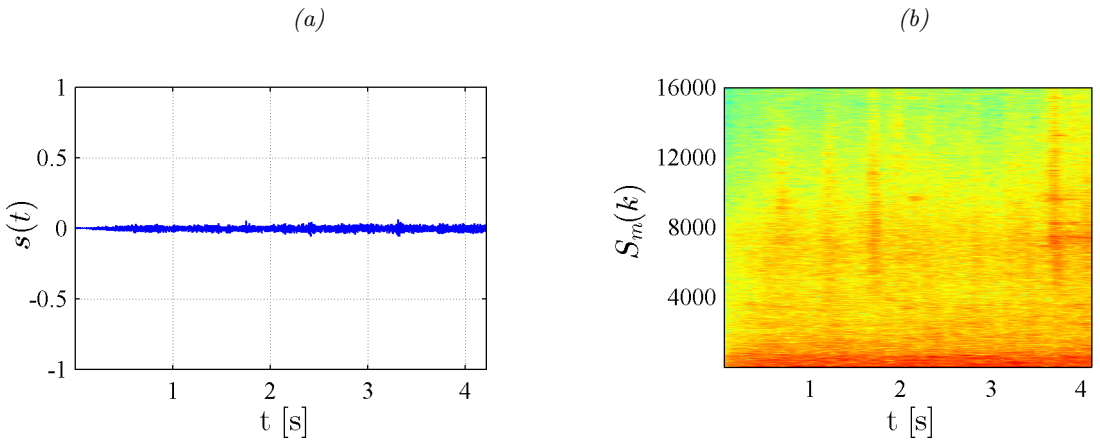

Figure 5.6: Background: (a) time domain, (b) frequency domain.

The sound libraries available for training and testing the event detection system are listed in Table 5.1. The files were individually selected from different libraries to improve the generalization of the resulting model, assigned to the corresponding classes and annotated if necessary to ensure high quality data.

Table 5.2 gives an overview of the number of recordings in use for each class and the actual frequency of occurrence of each event. Usually, audio files originating from such libraries contain only one event, but all audio files were checked manually for proper labeling and re-annotated if necessary.

Name	Type	CDs	Recordings
Sound Ideas			
Series 6000 General	general	40	3294
BBC Sound Effects Library	general	60	2224
Lucas Film Sound Library	movie	6	420
Dimension Sound Effects Library	general	10	985
Series 6000 Extension VII	general	10	930
Sony Pictures Sound Effects Library	movie	10	2308
The Network Sound Effects Library	general	120	9679
Universal Studio Sound Effects Library	movie	5	484
Warner Brothers Sound Effects	movie	5	490
Others			
Audio Pro Sound Effects Library by Yannick Chevalier	general	5	136
Sound Effects Bible	general	8	1669
Valentino Production Sound Effects Library	movie	50	3803
TIMIT [69]	speech	n/a	6300

Table 5.1: Sound effect libraries in use.

	scream	**gunshot**	**glass break**	**explosion**	**vocal**	**background**
recordings	123	83	128	105	42	16
duration (mm:ss)	24:19	9:10	27:01	28:59	34:11	62:24

Table 5.2: Library file statistics.

It was possible to get many libraries containing general, everyday sounds. Due to discretion reasons, expressions of horror, like screams, are often omitted in common effect libraries. This problem was addressed in [70]. The same problem occurs with events that rarely happen in real life recordings but are security-relevant, like the gunshot, explosion or glass break events. Data sets provided for acoustic event detection challenges like DCASE 2013 [42] or DCASE 2016 [71] are recorded in everyday situations. They cover e.g. offices, different public places like train stations, streets, etc. without those security-relevant events. It was found that sound effect libraries used in the movie industry do contain the required material and because of that the pool of audio data also includes such collections.

5.2 System Topology

For the following experiments, two acoustic event detection system topologies have been evaluated. They differ in their classifiers. The first system in Figure 5.7 is based on GMMs. Each event class and the background sound is modeled by one GMM. The most likely class is then predicted with the help of the maximum-aposteriori estimation presented in Section 4.1.3. The second system developed in Figure 5.8 replaces the GMMs by SVMs. They are deployed for *one-to-many* operation, i.e. the SVM of each class to be detected uses the data of the actual event class as the positive data and the data of all other events as well as the background sounds as negative data. Hence, every SVM classifier independently delivers a binary result that states if the specific event has been detected or not.

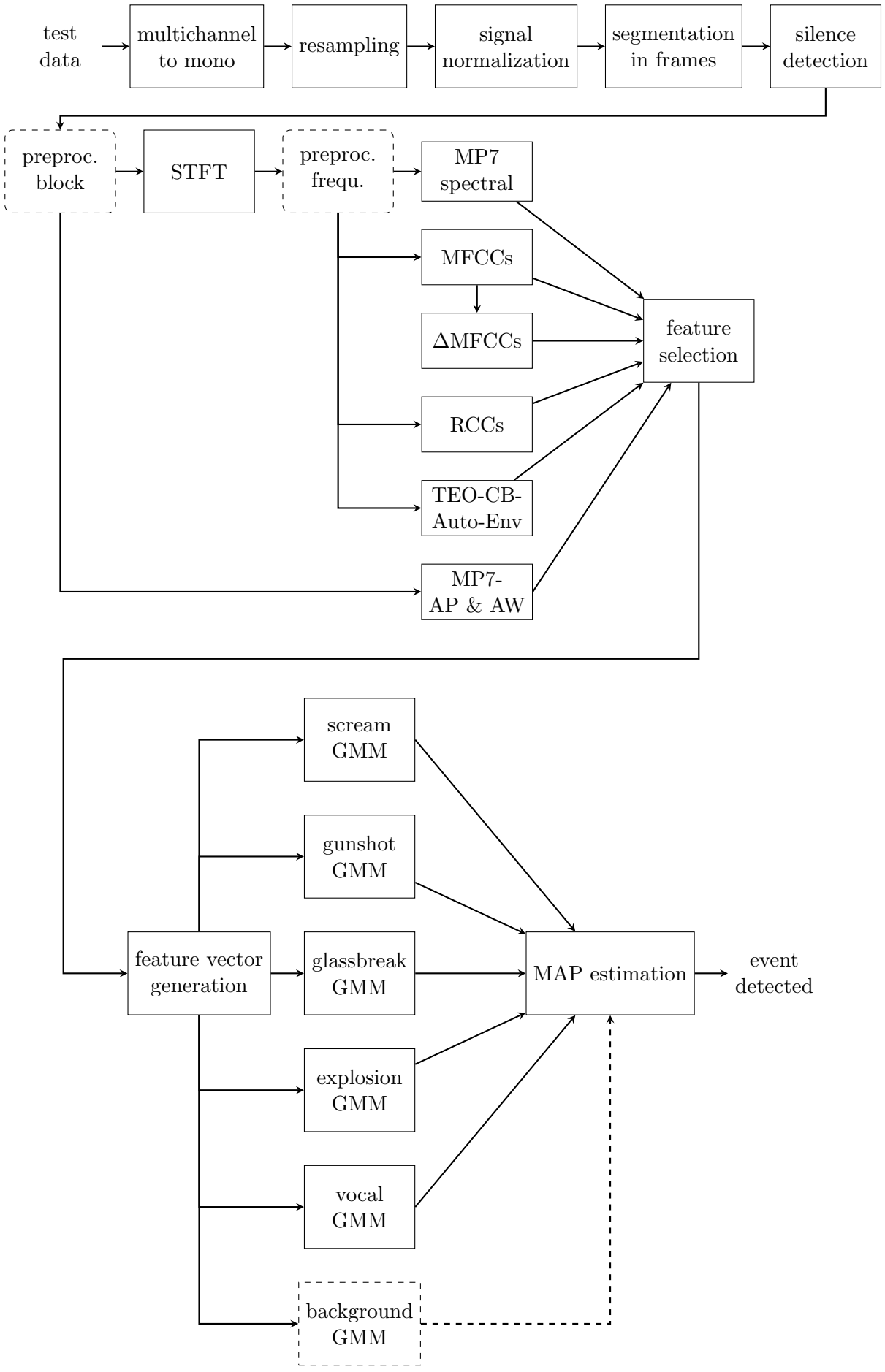

Figure 5.7: GMM-based AED system fed with test data.

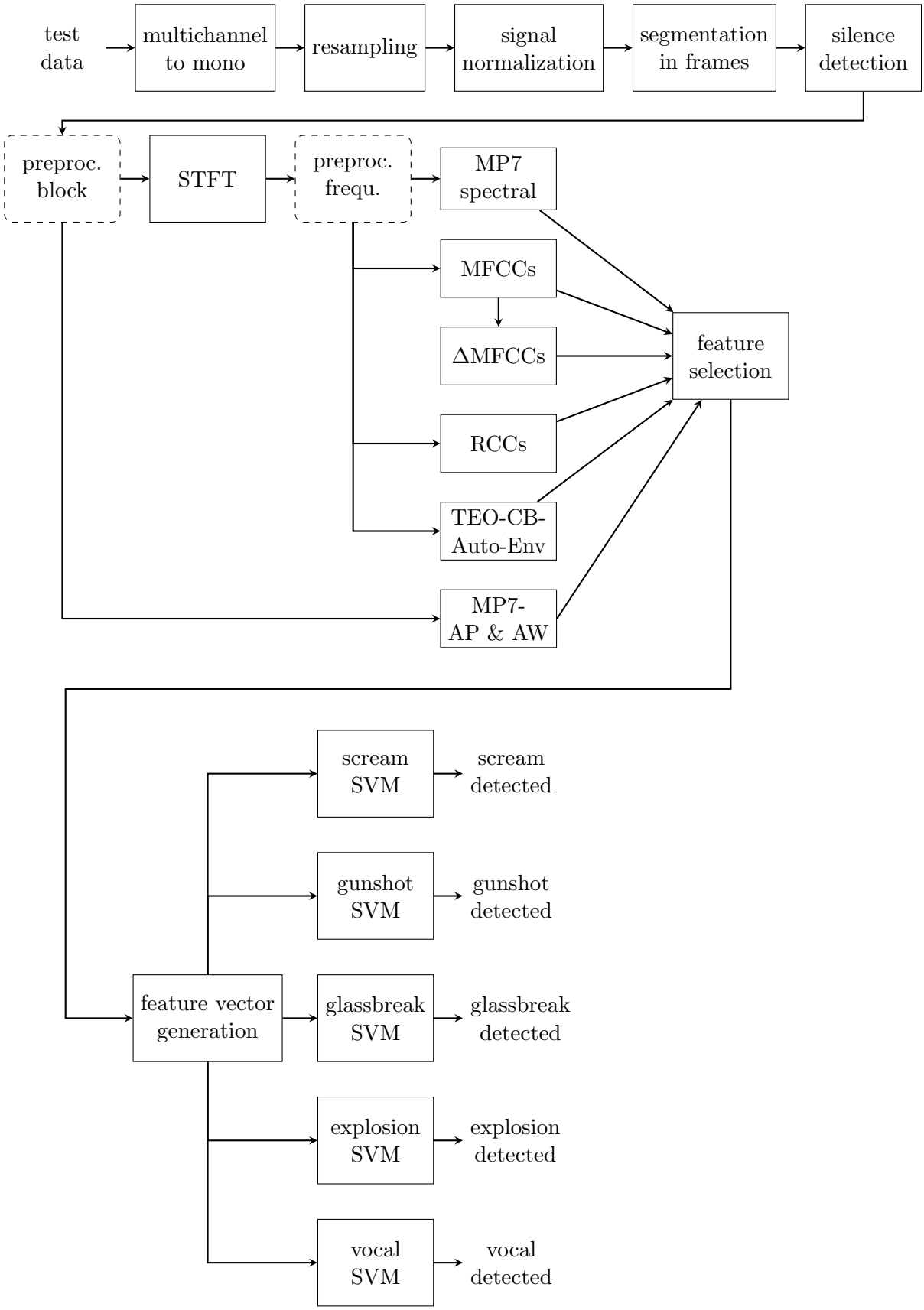

Figure 5.8: SVM-based AED system fed with test data.

5.3 Preprocessing and Feature Extraction

Figure 5.9 shows a block diagram of the processing chain in use. If an input audio signal consists of more than one channel, it is converted to a mono signal by either dropping the excess channels or by performing a mono-downmix. If the sample rates of the input file and the event detection system mismatch, a resampler adjusts the sampling rate of the input accordingly. The input signal is then normalized and segmented into frames according to a given *frame length* and *overlap* parameter. After dropping silent frames below a threshold, the preprocessed time domain signal is available for block-wise feature extraction.

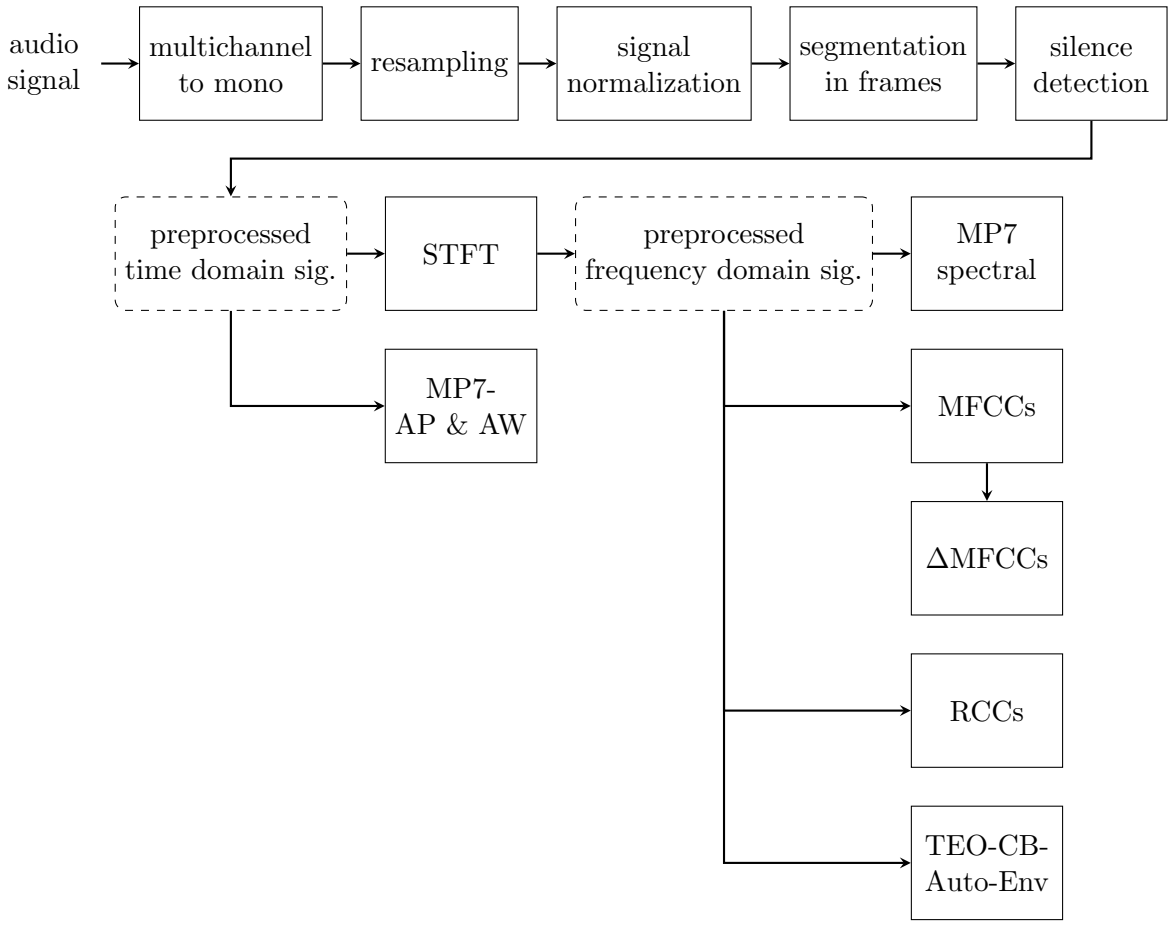

Figure 5.9: Feature extraction with preprocessing.

The MP7 basic descriptors *Audio Waveform* and *Audio Power* are extracted directly from the preprocessed time domain signal. All other features in use are based on the power spectrum, i.e. the short-term Fourier transform is applied resulting in the preprocessed frequency domain signal. Then the MP7 *Basic Spectral Descriptors* Audio Spectrum Centroid, Spread and Flatness and the *Audio Harmonicity Descriptors* Harmonic Ratio, Upper Limit of Harmonicity and Audio Fundamental Frequency, the MFCCs with the ΔMFCCs, the RCCs and the Teager Energy Operator critical band based auto-envelope are calculated.

The feature vector is individually specified for each class to be able to select the proper feature set for each event to be detected. Ntalampiras [14] recommends feature sets for the classification of the same events as defined in this work. These are presented in Table 5.3.

class	features
scream	MFCCs, ΔMFCCs, TEO-CB-Auto-Env, HR
gunshot	MFCCs, ΔMFCCs
glassbreak	MFCCs, ΔMFCCs, ASC
explosion	MFCCs, ΔMFCCs

Table 5.3: Features of the event classes.

5.4 Model Training

The event detection system is designed to be used with two different models: GMMs and SVMs. For GMM training, a generative and a discriminative approach can be evaluated. The SVMs are used for *one-to-many* classification with a kernel function. The feature set is manually defined depending on the experiment, the selection is based on the recommendations in Table 5.3.

5.4.1 Generative GMM Training

For generative GMM training, the feature vectors of the training data are split into subsets, each subset representing one class to be detected or the background sound. As the first training step the k-means algorithm is executed to initialize the centers of the diagonal GMMs, followed by the Expectation-Maximization (EM) algorithm to estimate all parameters of the model. The training process in Figure 5.10 has to be repeated for each class as well as the background.

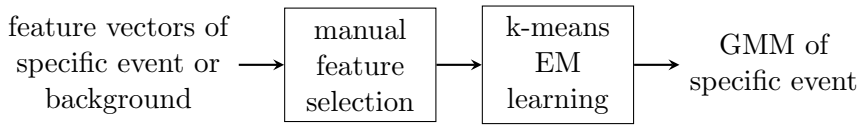

Figure 5.10: GMM training with generative algorithm.

For N_c different event classes to be detected, this procedure results in $N_c + 1$ trained GMMs which can be used for classification in the event detection system.

5.4.2 Discriminative GMM Training

When using discriminative algorithms, the feature vectors of all classes including the background are supplied to the algorithm. The CLL or max-margin optimization algorithm is used for discriminative GMM training. Class labels are also supplied to assign a feature vector to the class it belongs.

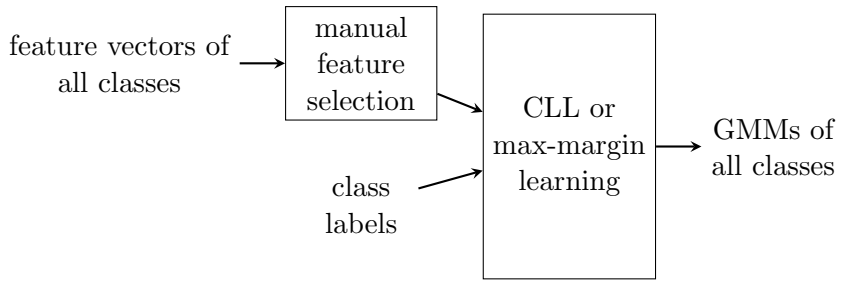

Figure 5.11: GMM training with discriminative algorithms.

The discriminative training shown in Figure 5.11 results in $N_c + 1$ GMMs of all classes after the algorithm completed its run.

5.4.3 SVM Training

For training an SVM for *one-to-many* classification, the training algorithm needs to be supplied with the feature vector of the specific event to be detected with this particular SVM and the feature vectors of all other classes in the detection scenario as illustrated in Figure 5.12. Each SVM corresponding to an event class has to be trained individually, so the algortihm runs N_c times.

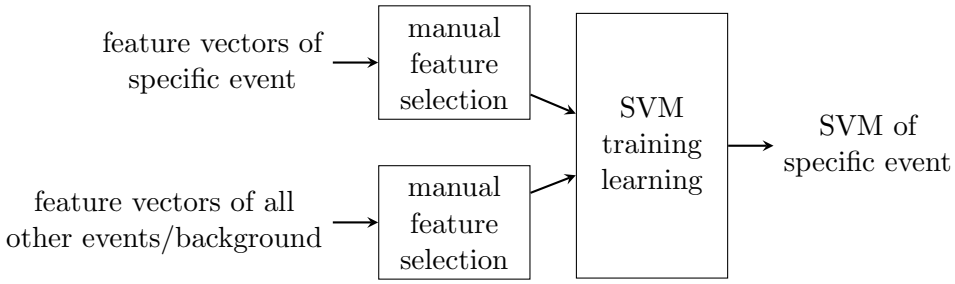

Figure 5.12: SVM training.

This results in N_c SVMs.

5.5 Classification and System Evaluation

The classification with the use of GMMs is performed with the maximum-aposteriori estimate as shown in Figure 5.7. The SVMs independently deliver results as illustrated in Figure 5.8. For evaluation, the number of *true positives*, *true negatives*, *false positives* and *false negatives*, explained in Section 4.4, are counted per frame for each class individually. Based on these numbers the measures *precision*, *recall* and *accuracy* are calculated (cf. Section 4.4.1). For better comparability with a single measure only, the F_1-score was computed, which is also widely used in acoustic event detection related publications and challenges. The performance of the system as a whole is obtained by macro-averaging [72] the class-wise measures which puts equal weight on each class. Additionally, the AEER of the system is computed.

5.6 MATLAB Implementation

A MATLAB framework has been implemented to perform all tasks of the Acoustic Event Detection scenario. It is modularly built to independently perform feature extraction, model training and evaluation of the classification. An initialization script *aedinit* was written to set all paths for the environment. For each of the three steps a `run` script was created:

1. Preprocessing and feature extraction: `runfeatureextraction.m`
2. Model learning and classification: `runtraining.m`
3. Event detection system evaluation: `runevaluation.m`

Each script depends on the output of its predecessor. All output data is organized in cells and structs which can easily be stored and loaded.

Each of the blocks represents an independently runnable module to efficiently conduct the experiments. Every module save its output in a `MAT` file. The feature extraction module processes the audio data input and stores the extracted signal features by class in a structured way. The actual learning algorithm runs within the model training module. Finally, the system's classification metrics are calculated in the evaluation module. In the next sections these modules are explained in detail.

5.6.1 Feature Extraction Module

The whole signal preprocessing and feature extraction processes are encapsulated in this module. As inputs the audio file database and a parameter set are given. The processing chain is implemented according to the block diagram in Figure 5.9. Variable parameters and options are defined via the `FEO` (**F**eature **E**xtractor **O**ptions) struct. These parameters, crucial to be consistent over various feature extraction runs, are summarized in Table 5.4.

parameter name	**value**
sample rate	32 kHz
frame length	varies between 30 and 200 ms
overlap [%]	75%
silence threshold [%]	10%
# MFCC coefficients	13
# Mel bands for MFCCs	36
MFCC cepstral mean subtraction	on
normalization	on

Table 5.4: Feature extractor options.

The script shown in Listing A.1 in the Appendix uses the files per class from the manually pregenerated file lists. A feature structure is calculated for each file and stored in a vector of such structs. For this process the `getFeatureStruct` function is used. It performs the preprocessing steps necessary as well as the actual feature extraction:

1. Load the audio data as floating point from the WAV file with `audioread`.
2. Convert multi-channel, in many cases stereo files, to mono by summing all channels and divide the data by the number of channels.
3. Normalize the audio data to plus/minus one.

4. Segment the audio data in frames according to the *frame length* and *overlap* parameters.
5. Remove "silent" frames, i.e. those with an energy below a specified percentage of the average energy of the whole file via *silent threshold* parameter.
6. Calculate the signal features based on the preprocessed block-wise time domain signal (MP7-AP and -AW).
7. Perform the short-time Fourier transform after applying the Hanning window.
8. Calculate the MP7 spectral features, the MFCCs and ΔMFCCs, RCCs and the TEO-CB-Auto-Env features.

These steps are visible in this order after global error handling in Listing A.2 in the Appendix. All features are pre-computed using frame lengths $[30; 60; 100; 150; 200]$ ms.

The whole set of extracted features can be stored by simply saving the `FS` struct using MATLAB's `save` command. Each class is corresponds to a cell with each cell containing the vector of feature structs described above. To access e.g. the feature set of the third file of the scream class use `FS.scream{3}`. Note that in the learning phase the feature set can be selected individually for each class.

5.6.2 Model Training Module

The training module is used to abstract the underlying class model and training algorithms. Via the class model template `CMT` the options for the training run can be set. The settings differ between the usage of GMMs and SVMs. Table 5.5 shows the setup for GMM training. The data ratio is a common parameter for all methods. A data ratio of 0.8 means that the data available for each class is linearly split in a way that the first 80% of the corresponding events is used for training and the remaining 20% is designated as test data during the evaluation phase.

parameter name	value
data ratio	0.8
type	dgmm_ml, nl or my
# components	64, 128, or 256
kmeans max. iterations	100
EM max. iterations	1000
features	feature set in use

Table 5.5: Generative GMM training options.

GMMs with k-means/EM algorithm or CLL and MM optimization can be used. There are three implementations available, the `gmdistribution` class out of MATLAB's Statistics and Machine Learning Toolbox (ml), the Netlab Toolbox Implementation (nl) [73] and a self-developed version (my). All generative GMM experiments have been conducted with `gmdistribution`.

Alternatively, the GMMs can be trained discriminativly with the MATLAB framework published by Pernkopf et al. [63]. The parameters for the conditional-likelihood and the max-margin optimization algorithms are given in Table 5.6.

The usage of SVMs backed by `svmtrain` class out of MATLAB's Statistics and Machine Learning Toolbox is also possible. *libsvm* [74] could be used as an alternative.

All training methods, except for the CL and Max-Margin algorithm, are handled by the `runtraining` script in Listing A.3 and the `cmTrain` function in Listing A.4 in the Appendix. After successful training the class models are stored in `MAT` files for further processing in the evaluation module.

parameter name	value
data ratio	0.8
# components	64, 128, or 256
features	feature set in use
EM max. iterations	1000
CLL max. iterations	500
max-margin max. iterations	500
components	128
updatePrior	1
CL optimization parameter	
F_1	7
max-margin optimization parameters	
F_2	15
η	5
σ	0.45

Table 5.6: Discriminative GMM training options.

parameter name	value
data ratio	0.8
type	svm
max. iterations	100000
kernel function	radial (RBF)
σ	1.0
features	feature set in use

Table 5.7: SVM training options.

5.6.3 Evaluation Module

In the evaluation module the testing of the model with the use of the remaining data designated for testing takes place. The classification results are then used to calculate the performance metrics. They can be used to derive optimizations or iteratively find better models with model selection techniques like cross-validation. The final results can be exported for further use and be stored in `MAT` files.

5.7 Experiments

With the acoustic event detection system described in the previous sections various experiments have been conducted. The first set of experiments deals with a GMM-based classifier. Parameters of the system and the classifier were varied. Tests have been done with different number of components of the GMM and frame lengths as well as three different feature sets.

5.7.1 Classification Results with Generative GMMs

In this section a generatively trained GMM has been used for all experiments. The models were validated using *4-fold cross validation*.

Varying the Number of Components

In this experiment, the number of components of the GMMs in use has been varied. If the number of components is increased, better fitting of the model to the training data is expected, but over-fitting is more likely to occur. Table 5.8 shows the setup of this experiment.

parameter	**setting**
# GMM components	64, 128 or 256
frame length	200ms
feature set	MFCCs, ΔMFCCs, MP7, TEO-CB-Auto-Env

Table 5.8: Setup for the components variation experiment.

The accuracy tends to increase when increasing the components of the GMM. This can be observed in Table 5.9. The actual precision and recall results vary between the classes.

	scream			gunshot			glassbreak			explosion			vocal		
#	preci.	recall	accur.	preci.	recall	accur.	preci.	recall	accur.	preci.	recall	accur.	preci.	recall	accur.
64	**88.05**	58.54	95.13	21.16	**31.67**	92.12	**32.76**	**82.75**	94.24	32.09	**89.36**	77.67	73.10	32.41	81.65
128	85.89	63.11	95.35	20.25	26.67	92.45	32.18	75.98	**94.34**	33.10	88.82	78.69	**73.24**	33.21	81.78
256	85.31	**64.95**	**95.45**	**23.44**	23.81	**93.49**	31.35	71.72	94.30	**33.36**	87.71	**79.05**	72.64	**34.42**	**81.87**

Table 5.9: Class-wise classification metrics of the components variation experiment.

In terms of F_1-score scream, Table 5.10 shows that the scream, explosion and vocal class perform best with 256 components, whereas gunshot and glassbreak deliver the best score with the use of 64 components. Considering the system's average, the best score is reached with 256 components.

	F_1 score					
#	scream	gunshot	glassbreak	explosion	vocal	∅
64	70.33	**25.37**	**46.93**	47.22	44.91	46.95
128	72.76	23.02	45.21	48.23	45.70	46.98
256	**73.75**	23.62	43.63	**48.34**	**46.71**	**47.21**

Table 5.10: F_1 scores of the components variation experiment.

For the whole system, Table 5.11 shows that precision and recall are lowest for 64 components, but the accuracy and AEER are the best with the maximum number of components in this experiment.

	overall scores			
#	precision	recall	accuracy	AEER
64	**49.43**	**58.95**	88.16	90.30
128	48.93	57.56	88.52	87.51
256	49.22	56.52	**88.83**	**85.53**

Table 5.11: Overall classification metrics of the components variation experiment.

Varying the Frame Length

In the following experiment the frame length of the segmented input audio signal has been varied in steps between 60 and 200ms. Frequency resolution is increased while temporal resolution is

reduced when using higher frame lengths. This could be of advantage or disadvantage for certain classifiers. Table 5.12 shows the parameters used in this setup.

parameter	**setting**
# GMM components	128
frame length	60, 100, 150, 200ms
feature set	MFCCs, ΔMFCCs, MP7, TEO-CB-Auto-Env

Table 5.12: Setup for the frame length variation experiment.

The results in Table 5.13 vary between the classes. The voice-based classes scream and vocal deliver the best results with the shortest frame length of 60ms. Gunshot and glass break can be detected best with shorter frame lengths but their precision increases with longer lengths. For the explosion the best recall is found with the longest frame length, but precision and accuracy are best with the shortest frame length.

	scream			gunshot			glassbreak			explosion			vocal		
#	preci.	recall	accur.	preci.	recall	accur.	preci.	recall	accur.	preci.	recall	accur.	preci.	recall	accur.
60	**89.85**	67.62	**96.18**	10.94	**29.56**	88.25	30.93	**82.81**	94.07	**47.95**	80.48	**88.09**	**79.89**	**34.20**	**82.88**
100	88.15	**67.83**	96.03	13.54	27.72	90.20	31.24	79.53	94.18	40.22	84.05	84.26	79.46	33.72	82.72
150	85.92	63.99	95.47	15.86	25.09	91.47	**31.45**	77.06	**94.20**	35.30	86.46	80.78	76.36	33.07	82.20
200	87.05	61.88	95.34	**21.06**	26.90	**92.64**	31.16	76.31	94.09	32.96	**88.49**	78.60	73.54	34.17	81.97

Table 5.13: Class-wise classification metrics of the frame length variation experiment.

All averaged metrics in Table 5.14 clearly show that in this system the best results can be achieved with the lowest frame length of 60ms.

	overall scores			
	precision	recall	accuracy	AEER
60	**51.91**	**58.93**	**89.90**	**72.44**
100	50.52	58.57	89.48	78.25
150	48.98	57.13	88.82	84.33
200	49.15	57.55	88.53	87.47

Table 5.14: Overall classification metrics of the frame length variation experiment.

The F_1-scores in Table 5.15 are highest for all but the gunshot class when using a frame length of 60ms. The trend to worse results with increasing frame length is observable for all classes, except the gunshot class follows the opposite direction. Its results get better when decreasing the frame length. One reason for this contradictory behavior could be the fact that many features are based on the short-term Fourier spectrum and the frequency resolution could be too low to capture the gunshot.

	F_1 score					
	scream	gunshot	glassbreak	explosion	vocal	∅
60	**77.16**	15.97	**45.04**	**60.09**	**47.90**	**49.23**
100	76.67	18.19	44.86	54.41	47.34	48.29
150	73.35	19.44	44.67	50.13	46.15	46.75
200	72.34	**23.62**	44.25	48.03	46.66	46.98

Table 5.15: F_1 scores of the GMM/frame lengths experiment.

Varying the Feature Sets

Three different feature sets have been evaluated in this experiment. The parameters can be seen in Table 5.16. As previously described (cf. Section 2.4.3) the selection of the feature set is crucial and different classes may require different features.

parameter	**setting**
# GMM components	128
frame length	200ms
feature set	MFCCs, ΔMFCCs, MP7, TEO-CB-Auto-Env MFCCs and ΔMFCCs MP7 and TEO-CB-Auto-Env

Table 5.16: Setup for the feature sets variation experiment.

The results in Table 5.17 show the best classification metrics of all classes for the MP7+TEO feature set. The MFCC+ΔMFCC and the combined feature set deliver lower numbers in terms of performance compared to the MP7+TEO feature set.

	scream			gunshot			glassbreak			explosion			vocal		
FS	preci.	recall	accur.	preci.	recall	accur.	preci.	recall	accur.	preci.	recall	accur.	preci.	recall	accur.
all	87.05	61.88	95.34	21.06	26.90	92.64	**31.16**	76.31	**94.09**	32.96	**88.49**	78.60	73.54	34.17	81.97
MFCCs	60.00	50.22	91.80	10.72	16.11	90.77	6.12	19.43	88.35	37.36	67.04	83.75	51.89	27.00	77.37
MP7	**96.72**	**68.33**	**96.65**	**29.49**	**36.35**	**93.63**	30.74	**89.63**	93.47	**48.26**	58.05	**88.36**	**77.57**	**94.46**	**92.41**

Table 5.17: Class-wise classification metrics of the feature sets variation experiment.

The overall results in Table 5.18 also confirm the MP7+TEO feature set.

	overall scores			
FS	precision	recall	accuracy	AEER
all	49.15	57.55	88.53	87.47
MFCCs	33.22	35.96	86.41	101.80
MP7	**56.56**	**69.36**	**92.90**	**49.22**

Table 5.18: F_1 scores of the feature sets variation experiment.

Table 5.19 shows the F_1-scores, where the same results can be interpreted as with the basic classification metrics.

	F_1 score					
FS	scream	gunshot	glassbreak	explosion	vocal	∅
all	72.34	23.62	44.25	48.03	46.66	46.98
MFCCs	54.68	12.87	9.30	47.98	35.52	32.07
MP7	**80.08**	**32.56**	**45.78**	**52.71**	**85.18**	**59.26**

Table 5.19: F_1 scores of the feature sets variation experiment.

5.7.2 GMM Generative vs. Discriminative Training

In this experiment different learning algorithms for the GMMs have been compared. The generative approach using the EM algorithm as well as the conditional-log-likelihood and max-margin optimization have been used. The setup of the experiment is shown in Table 5.20.

parameter	**setting**
# GMM components	128
frame length	200ms
feature set	MP7, TEO-CB-Auto-Env
algorithm	generative conditional-likelihood maximum-margin

Table 5.20: Setup for the generative vs. discriminative training experiment.

Table 5.21 shows the results of the three different learning strategies. From experiments on synthetic data in [63] it is expected that the Maximum-Margin GMM delivers the best performance, followed by the Conditional-Likelihood GMM and the generatively trained GMM. The experiments with real-life acoustic data underlines this trends but there are exceptions for the individual classes.

	scream			gunshot			glassbreak			explosion			vocal		
#	preci.	recall	accur.	preci.	recall	accur.	preci.	recall	accur.	preci.	recall	accur.	preci.	recall	accur.
EM	**96.72**	68.33	96.65	29.49	**36.35**	93.63	30.74	**89.63**	93.47	48.26	58.05	88.36	77.57	**94.46**	92.41
CL	95.28	92.23	**98.78**	60.69	32.22	96.25	55.59	79.26	97.41	85.93	**77.10**	**96.03**	**87.59**	80.90	92.94
MM	93.98	**92.64**	98.69	**62.94**	33.97	**96.36**	**58.43**	77.18	**97.61**	**86.61**	75.39	95.95	86.17	83.36	**93.07**

Table 5.21: Class-wise classification metrics of the generative vs. discriminative training experiment.

Considering the system-wide metrics in Table 5.22, the training methods are ranked as expected. The MM-GMM delivers the best results, followed by the CLL-GMM and the generatively trained GMM.

	overall scores			
#	precision	recall	accuracy	AEER
EM	56.56	69.36	92.90	49.22
CL	77.02	72.34	96.28	24.23
MM	**77.62**	**72.51**	**96.34**	**23.66**

Table 5.22: Overall classification metrics of the generative vs. discriminative training experiment.

Based on the F_1-score in Table 5.23 gunshot and glassbreak work best with a max-margin optimized GMM. The difference in the score between conditional-likelihood and max-margin optimized GMMs is generally slim over all classes. Scream and gunshot deliver the best results witl CL GMMs, the exception is the vocal class which could be modeled best with an EM-trained GMM.

	F_1 score					
#	scream	gunshot	glassbreak	explosion	vocal	∅
EM	80.08	32.56	45.78	52.71	**85.18**	59.26
CL	**93.73**	42.09	65.35	**81.28**	84.11	73.31
MM	93.30	**44.12**	**66.51**	80.61	84.74	**73.86**

Table 5.23: F_1 scores of the generative vs. discriminative training experiment.

5.7.3 SVM vs. GMM

In the following experiments GMMs and SVMs are compared. The frame length was fixed to 200ms and a GMM with 128 components was used. The two different feature sets were used. Because traditional SVMs can only be used for binary classification problems, they were trained in a *one-versus-all* fashion. That means, the positive samples contain the samples belonging to the class to be detected by that particular SVM and the negative data was composed by samples from all other classes and background sounds. It was expected that the discriminative nature of SVMs brings advantages in terms of classification performance. The classification metrics are only given class-wise because the system treats each class independently. A system-wide AEER is also not available in this case.

The SVM training and testing has been done with `svmtrain` and accordingly `svmclassify` from MATLAB's Statistics and Machine Learning Toolbox [74]. Radial basis functions have been used and the iterations were limited to 10^6. The CPU and RAM requirements for SVM training and testing were enormous. One training fold took more than four hours, therefore the experiment runs were limited.

MFCC + ΔMFCC Features

Table 5.24 shows the results for the MFCC feature set. The SVM classifiers delivers much better results than the GMM except for the explosion class.

	scream			gunshot			glassbreak			explosion			vocal		
#	preci.	recall	accur.	preci.	recall	accur.	preci.	recall	accur.	preci.	recall	accur.	preci.	recall	accur.
GMM	60.00	50.22	91.80	10.72	16.11	90.77	6.12	19.43	88.35	37.36	**67.04**	83.75	51.89	27.00	77.37
SVM	**96.72**	**68.33**	**96.65**	**29.49**	**36.35**	**93.63**	**30.74**	**89.63**	**93.47**	**48.26**	58.05	**88.36**	**77.57**	**94.46**	**92.41**

Table 5.24: Class-wise classification metrics of the SVM vs. GMM with MFCC feature set experiment.

The F_1-scores in Table 5.25 show a big improvement when using SVMs for the scream, gunshot, glassbreak and vocal event. The explosion's score is also better but the difference is not large. The given average is not valid for a system as such, because in this topology each SVM operates on its own.

	F_1 score					
	scream	gunshot	glassbreak	explosion	vocal	∅
GMM	54.68	12.87	9.30	47.98	35.52	32.07
SVM	**80.08**	**32.56**	**45.78**	**52.71**	**85.18**	**59.26**

Table 5.25: F_1 scores of the SVM vs. GMM with MFCC feature set experiment.

MP7 + TEO-CB-Auto-Env Features

When using the MP7 + TEO-CB-Auto-Env features the results in Table 5.26 show that the GMM can outperform the SVM classifier more often in terms of precision and recall. The overall accuracy is always best with the SVM classifiers.

	scream			gunshot			glassbreak			explosion			vocal		
#	preci.	recall	accur.	preci.	recall	accur.	preci.	recall	accur.	preci.	recall	accur.	preci.	recall	accur.
GMM	**96.72**	68.33	96.65	29.49	**36.35**	93.63	30.74	**89.63**	93.47	48.26	58.05	88.36	77.57	**94.46**	92.41
SVM	96.18	**72.18**	**96.98**	**61.73**	19.84	**96.09**	**56.63**	55.46	**97.32**	**76.87**	**72.99**	**94.53**	**91.01**	75.81	**92.69**

Table 5.26: Class-wise classification metrics of the SVM vs. GMM with MP7 feature set experiment.

Table 5.27 shows that in terms of F_1 score a well-trained GMM can be better than an SVM-based system, this is the case for the gunshot and vocal class.

	F_1 score					
	scream	gunshot	glassbreak	explosion	vocal	∅
GMM	80.08	**32.56**	45.78	52.71	**85.18**	59.26
SVM	**82.47**	30.03	**56.04**	**74.88**	82.71	**65.23**

Table 5.27: F_1 scores of the SVM vs. GMM with MP7 feature set experiment.

It is very likely that the proposed *one-to-many* classification scheme is not suited very well for this task, e.g. a binary tree based scheme could perform better. Dedicated multi-class SVMs will also be more reasonable for this task.

5.7.4 Summary of the Results

Table 5.28 summarizes the F_1-scores for all experiments conducted in this work. The overall best scores are achieved with conditional-likelihood and max-margin optimized GMMs for all events but the vocal/human sound. SVMs with *one-to-many* classification were unable to outperform well-optimized GMMs.

	F_1 score					
	scream	gunshot	glassbreak	explosion	vocal	∅
GMM 60ms	77.16	15.97	45.04	60.09	47.90	49.23
GMM 100ms	76.67	18.19	44.86	54.41	47.34	48.29
GMM 150ms	73.35	19.44	44.67	50.13	46.15	46.75
GMM 200ms	72.34	23.62	44.25	48.03	46.66	46.98
GMM 64K	70.33	25.37	46.93	47.22	44.91	46.95
GMM 128K	72.76	23.02	45.21	48.23	45.70	46.98
GMM 256K	73.75	23.62	43.63	48.34	46.71	47.21
GMM MFCC	54.68	12.87	9.30	47.98	35.52	32.07
GMM MP7	80.08	32.56	45.78	52.71	**85.18**	59.26
GMM CL	**93.73**	42.09	65.35	**81.28**	84.11	73.31
GMM MM	93.30	**44.12**	**66.51**	80.61	84.74	**73.86**
SVM MFCC	80.08	32.56	45.78	52.71	85.18	59.26
SVM MP7	82.47	30.03	56.04	74.88	82.71	65.23

Table 5.28: F_1 scores of all experiments conducted.

6 Conclusion

In this thesis the theoretical and practical aspects of the acoustic detection of general events were discussed. The concept of human auditory scene analysis was explained, leading to computational auditory scene analysis, where the human's auditory abilities are going to be transferred to a machine. Based on this findings an acoustic event detection system can be derived. Various systems, requirements and topologies were explained with special focus on audio signal feature extraction in Chapter 3 and machine learning algorithms in Chapter 4. This work concludes with the implementation of an acoustic event detection system. Its parts and parameters were explained and experiments with MATLAB were conducted. It was shown that the choice of features, parameters and classifiers has a direct influence on the classification performance. For most of the experiments GMMs were used for classification. All GMM-related experiments showed different results for the event classes, except for the feature set and learning algorithm variation. When examining different frame lengths, a tendency to getting better results with shorter values was observed. The optimal number of components is obviously highly dependent of the event type to be detected. Some event classes are modeled better with a lower number of components while others perform better with a higher number. Basically, the results are similar to other work published using similar feature selection and classifiers. The MP7+TEO feature set was consistently delivering the best metrics for all event classes. The MFCC feature set performed the worst and the combination of the MFCC and MP7+TEO feature set was the second best. The classification results are not getting better just by increasing the length of the feature vector. As seen in this experiment, adding the wrong features was degrading the performance of the system. When evaluating discriminative learning algorithms for GMMs, the classification results improved significantly. Combining a good learning algorithm with the best feature set of the previous experiment, a medium number of components and a fairly large frame length lead to the best performance over all experiments with the Max-Margin GMM, followed by the CLL-GMM.

Replacing the generatively trained GMMs with SVMs results in better average classification results. The results vary between the classes and depend on the feature set in use. The MFCC feature set delivered much better results when used with SVMs instead of GMMs. It is very likely that the proposed usage of the *one-to-many* scheme is not the best for this task. Due to limited computational resources no special optimization has been done for improving the

performance with SVMs. The optimal model could not have been found and over-fitting cannot be ruled out. The discriminatively trained GMMs still deliver higher classification rates in this system.

Precision and recall are commonly used metrics for the performance of a classifier. The F_1-score combines them into a single measure, giving equal weight on precision and recall. The averaging method chosen has an impact on the value of these metrics. In this evaluation the metrics are calculated class-wise and are then averaged to equally weight the classes because of the imbalance of events in the audio data available as seen in Table 5.2. Accuracy has to be interpreted with care if a lot more background frames to be rejected by the classifiers than events to be detected are contained in test data, which is often the case. A high number of correctly rejected frames (i.e. true negatives) in contrary to a small number of true positives increase the accuracy value and reduce the information on detecting true positives is mostly lost. The purpose-designed AEER does not take correctly detected events into account but is a relation between wrongly detected or missed events and the ground truth number of events to be detected.

It is also very difficult to predict the generalization of the resulting models. They are highly dependent on the audio data used for training and validation. A huge sound library containing all event classes with a high degree of diversity, i.e. different situation, speakers, countries, etc. to rule out any over-fitting of the model. Cross-validation has been used to validate the model and reasonable classification rates were the result.

From a practical point of view, these classification metrics used during the experiments do not have much relevance in an operational situation. There is no guarantee that the correct model for the individual situation was selected. Data available for training is always limited, the ground truth over days, weeks or months of audio data is commonly not available for further processing, so the metrics cannot be calculated for a real-life situation. An operator in e.g. control room expects an event detection system to be supportive, i.e. the number of false alarms must be as low as possible to not be distracted from normal operation. On the other hand, relevant events must be detected immediately. If an event is missed, it is lost and usually there is no count of false negatives. To summarize, an event detection system has to prove itself in everyday use in the environment it has been designed for.

6.1 Outlook

The proposed acoustic event detection system was developed with real-time operation in mind. Practical field tests in a real environment would give a better understanding of how classification metrics can be transferred to an operational system and an insight of the system's quality. Block-processing is already implemented, an interface to live audio capture and performance tests in real-time mode are to be done.

To improve the detection results, more extensive model selection and the extension to a multi-stage detection system, e.g. pre-distinguishing of vocalic and non-vocalic sound events for better scream detection, should be considered. More computational resources would enable faster and therefore more training runs to calculate the optimal model. In the proceedings [71] of the recently held Detection and Classification of Acoustic Scenes and Events 2016 Workshop (DCASE2016) new trends in acoustic event detection are shown. Multichannel systems have become more popular, as well as Deep Neural Networks (DNNs). Convolutional Neural Nets (CNNs) and Recurrent Neural Nets (RNNs) with Long-Short-Term Memories (LSTMs) are now used. GMMs and HMMs are still utilized and perform well, especially in fused classification systems with DNNs, RNNs and LSTMs. Random decision trees are also capable of detecting events, no big changes are seen in the feature sets.

An alternative approach for increasing the detection rates and lowering the number of false

alarms is the fusion of detection systems operating in different domains, e.g. acoustic, video, motion, etc. for reaching higher detection rates while lowering the number of false alarms.

A Listings

```
% runfeaturextraction
% script performing step 1 of the aeds

clear all
close all
clc

aedinit

framesizes = [30 60 100 150] * 1e-3;

FEO.ovlp_pc = 0.75;
FEO.fs = 16000;
FEO.silence_pc = 0.1;
FEO.normalize = 1;
FEO.mfcc_ncoeff = 13;
FEO.mfcc_melbands = 36;
FEO.mfcc_cms = 1; % cepstral mean subtraction

scream_files = readFilelist('lib/scream_files.txt');
glassbreak_files = readFilelist('lib/glassbreak_files.txt');
gunshot_files = readFilelist('lib/gunshot_files.txt');
explosion_files = readFilelist('lib/explosion_files.txt');
vocal_files = readFilelist('lib/vocal_files.txt');
background_files = readFilelist('lib/background_files.txt');

for framesize = framesizes

    FEO.framelen = framesize;


    FS(1).features = getFeatureStruct(FEO,scream_files);
    FS(1).name = 'scream';

    FS(2).features = getFeatureStruct(FEO,gunshot_files);
    FS(2).name = 'gunshot';

    FS(3).features = getFeatureStruct(FEO,glassbreak_files);
    FS(3).name = 'glassbreak';

    FS(4).features = getFeatureStruct(FEO,explosion_files);
    FS(4).name = 'explosion';

    FS(5).features = getFeatureStruct(FEO,vocal_files);
    FS(5).name = 'vocal';

    FS(6).features = getFeatureStruct(FEO,background_files);
    FS(6).name = 'background';

    save(['FS' num2str(framesize*1000) 'ms.mat'], 'FS', 'FEO', '-v7.3');

end
```

Listing A.1: feature extraction ***run*** *script*

```
% extract features from file into feature struct
% AEDS project
% m.peitler

function FS = getFeatureStruct(FEO,file)

    if iscell(file)
        for f=1:length(file)
            try
                FS{f} = getFS(FEO,file{f});
            catch err
                FS{f}.error = err;
                clog(1,['#### -> error occurred while feature extraction! ' err.message]);
            end
        end
    else
        FS = getFS(FEO,file);
    end

end

function FS = getFS(FEO,file)

    clog(3,['calculating feature struct for ' file]);
    tic;
    FS.file = file;
    [FS.x,FS.file_fs] = audioread(FS.file);

    % preprocessing
    FS = multich2mono(FEO,FS);
    FS = audioResampler(FEO,FS);
    FS = audioNorm(FEO,FS);
    FS = audioSegmenting(FEO,FS);
    FS = removeSilentFrames(FEO,FS);
    FS = stft(FEO,FS);

    % feature extraction
    FS = mp7_awf(FEO,FS);
    FS = mp7_ap(FEO,FS);
    FS = mp7_asf(FEO,FS);
    FS = mp7_asc_ass(FEO,FS);
    FS = mp7_hr(FEO,FS);
    FS = mp7_ulh(FEO,FS);
%     FS = mp7_aff(FEO,FS);

    FS = mfccs(FEO,FS);
    FS = dMFCCs(FEO,FS);
    FS = rootCepstrumCoeffs(FEO,FS);

    FS = teo_cb_auto_env(FEO,FS);

    FS.feTime = toc;

end
```

Listing A.2: feature extraction script

```
% runtraining
% aeds project, m.peitler

CMT.tratio = 0.8;

CMT.type = 'dgmm_ml';
CMT.K = 128;
CMT.kmeans_maxiter = 100;
CMT.em_maxiter = 1000;

% framesizes_ms = [60 100 150];
framesizes_ms = [200];

CMT.featureset.mfcc = {'MFCC', 'dMFCC'};
CMT.featureset.mp7 = {'AWF','AP','ASC','ASS','HR','ULH','tcbae' };
CMT.featureset.all = {'MFCC','dMFCC','AWF','AP','ASC','ASS','HR','ULH','tcbae'};
CMT.features = CMT.featureset.mfcc;
% CMT.features = {'AWF','AP','ASC','ASS','HR','ULH','tcbae'};
for frmsz = framesizes_ms
    clear FS
    clear CM
    load(['FS' num2str(frmsz) 'ms.mat'])

    for k = 1:length(FS)
        [fv,CMT.Ni] = getFeatureVector(FS(k).features,CMT.features);
        [train,~] = dataSplitting(fv,CMT.tratio);

        CM(k) = cmTrain(CMT,train);

    end

    save(['cm_all_' num2str(frmsz) '.mat'], 'CM', 'CMT')

end
```

```
37  CMT.features = CMT.featureset.mp7;
38  CMT.K = 128;
39
40  load FS200ms
41  clear CM
42
43  for k = 1:length(FS)
44      [fv,CMT.Ni] = getFeatureVector(FS(k).features,CMT.features);
45      [train,~] = dataSplitting(fv,CMT.tratio);
46
47      CM(k) = cmTrain(CMT,train);
48
49  end
50  save('cm_mp7_200.mat','CM','CMT');
51
52  break
53
54  %%
55  CMT.name = 'scream';
56
57  CMT.features = {'MFCC','dMFCC','tcbae','HR'};
58
59  [scream_fv,CMT.Ni] = getFeatureVector(FS{1}.features,CMT.features);
60  [scream_train,scream_test] = dataSplitting(scream_fv,tratio);
61  clear scream_fv
62
63  CM(1) = cmTrain(CMT,scream_train);
64  %%
65  CMT.name = 'gunshot';
66  CMT.features = {'MFCC','dMFCC'};
67  [gunshot_fv,CMT.Ni] = getFeatureVector(FS.gunshot,CMT.features);
68  [gunshot_train,gunshot_test] = dataSplitting(gunshot_fv,tratio);
69  clear gunshot_fv
70
71  % CM(2) = cmTrain(CMT,gunshot_train);
72  %%
73  CMT.K = 256;
74  CMT.name = 'glassbreak';
75  CMT.features = {'MFCC','dMFCC','ASC'};
76  [glassbreak_fv,CMT.Ni] = getFeatureVector(FS.glassbreak,CMT.features);
77  [glassbreak_train,glassbreak_test] = dataSplitting(glassbreak_fv,tratio);
78  clear glassbreak_fv
79
80  % CM(3) = cmTrain(CMT,glassbreak_train);
81
82  %%
83  CMT.name = 'explosion';
84  CMT.features = {'MFCC','dMFCC'};
85
86  [explosion_fv,CMT.Ni] = getFeatureVector(FS.explosion,CMT.features);
87  [explosion_train,explosion_test] = dataSplitting(explosion_fv,tratio);
88  clear explosion_fv
89
90  % CM(4) = cmTrain(CMT,explosion_train);
```

Listing A.3: model training **run** *script*

```
1   % AEDS project
2   % class model training
3   % m.peitler, 12/13
4
5   function [CM] = cmTrain(CM,data,groups)
6
7       [CM.D,CM.N] = size(data);
8
9       switch CM.type
10
11          case 'diagGMM_nl'
12
13              % diagonal GMM of netlab toolbox
14              % k-means init followed by em
15
16              CM.g = gmm(CM.D,CM.K,'diag');
17              options = foptions;
18              options(1) = 1; % display errors
19              options(9) = 1; % check gradient calcs.
20              options(14) = CM.kmeans_maxiter;
21              CM.g = gmminit(CM.g,data.',options);
22              disp('k-means completed')
23
24              options = foptions;
25              options(1) = 1; % display errors
26              options(5) = 1; % cov checking
27              options(9) = 1; % check gradient calcs.
28              options(14) = CM.em_maxiter;
29              options(3) = 1e-6;
30
31              [CM.g,CM.ret,CM.errlog] = gmmem(CM.g, data', options);
32
33          case 'dgmm_ml'
34              options = statset('Display','iter','MaxIter',CM.kmeans_maxiter);
35              [idx] = kmeans(data.',CM.K,'Options',options);
36
```

```
37              options = statset('Display','iter','MaxIter',CM.em_maxiter);
38              CM.g = gmdistribution.fit(data.',CM.K,'Start',idx,'CovType','diagonal','Regularize',1,...
39                  'Options',options);
40
41          case 'svm'
42              opt = statset('Display','iter','MaxIter',50000);
43              CM.SVM = svmtrain(data.',groups,'Kernel_Function','rbf','options',opt);
44      end
45
46  end
```

Listing A.4: class model training script

Bibliography

[1] M. Gill and M. Hemming, "Evaluation of cctv in the london borough of lewisham," 2004.

[2] D. d. Waard, J. Godthelp, F. Kooi, and K. Brookhuis, *Human Factors, Security and Safety. Human Factors and Ergonomics Society Europe Chapter.* Shaker, 2009.

[3] D. Wang and G. J. Brown, *Computational auditory scene analysis: Principles, algorithms, and applications.* Wiley-IEEE Press, 2006.

[4] A. S. Bregman, *Auditory scene analysis: The perceptual organization of sound.* MIT press, 1994.

[5] L. P. A. S. van Noorden, "Temporal coherence in the perception of tone sequences," Ph.D. dissertation, Institute for Perceptual Research, Eindhoven, 1975.

[6] S. A. Shamma, M. Elhilali, and C. Micheyl, "Temporal coherence and attention in auditory scene analysis," *Trends in neurosciences*, vol. 34, no. 3, pp. 114–123, 2011.

[7] M. Slaney, "The history and future of casa," in *Speech separation by humans and machines.* Springer, 2005, pp. 199–211.

[8] V. Hohmann, "Signal processing in hearing aids," in *Handbook of Signal Processing in Acoustics.* Springer, 2009, pp. 205–212.

[9] C. ALVAREZ ANGULO, "Music recommender system," 2010.

[10] L. Barrington, R. Oda, and G. R. Lanckriet, "Smarter than genius? human evaluation of music recommender systems." in *ISMIR*, vol. 9, 2009, pp. 357–362.

[11] A. Klapuri and T. Virtanen, "Automatic music transcription," in *Handbook of Signal Processing in Acoustics.* Springer, 2009, pp. 277–303.

[12] P. Smaragdis and J. C. Brown, "Non-negative matrix factorization for polyphonic music transcription," in *Applications of Signal Processing to Audio and Acoustics, 2003 IEEE Workshop on.* IEEE, 2003, pp. 177–180.

[13] B. W. Schuller, *Intelligent audio analysis.* Springer, 2013.

[14] S. Ntalampiras, I. Potamitis, and N. Fakotakis, "An adaptive framework for acoustic monitoring of potential hazards," *EURASIP Journal on Audio, Speech, and Music Processing*, vol. 2009, p. 13, 2009.

[15] ——, "Acoustic detection of human activities in natural environments," *Journal of the Audio Engineering Society*, vol. 60, no. 9, pp. 686–695, 2012.

[16] C.-F. Chan and E. W. Yu, "An abnormal sound detection and classification system for surveillance applications," in *Signal Processing Conference, 2010 18th European.* IEEE, 2010, pp. 1851–1855.

[17] C. Clavel, T. Ehrette, and G. Richard, "Events detection for an audio-based surveillance system," in *Multimedia and Expo, 2005. ICME 2005. IEEE International Conference on.* IEEE, 2005, pp. 1306–1309.

[18] P. Giannoulis, G. Potamianos, A. Katsamanis, and P. Maragos, "Multi-microphone fusion for detection of speech and acoustic events in smart spaces," in *2014 22nd European Signal Processing Conference (EUSIPCO).* IEEE, 2014, pp. 2375–2379.

[19] J. M. Wiebe, R. F. Bruce, and T. P. O'Hara, "Development and use of a gold-standard data set for subjectivity classifications," in *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics.* Association for Computational Linguistics, 1999, pp. 246–253.

[20] T. Heittola, A. Mesaros, T. Virtanen, and A. Eronen, "Sound event detection in multisource environments using source separation," in *Workshop on machine listening in Multisource Environments*, 2011, pp. 36–40.

[21] C. Ittichaichareon, S. Suksri, and T. Yingthawornsuk, "Speech recognition using mfcc."

[22] A. Mesaros, T. Heittola, A. Eronen, and T. Virtanen, "Acoustic event detection in real life recordings," in *18th European Signal Processing Conference*, 2010, pp. 1267–1271.

[23] T. Heittola, A. Mesaros, A. Eronen, and T. Virtanen, "Audio context recognition using audio event histograms," in *Proc. of the 18th European Signal Processing Conference (EUSIPCO 2010)*, 2010, pp. 1272–1276.

[24] ——, "Context-dependent sound event detection," *EURASIP Journal on Audio, Speech, and Music Processing*, vol. 2013, no. 1, pp. 1–13, 2013.

[25] X. Zhuang, X. Zhou, T. S. Huang, and M. Hasegawa-Johnson, "Feature analysis and selection for acoustic event detection," in *Acoustics, Speech and Signal Processing, 2008. ICASSP 2008. IEEE International Conference on.* IEEE, 2008, pp. 17–20.

[26] H.-G. Kim and T. Sikora, "How efficient is mpeg-7 for general sound recognition?" in *Audio Engineering Society Conference: 25th International Conference: Metadata for Audio.* Audio Engineering Society, 2004.

[27] C. M. Bishop and N. M. Nasrabadi, *Pattern recognition and machine learning.* Springer New York, 2006, vol. 1.

[28] K. Delac, M. Grgic, and S. Grgic, "Independent comparative study of pca, ica, and lda on the feret data set," *International Journal of Imaging Systems and Technology*, vol. 15, no. 5, pp. 252–260, 2005.

[29] Y. Lu, I. Cohen, X. S. Zhou, and Q. Tian, "Feature selection using principal feature analysis," in *Proceedings of the 15th international conference on Multimedia.* ACM, 2007, pp. 301–304.

[30] L. Vuegen, B. Broeck, P. Karsmakers, J. Gemmeke, B. Vanrumste, and H. Hamme, "An mfcc-gmm approach for event detection and classification," in *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, 2013, pp. 1–3.

[31] E. Vozáriková, J. Juhár, and A. Čižmár, "Acoustic events detection using mfcc and mpeg-7 descriptors," in *International Conference on Multimedia Communications, Services and Security.* Springer, 2011, pp. 191–197.

[32] A. Temko, C. Nadeu, and J.-I. Biel, "Acoustic event detection: Svm-based system and evaluation setup in clear'07," in *Multimodal Technologies for Perception of Humans*. Springer, 2008, pp. 354–363.

[33] D. Ellis, "Detecting alarm sounds," in *Proc. Workshop on Consistent and Reliable Acoustic Cues CRAC-2000*, 2001.

[34] D. Ramage, "Hidden markov models fundamentals," *Lecture Notes. http://cs229. stanford. edu/section/cs229-hmm. pdf*, 2007.

[35] L. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989.

[36] J. Schmalenstroeer, M. Bartek, and R. Haeb-Umbach, "Unsupervised learning of acoustic events using dynamic time warping and hierarchical k-means++ clustering." in *INTERSPEECH*, 2011, pp. 305–308.

[37] D. Arthur and S. Vassilvitskii, "k-means++: The advantages of careful seeding," in *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*. Society for Industrial and Applied Mathematics, 2007, pp. 1027–1035.

[38] S. M. Siddiqi, G. J. Gordon, and A. W. Moore, "Fast state discovery for hmm model selection and learning," in *International Conference on Artificial Intelligence and Statistics*, 2007, pp. 492–499.

[39] J. Ramos, S. Siddiqi, A. Dubrawski, G. Gordon, and A. Sharma, "Automatic state discovery for unstructured audio scene classification," in *Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on*. IEEE, 2010, pp. 2154–2157.

[40] B. Zhou and J. Hansen, "Unsupervised audio stream segmentation and clustering via the bayesian information criterion," in *in Proc. ISCLP 2000*. Citeseer, 2000.

[41] S.-L. Chua, S. Marsland, and H. W. Guesgen, "Unsupervised learning of patterns in data streams using compression and edit distance," in *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*, vol. 22, no. 1, 2011, p. 1231.

[42] D. Giannoulis, E. Benetos, D. Stowell, M. Rossignol, M. Lagrange, and M. D. Plumbley, "Detection and classification of acoustic scenes and events: an ieee aasp challenge," in *2013 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*. IEEE, 2013, pp. 1–4.

[43] D. A. Sadlier and N. E. O'Connor, "Event detection in field sports video using audio-visual features and a support vector machine," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 15, no. 10, pp. 1225–1233, 2005.

[44] G. Ye, I.-H. Jhuo, D. Liu, Y.-G. Jiang, D. Lee, S.-F. Chang *et al.*, "Joint audio-visual bimodal codewords for video event detection," in *Proceedings of the 2nd ACM International Conference on Multimedia Retrieval*. ACM, 2012, p. 39.

[45] A. Temko, R. Malkin, C. Zieger, D. Macho, C. Nadeu, and M. Omologo, "Clear evaluation of acoustic event detection and classification systems," in *Multimodal Technologies for Perception of Humans*. Springer, 2007, pp. 311–322.

[46] A. Mesaros, T. Heittola, and T. Virtanen, "Metrics for polyphonic sound event detection," *Applied Sciences*, vol. 6, no. 6, p. 162, 2016.

[47] S. Quackenbush and A. Lindsay, "Overview of mpeg-7 audio," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 11, no. 6, pp. 725–729, 2001.

[48] H.-G. Kim, N. Moreau, and T. Sikora, *MPEG-7 audio and beyond: Audio content indexing and retrieval*. John Wiley & Sons, 2006.

[49] G. Peeters, "A large set of audio features for sound description (similarity and classification) in the cuidado project," http://recherche.ircam.fr/anasyn/peeters/ARTICLES/Peeters_2003_cuidadoaudiofeatures.pdf, 2004.

[50] J. Moorer, "The optimum comb method of pitch period analysis of continuous digitized speech," *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 22, no. 5, pp. 330–338, 1974.

[51] Y. D. Cho, M. Y. Kim, and S. R. Kim, "A spectrally mixed excitation (smx) vocoder with robust parameter determination," in *Acoustics, Speech and Signal Processing, 1998. Proceedings of the 1998 IEEE International Conference on*, vol. 2. IEEE, 1998, pp. 601–604.

[52] Y. D. Cho, H. K. Kim, M. Y. Kim, and S. R. Kim, "Pitch estimation using spectral covariance method for low-delay mbe vocoder," in *Speech Coding For Telecommunications Proceeding, 1997, 1997 IEEE Workshop on*. IEEE, 1997, pp. 21–22.

[53] J. F. Kaiser, "On teager's energy algorithm and its generalization to continuous signals," in *Proc. 4th IEEE digital signal processing workshop*, 1990.

[54] G. Zhou, J. H. Hansen, and J. F. Kaiser, "Nonlinear feature based classification of speech under stress," *Speech and Audio Processing, IEEE Transactions on*, vol. 9, no. 3, pp. 201–216, 2001.

[55] P. Maragos, T. F. Quatieri, and J. F. Kaiser, "Speech nonlinearities, modulations, and energy operators," in *Acoustics, Speech, and Signal Processing, 1991. ICASSP-91., 1991 International Conference on*. IEEE, 1991, pp. 421–424.

[56] E. Kvedalen, "Signal processing using the teager energy operator and other nonlinear operators," *Master, University of Oslo Department of Informatics*, 2003.

[57] J. Benesty, M. M. Sondhi, and Y. Huang, *Springer handbook of speech processing*. Springer, 2008.

[58] D. Jurafsky and J. H. Martin, *Speech & Language Processing*. Pearson Education India, 2000.

[59] H. Niemann, "Klassifikation von mustern, vol.3," http://www5.informatik.uni-erlangen.de/fileadmin/Persons/NiemannHeinrich/klassifikation-von-mustern/m00-www.pdf, 2007.

[60] F.-H. Liu, R. M. Stern, X. Huang, and A. Acero, "Efficient cepstral normalization for robust speech recognition," in *Proceedings of the workshop on Human Language Technology*. Association for Computational Linguistics, 1993, pp. 69–74.

[61] M. Westphal, "The use of cepstral means in conversational speech recognition." in *EUROSPEECH*, 1997.

[62] D. A. Reynolds, "Gaussian mixture models." http://www.ll.mit.edu/mission/communications/ist/publications/0802_Reynolds_Biometrics-GMM.pdf", 2009.

[63] F. Pernkopf and M. Wohlmayr, "Large margin learning of bayesian classifiers based on gaussian mixture models," in *Machine Learning and Knowledge Discovery in Databases*. Springer, 2010, pp. 50–66.

[64] Y. Guo, D. Wilkinson, and D. Schuurmans, "Maximum margin bayesian networks," *arXiv preprint arXiv:1207.1382*, 2012.

[65] B. Schölkopf and A. J. Smola, *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2002.

[66] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern classification*. John Wiley & Sons, 2012.

[67] T. Torbjørn, "How to construct a confusion matrix in latex?" http://tex.stackexchange.com/questions/20267/how-to-construct-a-confusion-matrix-in-latex", 2011.

[68] D. L. Olson and D. Delen, *Advanced data mining techniques*. Springer Publishing Company, Incorporated, 2008.

[69] J. S. Garofolo, L. D. Consortium *et al.*, "Timit: acoustic-phonetic continuous speech corpus," 1993.

[70] C. Clavel, I. Vasilescu, L. Devillers, and T. Ehrette, "Fiction database for emotion detection in abnormal situations," in *Proceedings of the International Conference on Spoken Language Processing*, 2004.

[71] T. Virtanen, A. Mesaros, T. Heittola, M. Plumbley, P. Foster, E. Benetos, and M. Lagrange, *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2016 Workshop (DCASE2016)*. Tampere University of Technology. Department of Signal Processing, 2016.

[72] M. Sokolova and G. Lapalme, "A systematic analysis of performance measures for classification tasks," *Information Processing & Management*, vol. 45, no. 4, pp. 427–437, 2009.

[73] I. Nabney, *NETLAB: algorithms for pattern recognition*. Springer, 2002.

[74] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, pp. 27:1–27:27, 2011, software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.