TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

Toni Mäkinen

Spatial and Content-based Audio Processing using Stochastic Optimization Methods

Julkaisu 1133 • Publication 1133

Tampere 2013

Tampereen teknillinen yliopisto. Julkaisu 1133
Tampere University of Technology. Publication 1133

Toni Mäkinen

Spatial and Content-based Audio Processing using Stochastic Optimization Methods

Thesis for the degree of Doctor of Science in Technology to be presented with due permission for public examination and criticism in Tietotalo Building, Auditorium TB222, at Tampere University of Technology, on the 21st of May 2013, at 12 noon.

Tampereen teknillinen yliopisto - Tampere University of Technology
Tampere 2013

ISBN 978-952-15-3075-3 (printed)
ISBN 978-952-15-3082-1 (PDF)
ISSN 1459-2045

Abstract

STOCHASTIC optimization (SO) represents a category of numerical optimization approaches, in which the search for the optimal solution involves *randomness* in a constructive manner. As shown also in this thesis, the stochastic optimization techniques and models have become an important and notable paradigm in a wide range of application areas, including transportation models, financial instruments, and network design. Stochastic optimization is especially developed for solving the problems that are either too difficult or impossible to solve analytically by deterministic optimization approaches.

In this thesis, the focus is put on applying several stochastic optimization algorithms to two audio-specific application areas, namely sniper positioning and content-based audio classification and retrieval. In short, the first application belongs to an area of spatial audio, whereas the latter is a topic of machine learning and, more specifically, multimedia information retrieval. The SO algorithms considered in the thesis are particle filtering (PF), particle swarm optimization (PSO), and simulated annealing (SA), which are extended, combined and applied to the specified problems in a novel manner. Based on their iterative and evolving nature, especially the PSO algorithms are often included to the category of evolutionary algorithms.

Considering the sniper positioning application, in this thesis the PF and SA algorithms are employed to optimize the parameters of a mathematical shock wave model based on observed firing event wavefronts. Such an inverse problem is suitable for Bayesian approach, which is the main motivation for including the PF approach among the considered optimization methods. It is shown – also with SA – that by applying the stated shock wave model, the proposed stochastic parameter estimation approach provides statistically reliable and qualified results.

The content-based audio classification part of the thesis is based on a dedicated framework consisting of several individual binary classifiers. In this work, artificial neural networks (ANNs) are used within the framework, for which the parameters and network structures are optimized based the desired item outputs, i.e. the ground truth class labels. The optimization process is carried out using a

multi-dimensional extension of the regular PSO algorithm (MD PSO). The audio retrieval experiments are performed in the context of feature generation (synthesis), which is an approach for generating new audio features/attributes based on some conventional features originally extracted from a particular audio database. Here the MD PSO algorithm is applied to optimize the parameters of the feature generation process, wherein the dimensionality of the generated feature vector is also optimized.

Both from practical perspective and the viewpoint of complexity theory, stochastic optimization techniques are often computationally demanding. Because of this, the practical implementations discussed in this thesis are designed as directly applicable to parallel computing. This is an important and topical issue considering the continuous increase of computing grids and cloud services. Indeed, many of the results achieved in this thesis are computed using a grid of several computers. Furthermore, since also personal computers and mobile handsets include an increasing number of processor cores, such parallel implementations are not limited to grid servers only.

Preface

This work has been carried out at the Department of Signal Processing, Tampere University of Technology, during 2008-2013. First, I wish to express my gratitude to my supervisor Prof. Serkan Kiranyaz for his support, guidance and belief in me during my thesis work. I would also like to thank my former supervisor Prof. Anssi Klapuri for introducing me to the field of audio signal processing and trusting and hiring me as a summer trainee to the Department of Signal Processing in 2007. I am also deeply grateful to Prof. Moncef Gabbouj for his patience and support during the research.

I am very grateful to the pre-examiners of my thesis, Dr. Kalle Palomäki and Dr. Antti Eronen, for their valuable and precise comments. Those provided several essential improvements and clarifications to the text and the overall quality of the thesis. I also address a special thank and gratitude for my opponents, Assistant Professor, Dr. George Tzanetakis and Academy Research Fellow, Dr. Kalle Palomäki.

I want to thank my colleagues, both the Audio Research Team (ART) and the MUVIS team for providing the pleasant working environment with relaxing coffee breaks and non-formal discussions. Especially, I want to thank Pasi Pertilä for inviting me to the TÄPLÄ project (Teknologiat Ääneen, Puheeseen ja moniaistisuuteen perustuvaan Läsnä-Älyyn) in 2007 and, more importantly, providing the guidance and support needed in taking the first steps of this work. Also many thanks to Tuomas Virtanen for helping me to continue my research after the TÄPLÄ project, leading the audio research team, and arranging several non-office activities during the years. Thanks to Hasse Sinivaara, Pasi Auranen, Teemu Korhonen and Antti Löytynoja for the help in making the gunshot recordings in 2008. Considering the work done in the MUVIS team, I am very grateful to Jenni Raitoharju and Stefan Uhlman for their valuable co-operation and help. Also thanks to the rest of my colleagues, including, but not limited to, Jouni Paulus, Marko Helén, Hanna Silén, Elina Helander, Joonas Nikunen and Mikko Roininen.

The funding and financial support of the Finnish Agency for Technology and

Innovation (TEKES) is gratefully acknowledged. This work was also partly supported by the Academy of Finland (Finnish Centre of Excellence Program 2006-2011).

Thanks to my parents Kristina and Kauko for their love and support during all my years of study (Kiitos!). My deepest thanks goes to my dear wife Marianne for her love, support and understanding during the times when I spent way more time sitting on a laptop than besides her. I'm also very grateful for the snacks and hugs you delivered in the middle of my writing sessions at evenings... That's true love!

Finally, I want to express my humble and profound gratitude to my Saviour, Jesus Christ. If it wasn't for you, among many other good things in my life, I don't think this thesis would exist.

Toni Mäkinen
Tampere, April 2013

Contents

List of Figures

List of Tables

List of Algorithms

List of Abbreviations

Abbreviations

Term or acronym	Explanation
AF	Analytical Features
AGP	Artificial Global Best
ANMRR	Average Normalized Modified Retrieval Rank
ANN	Artificial Neural Network
AP	Average Precision
BC	Binary Classifier
BE	(Sub)-Band Energy
BER	(Sub)-Band Energy Ratio
BFS	Basic Feature Set
BP	Back-Propagation
BW	Bandwidth
CNBC	Collective Network of Binary Classifiers
CE	Classification Error
CPA	Closest Point of Approach
CPU	Central Processing Unit
CRLB	Cramér-Rao Lower Bound
CV	Class Vector
DCT	Discrete Cosine Transform
DFT	Discrete Fourier Transform
DM	Discrimination Measure
DOA	Direction of Arrival
ECOC	Error Correcting Output Codes
EFS	Evolutionary Feature Synthesis
FGBF	Fractional Global Best Formation
FV	Feature Vector
GA	Genetic Algorithm
GCC	Generalized Cross Correlation

Term or acronym	Explanation
GP	Genetic Programming
GMM	Gaussian Mixture Model
HMM	Hidden Markov Model
HPSO	Heterogeneous Particle Swarm Optimization
IDFT	Inverse Discrete Fourier Transform
KDD	Knowledge Discovery in Databases
LDA	Linear Discriminant Analysis
LPC	Linear Prediction Coefficient
LPCC	Linear Prediction Cepstral Coefficient
MC	Monte Carlo
MCMC	Markov Chain Monte Carlo
MDPSO	Multi-Dimensional Particle Swarm Optimization
MFCC	Mel-Frequency Cepstral Coefficient
ML	Maximum Likelihood
MLP	Multilayer Perceptron
MSE	Mean Squared Error
MST	Minimum Spanning Tree
NBC	Network of Binary Classifiers
NMRR	Normalized Modified Retrieval Rank
PCA	Principal Component Analysis
PDF	Probability Density Function
PF	Particle Filter
PHAT	PHAse Transform
PSO	Particle Swarm Optimization
RF	Random Forest
SA	Simulated Annealing
SAE	Spectral Average Energy
SC	Spectral Centroid
SF	Spectral Flux
SFV	Segment Feature Vector
SIR	Sampling Importance Sampling
SIS	Sequential Importance Sampling
SLP	Single-Layer Perceptron
SNR	Signal-to-Noise Ratio
SPR	Statistical Pattern Recognition
SR	Spectral Roll-off
SS	Spectral Spread
STFT	Short Time Fourier Transform
SVM	Support Vector Machine
TDOA	Time Difference Of Arrival
TOA	Time of Arrival
ZCR	Zero-Crossing Rate

List of Included Publications

This thesis is a compound thesis and is based on the following publications:

P1 **Toni Mäkinen, Pasi Pertilä, and Pasi Auranen**, Supersonic bullet state estimation using particle filtering. *IEEE International Conference on Signal and Image Processing Applications (ICSIPA)*, pages 150 – 155, Kuala Lumpur, Malaysia, November 2009.

P2 **Toni Mäkinen and Pasi Pertilä**, Shooter localization and bullet trajectory, caliber, and speed estimation based on detected firing sounds. *Applied Acoustics*, vol. 71, pages 902 – 913, 2010.

P3 **Toni Mäkinen, Serkan Kiranyaz, and Moncef Gabbouj**, Content-based audio classification using collective network of binary classifiers. *IEEE Workshop on Evolving and Adaptive Intelligent Systems (EAIS)*, pages 116 – 123, Paris, France, April 2011.

P4 **Serkan Kiranyaz, Toni Mäkinen, and Moncef Gabbouj**, Dynamic and scalable audio classification by collective network of binary classifiers framework: An evolutionary approach. *Neural Networks*, vol. 34, pages 80 – 95, 2012.

P5 **Toni Mäkinen, Serkan Kiranyaz, Jenni Pulkkinen, and Moncef Gabbouj**, Evolutionary feature generation for content-based audio classification and retrieval. *The 20th European Signal Processing Conference (EUSIPCO)*, pages 1474 – 1478, Bucharest, Romania, August 2012.

P6 **Toni Mäkinen, Serkan Kiranyaz, Jenni Raitoharju, and Moncef Gabbouj**, An evolutionary feature synthesis approach for content-based audio retrieval. *EURASIP Journal on Audio, Speech, and Music Processing*, vol. 2012, no. 23, pages 1 – 23, 2012.

The publications are cited as [P1], [P2], etc.

Author's Contributions to the Publications

The author of the thesis acts as the main author in all the included publications, except for [P4]. In [P1], most of the implementations, geometric derivations and all the simulations and tunings (in terms of heuristic enhancements) were done by the author. The fundamental idea was provided by Dr. Pasi Pertilä, who also suggested using the mathematical model applied in both [P1] and [P2]. The data gathering for [P1] and [P2] was done by the author with the assistance of Hasse Sinivaara, Pasi Auranen, Dr. Teemu Korhonen and Antti Löytynoja. The optimization algorithm applied in [P2] and all the simulations and evaluations were implemented and performed by the author. Application of the optimization algorithm implemented for the problem was first suggested by T. Korhonen.

The classifier topology and the underlying modified particle swarm optimization algorithm applied for classification in [P3] and [P4] were first invented and implemented by Prof. Serkan Kiranyaz and Prof. Moncef Gabbouj. The database collection, audio feature implementations and feature extraction, framework conversion from image-specific to audio-specific, and all the evaluations and parameter testing in [P3] and [P4] were performed by the author. Also the modifications and additional implementations required for running the framework in a computational grid were done by the author. The core implementations regarding to [P5] and [P6] were done by Jenni Raitoharju (former Pulkkinen), and the original idea of the feature synthesis approach was suggested by Prof. Serkan Kiranyaz. The proposed optimization enhancements (simulated annealing in [P5] and heterogeneous particle behaviour in [P6]) were suggested to the context and implemented by the author. Also the required conversions, e.g. in the sense of file formats – from the original image and video domain to audio domain – all the evaluations and all parameter tunings in both [P5] and [P6] were performed by the author. Finally, in all the publications except [P4] the author has done most of the writing work.

Chapter 1

Introduction

IMAGINE yourself walking in a war zone with several obstacles and covered hiding places. Suddenly, without a warning, you hear a very sharp and short "snapping" sound nearby, immediately revealing that you have been detected by a hostile sniper. Being unable to determine the direction of the sound, you rush in to find a quick cover. As you have no cue about the shooter location, all you can do is remain still and quiet. In such a situation, obtaining even a small hint of the sniper direction – or ideally the exact location – would be of great importance, not to mention estimating the trajectory, caliber and speed of the fired bullet. This information would significantly improve one's chances of survival from the situation.

In another scenario, imagine yourself wandering in a city side walk and listening to your favourite music with headphones. As you are connected to an on-line database, you decide to browse for new music clips similar in content to that you are currently listening (i.e., representing the same genre). The browsing system thus performs a gradual similarity search over the database – also called as content-based retrieval or a "query-by-example" – and proposes the most similar music pieces found for you. In a broader sense, such content-based audio retrieval is applicable to various types of databases other than music. For example, you may want to search for film scenes including a specific actor/actress based on his/her tone of voice, or you might be simply interested in finding out all the, say, dog barking samples from a database consisting of animal sounds. As can be seen, the spectrum of potential applications is wide in this area.

The above examples demonstrate two fundamentally different types of audio-related estimation problems. Despite their differences, however, in this thesis it is shown that similar audio signal processing and stochastic optimization methods can be applied to them providing satisfactory estimation results. As can be seen, it may be desired to estimate the *direction* of an incoming sound wave, whereas in some cases we are more interested about the *content* of a received audio. Combining the two targets may be also desirable in some cases: considering the

second scenario a bit further, imagine that while listening to your music, a car approaches you from behind. As you cannot hear the car and carelessly decide to cross the street through a cross walk, you almost directly step in front of the car and force the driver brake heavily to avoid contact. Although one should be more careful than this in a traffic, in such situations an automatic detection and recognition of incoming vehicles would definitely enhance safety.

Even if it would be – in some cases – possible to manage the aforementioned situations by ourselves (as we humans do have remarkable abilities created in determining sound source directions and recognize the audio contents [Rum12, Chapter 2]), in many cases it would still be very beneficial to perform the estimation automatically. For example, automatic area surveillance, intelligent cars with related (mobile) devices and robot technology are all potential areas for these kind of audio processing solutions. In short, spatial and content-based audio analysis and estimation provide interesting and important possibilities for several potential application areas. Sound source localization, sound wave *direction of arrival* (DOA) estimation, and content-based audio recognition have all been studied actively in the audio signal processing field. It is thus highly probable to see an increasing number of novel and (also) commercially interesting innovations around the topic in the near future.

1.1 Spatial Audio Analysis

As a term, the *spatial audio analysis* involves estimating the DOA of different sound sources in a specific environment [Rum12, Chapter 1]. The sound sources may be fixed to a certain position, or they may be moving around the spatial environment. Furthermore, there may be either a single or several simultaneous sound sources active, where in the latter case specific source separation methods [Man10] or, alternatively, some *detection* and content recognition algorithm may be required. In this work, a modified version of the detection algorithm described in [Kla06, Chapter 4] is applied, which is to be discussed in Chapter 3.

The most commonly applied DOA estimation techniques are based on *time-difference-of-arrival* (TDOA) estimations between several microphones [Ben08]. The microphones are typically mounted on an array with relatively small distances between each other. A soldier helmet is considered and mimicked for this purpose in the shooter localization application of this thesis. In order to estimate a bullet trajectory and, hence, the shooter location, two separate stochastic optimization methods are applied. The first one, *particle filtering*, is a sequential *Monte Carlo* approach relying on Bayesian inference [Kal09], while the second approach is based on the *simulated annealing* algorithm [vL87]. Both approaches – together with the corresponding stochastic optimization methods – are discussed in detail in Chapters 2 and 3.

1.2 Content-based Audio Management

Knowledge discovery in databases (KDD) [PS91] is a general term used to describe a process of creating and obtaining new knowledge from particular sources. The analysis step of KDD is called *data mining* [Han11], which is a field concentrating on automatic discovery of previously unknown patterns from the input data. The term often overlaps with *machine learning* [Dud01], which focuses on construction and study of systems that can *learn* from data. This includes predicting and *recognizing* the content of a specific media, such as image, audio, or video. The machine learning algorithms can be divided to *supervised* and *unsupervised* learning categories [Dud01]. The principal difference between the two is that in supervised learning a separate *training* dataset with correct output values (the ground truth information) is required, whereas methods belonging to the unsupervised learning category aim to *model* the input data without any form of training involved. The learning algorithms can be used to train specific *classifiers*. These are essentially systems which, given a specific data sample input, output the class label to which the input sample most likely belongs to. In this thesis, the content analysis focus is put on supervised learning.

An adaptive classifier *network* topology is proposed and described for the audio content analysis purposes of this work. The classifier network is called as *collective network of binary classifiers* (CNBC), which analyses and combines the outputs of several distinct classifiers with binary outputs. These binary classifiers are *evolved* by using the *multi-dimensional particle swarm optimization* (MD PSO) [Kir09] algorithm. The network topology is specifically designed for versatile and *dynamic* databases, meaning that audio classes with highly varying content and dynamic number of samples and classes are handled in a fluent and efficient manner.

1.3 Audio Feature Generation

In many cases where data classification or machine learning in general is to be performed, certain *features* (also called as *attributes*) need to be extracted from the input data. Attempts have been made in the past research to generate new and enhanced audio features in the sense of classification and retrieval performance. These approaches share a common principal idea of *selecting* (feature selection) and *modifying* (feature generation) some set of low-level audio features. Stochastic evolutionary algorithms – usually *genetic algorithms* (GA) [Gol89] – are applied also in this field for the selection and generation processes. However, the number of successful and generalizable audio feature generation methods in the literature is still rather limited, as is comprehensively discussed in [Pac09].

In this thesis, the MD PSO algorithm applied to the CNBC network is also employed to generate new audio features with enhanced class discrimination abil-

ity. In addition to the actual features, the approach also provides the possibility to optimize the *dimensionality* of the generated output feature vector.

1.4 Outline of the Thesis

Put in a single sentence, this doctoral thesis concentrates on applying specific stochastic and *evolutionary* optimization methods in the areas of spatial audio source localization and content-based audio classification and retrieval. The fundamental optimization methods applied throughout the thesis are first introduced and discussed in Chapter 2. These methods include, together with their derivatives, particle filtering (PF) [Aru02], particle swarm optimization (PSO) [Ken95], and simulated annealing (SA) [vL87]. As already discussed above, the spatial audio research concentrates on a military application, in which the goal is to detect and localize hostile snipers using a microphone array. This application is discussed in detail in Chapter 3. The work related to content-based audio analysis is the topic of Chapter 4, in which the focus is drawn mainly on describing the indexing (classification) of versatile and dynamic audio databases using the dedicated classifier network. The feature generation approach regarding to indexing and retrieval tasks is discussed in Chapter 5, where the topic in general is also brought into a broader discussion. Finally, the main observations and conclusions of the thesis are drawn in Chapter 6, after which the included publications and errata are attached at the end.

1.5 Main Results of the Thesis

The main results of the thesis are listed in this section, separately for each publication and the two main application areas.

1.5.1 Shooter Localization

The publications [P1] and [P2] consider the problem of gunshot detection, bullet trajectory, caliber and speed estimation, and shooter localization.

Publication 1

In publication [P1], particle filtering is applied for estimating the trajectory, caliber and speed (i.e. the *state*) of a supersonic bullet. In short, introduction of such a stochastic Bayesian inference approach to the estimation is the main contribution of the article. In the process, mathematical shock wave modelling is used together with measured gunshot data to allow applying the Bayesian approach. The problem is of high dimension with an irregular likelihood distribution, which is a suitable venue for stochastic particle filters. The method is evaluated by

gunshot simulations and real gunshot recordings, and the results demonstrate a high convergence ability of the applied PF on high-dimensional optimization problems. The achieved estimation accuracy is well-comparable to other trajectory estimation approaches proposed in the literature, with the advantage of not requiring any prior information of the weapon or terrain types.

Publication 2

Publication [P2] broadens the research made in [P1] by including the gunshot *detection* phase to the research scope and by increasing the amount of both simulated and recorded gunshot data. Instead of particle filtering, simulated annealing is applied to the optimization problem with two different fitness measures. The first fitness measure is the one applied also in [P1], i.e., a generalized cross correlation (GCC) - based signal comparison, while the mean-squared error (MSE) is considered as the second measure. These measures allow performing both time- and frequency domain fitness evaluation. The research shows the capability of SA in providing satisfactory solutions for the multi-dimensional optimization problem encountered in the estimation process. In a broader sense, it is shown that the shooter localization and bullet state estimation problem can be described as a multi-dimensional optimization problem and solved by applying stochastic optimization approaches. The obtained results are comparable or better than those achieved with the state-of-the-art approaches.

1.5.2 Content-based Audio Classification and Retrieval

The rest of the publications are related to content-based audio classification and retrieval problems.

Publication 3

An approach of constructing a *network* of binary classifiers for audio classification is proposed in publication [P3]. The principal idea is to construct a separate binary classifier network for each audio class involved to a classification problem. The solution allows combining a large amount of different types of audio features, as each network can be used to divide the extracted audio features into several feature subsets. Not all features can distinguish all types of audio classes, which is why the subset providing the best classification result is favoured among the others by a dedicated fuser classifier. The described classifier topology allows also to dynamically increase or decrease the number of classes (and samples within) without necessarily retraining the classifier network from scratch. This provides a fast adaptation to any changes occurring in a database. In this work, artificial neural networks (ANNs) are used as the individual binary classifiers, which are "evolved" by searching both the optimal network parameters and *structures* in

a large search space using a *multi-dimensional* extension of the particle swarm optimization algorithm. Improvements in classification accuracy of an 8-class database were obtained in compared to support vector machine (SVM) classifiers.

Publication 4

Publication [P4] extends the research of [P3] in terms of audio features and classes. In addition, it also presents an extensive set of comparative evaluations. The article demonstrates the ability of the proposed CNBC topology to provide comparable and enhanced classification results against other state-of-the-art classifiers with their best parameter combinations found. The *scalability* property of the framework is further demonstrated by increasing the number of audio classes in the tested database and adding new binary classifier networks to the framework accordingly. The achieved classification results show successful dynamic adaptation of the framework to significant database variations.

Publication 5

Audio features play an essential role in any content-based classification and retrieval task. Due to this, a novel method for generating new and enhanced features out of the existing ones is proposed in publication [P5]. The approach is based on the MD PSO algorithm, which is applied in both selecting an optimal combination of audio features over an existing low-level feature set, as well as generating new (artificial) features by modifying the selected ones based on the optimization process. Also SA is applied in this paper to further assist the stochastic search process in converging to the global optimum of the set fitness function. The new features are tested with audio clustering, classification and retrieval tasks, where clear improvements are obtained compared to the original features. The successful application of MD PSO to feature generation with such a performance improvement is the major contribution of the article.

Publication 6

In publication [P6], the feature generation approach of [P5] is extensively experimented and compared against ANN classifiers. Instead of SA used in [P5], a heterogeneous particle behaviour approach is applied to enhance the convergence of the MD PSO algorithm. Moreover, two different optimization fitness functions are used in the experiments to illustrate the possibility of applying problem-specific fitness measures to the framework. The evolutionary aspect of the approach, which refers to feature generation (synthesis) over several individual synthesis *runs*, is especially on the scope of this paper. In most experiments, the results indicate notable performance improvements. Moreover, by enabling all the properties of the proposed feature synthesis technique, it is also shown

that – when compared to the original features and the ANN classifier – the approach is scalable with respect to the database size and the number of features extracted.

Chapter 2

Stochastic Optimization Methods

MANY engineering problems encountered in real-world applications require solving high-dimensional optimization tasks in a reasonable computational time. Often such problems cannot be solved in a deterministic manner, causing a need for alternative, stochastic approximation approaches. This chapter introduces the *three* main stochastic optimization methods applied to the optimization problems considered in this thesis, namely particle filtering, particle swarm optimization, and simulated annealing.

2.1 Particle Filtering

Many dynamic system models can be expressed in a *state-space* form. The idea behind the form is to capture the system *state* affecting behind an observed system output. One of the basic properties of the form defines that the future state values depend only on the *current* state in a probabilistic manner. The observed system outputs are also presumed to be related to the state through a specific observation equation.

2.1.1 Markov Chains

Assume we have a set of system states, $S = s_1, s_2, \ldots, s_n$. In *Markov Chains* [Gri97], the system process starts in one of the states and continues moving from one state to another according to certain *transition probabilities*. The probability of moving from a current state s_i to state s_j is denoted by p_{ij}, and it is not dependent upon which states the chain has visited before the current state s_i (the first order Markov property). Note that the process can also remain in the current state with probability p_{ii}. The starting state is usually specified based on the initial probability distribution of the system model [Gri97, Chapter 11].

In many scientific problems, in order to provide estimates of the state (and system) evolution, state-space modelling of the dynamic system at hand is needed. For this, two specific models are required: the system model and the measurement model. The former describes the state evolution with respect to time, while the latter is to relate the observed measurements to the underlying state [Aru02]. Thus, a sequence of (noisy) measurements – called as measurement vector – is required to build a state-space model for a particular dynamic system. The state-space approach is especially convenient with non-linear and non-Gaussian problems, such as the bullet state estimation problem discussed in this thesis. Analytical solutions, such as *Kalman filter* [Kal60], cannot be directly applied to such problems.

2.1.2 Markov Chain Monte Carlo Simulation

Monte Carlo (MC) techniques are approximate *inference* methods based on random numerical sampling [Kal09, Chapter 1]. They are thus stochastic processes, defined in [Kal09] as *sequences of states whose evolution is determined by random events*. Regarding to several types of system model probability distributions encountered in real-world applications, the Markov chain Monte Carlo (MCMC) [Ber04] simulation allows sampling from a wide range of such distributions. This is achieved by constructing a Markov chain that has the desired target distribution as its *equilibrium distribution*. For an *ergodic* Markov chain, the equilibrium distribution is defined as the (stationary) distribution, to which the chain converges when the number of steps approaches infinity. The chain ergodicity means that the convergence will occur irrespective of the choice of the initial system model distribution, and it can be shown that a homogeneous Markov chain will be ergodic with most target distributions and transition probabilities [Nea93]. At each step of the algorithm, a candidate sample is generated from the current distribution, which is then accepted according to some probability-based criterion. The state of the chain after a large number of transitions (steps) is taken as a sample from the desired distribution. Typically the MCMC sampling can only approximate the target distribution, and the sample quality is dependent on the number of steps performed in the chain [Bis06, Chapter 11].

2.1.3 Sequential Importance Sampling

As is shown in this work, probabilistic state-space modelling and formulation is directly applicable to Bayesian approach [Can11]. Here the idea is to construct a *posterior* probability density function (PDF) of the state at certain time (or iteration) index t, based on *prior* probabilities and the obtained measurements. Usually the state estimate is required every time a new measurement is obtained, in which case a *recursive* solution should be considered. This means *sequential* data processing (filtering), in which two essential phases are encountered: *predic-*

tion and *update*. The system model is applied in the prediction phase, where the state PDF is predicted forward from one measurement time to the next. In the update phase, instead, the latest measurement is used to modify (and correct) the predicted PDF. The process can be performed using Bayes theorem [Can11, Chapter 2].

The *sequential importance sampling* (SIS) algorithm [Aru02] is essentially a Monte Carlo method. The term "importance sampling" refers to sampling from a proposal distribution (called an importance density) that is different from the actual distribution of interest (posterior). The purpose is to provide estimates of the posterior distribution, from which it is often difficult to draw samples directly. The SIS approach is alternatively called particle filtering, bootstrap filtering, or survival of the fittest, and it forms the basis for most sequential MC filters proposed in the literature. It can be used for *approximating* a recursive Bayesian filter by MC simulations, where the posterior PDF is represented by a set of random samples (also called as "particles") with associated *weights*. These weights are approximations of the relative posterior probabilities of the particles, determined for each particle based on the corresponding particle value. The SIS filter approaches the optimal Bayesian estimate as the number of samples (particles) increases [Aru02].

Denoting a particle a at iteration t as $\mathbf{p}_a[t]$ and the system output observation matrix as $\mathbf{O}$, the Bayes theorem can be written as

$$P(\mathbf{O}|\mathbf{p}_a[t]) \propto L(\mathbf{p}_a[t]|\mathbf{O})P(\mathbf{O}), \tag{2.1}$$

where $P(\mathbf{O})$ is the (predicted/determined) *prior* measurement probability distribution, $L(\mathbf{p}_a[t]|\mathbf{O})$ is the *likelihood* function defined by an applied measurement model, $P(\mathbf{O}|\mathbf{p}_a[t])$ is the updated (posterior) distribution, and $\propto$ stands for proportionality. In other words, the formula reveals the probability of observing the measurement $\mathbf{O}$, given that the system state is the one denoted by the particle $\mathbf{p}_a[t]$.

2.1.4 Sampling Importance Resampling Filter

The SIS particle filter suffers from a *degeneracy* problem, meaning that the particle weights become negligible after a few iterations. To avoid this problem, a specific *resampling* algorithm can be applied whenever significant degeneracy is obtained. Here the main idea is to remove particles with low weights and focus on only those with large weights. The removed particles are replaced by resampling from the approximated posterior distribution.

A special case of the SIS algorithm is derived by choosing an appropriate importance density and performing the resampling step at every iteration t. This approach is called *sampling importance resampling* (SIR) filter [Aru02]. The

defined prior distribution is commonly applied as the importance density, as in [P1] where the SIR filter is applied for bullet trajectory, caliber, and speed estimation (forming the bullet state).

In [P1], a set of particles is applied and evaluated based on their locations in a pre-determined search space. The particle weights are calculated using a fitness (or likelihood) function obtained from the applied measurement model. After the weights are calculated, the particle positions are updated according to a *Brownian motion* [Mör10]. The position of particle a is then replaced by the updated one if

$$P(\mathbf{p}_a[\star], \mathbf{p}_a[t]) \geq \alpha, \tag{2.2}$$

where $\mathbf{p}_a[\star]$ represents the new candidate particle to replace the existing $\mathbf{p}_a[t]$ at iteration t, and α is a random number with uniform distribution over $(0, 1)$. The probability $P(\mathbf{p}_a[\star], \mathbf{p}_a[t])$ is defined as

$$P(\mathbf{p}_a[\star], \mathbf{p}_a[t]) = \min\left(1, \frac{W(\mathbf{p}_a[\star])}{W(\mathbf{p}_a[t])}\right), \tag{2.3}$$

where $W(\cdot)$ represents the particle weight. This means that whenever the update from $\mathbf{p}_a[t]$ to $\mathbf{p}_a[\star]$ increases the weight value, the position replacement is certain to be done, i.e. $\mathbf{p}_a[t+1] = \mathbf{p}_a[\star]$. On the other hand, also transitions to lower weight positions may occur in a probabilistic manner. This update approach is called *Metropolis* algorithm [Bis06, Chapter 11].

As a summary, particle filtering is a *sequential* Monte Carlo method, where the probability densities are represented by specific particles. Particle filters are the sequential (online) analogue of MCMC batch methods, and can be often performed much faster than MCMC. Note that, as the problem in [P1] does not include tracking, the processing is performed $I = 20$ times for each observation $\mathbf{O}$. A pseudo-code of the SIR filter with P particles is provided in Algorithm 1.

2.2 Particle Swarm Optimization

Iterative particle, or "point mass", approach is also employed in another stochastic optimization method called particle swarm optimization [Ken95]. Instead of approximating Bayesian estimation as in particle filtering, PSO is based on mathematical modelling of natural swarms, such as bird flocks or fish schools. It can be also combined with PF to overcome some of its limitations, as in e.g. [Wan06]. More specifically, Zhang et al. [Zha08] show theoretically that, in a Bayesian inference view, a sequential PSO framework is actually analogous to *multilayer* importance-sampling-based particle filter.

Input: Observed system measurement **O**
Output: Estimated system state S

```
Initialize the particles according to prior distribution P(O).
for t ← 0 to I do
    for a ← 1 to P do
        Draw sample (particle) from the current state distribution
        Calculate the sample weight W(p_a[t])
        Update p_a[t] using the Metropolis algorithm (Eq. (2.2), (2.3)).
    end
    Resample all particles with the systematic resampling algorithm [Aru02].
end
```

Algorithm 1: SIR particle filter with the Metropolis algorithm

As a population-based stochastic search process, in PSO, each particle of the swarm represents a potential solution for an underlying optimization task. As in PF, a (randomized) position vector is assigned for each particle, after which the particles navigate within a pre-defined search space searching for the global optimum of a (possibly) non-linear function or system. Based on their movement, the particles gain "experience" by keeping track of their *cognitive* and *social* components. The former component stands for the personal best position found so far by the corresponding particle, while the latter one indicates the global best position found among all the particles of the swarm in an iterative search process [Mik08].

The *velocity* of a particle a at iteration $t+1$, $\mathbf{v}_a[t+1]$, is affected by the mentioned components and the current velocity as

$$\mathbf{v}_a[t+1] = w[t]\mathbf{v}_a[t] + c_1\mathbf{r}_1[t](\mathbf{y}_a[t] - \mathbf{p}_a[t]) + c_2\mathbf{r}_2[t](\hat{\mathbf{y}}[t] - \mathbf{p}_a[t]), \tag{2.4}$$

where $\mathbf{p}_a[t]$ stands for the current particle position, $\mathbf{y}_a[t]$ is the personal best position (the cognitive component), and $\hat{\mathbf{y}}[t]$ is the global best position of the swarm (the social component) found until the iteration index t. The term $w[t]$ is the inertia weight, c_1 and c_2 are accelerator constants and r_{1j} and r_{2j} are uniformly distributed random variables, i.e. $r_{nj} \sim \mathcal{U}(0,1)$ in each each dimension $j = \{1, \ldots, D\}$, where D is the search space dimensionality. Thus, the social and cognitive components contribute randomly to the particle velocity in the next iteration. The particle position is then updated using the velocity as

$$\mathbf{p}_a[t+1] = \mathbf{p}_a[t] + \mathbf{v}_a[t+1]. \tag{2.5}$$

The accelerator constants – which affect the significance of the cognitive and social components – are generally set to $c_1 = c_2 = 1.49$ or $c_1 = c_2 = 2.0$,

from which the latter is applied in this work. The inertia weight is used to control the velocity *memory* term, and it is usually – as in this work – decreased during the PSO iterations from 0.9 to 0.4 [Shi98]. A larger value of $w(t)$ favours particle exploration while a small inertia weight strives for exploitation. The two remaining PSO parameters are the number of particles applied and iterations performed in the process. These are highly dependent on the problem and are usually set experimentally with the trade-off between the optimization accuracy and computational cost. Also, in order not to exceed suitable particle positional range, application-specific limits can be set to bind the solutions to a proper range.

2.2.1 Multi-dimensional Particle Swarm Optimization

A*multi-dimensional* extension to PSO (MD PSO) [Kir09] allows the particles to navigate through multiple search space dimensions. Thus, instead of operating in a fixed dimension, D, the MD PSO algorithm is designed to seek both *positional* and *dimensional* optima within a certain dimension range, $\{D_{min}, D_{max}\}$. This also means that the length of the particle position vector varies according to the corresponding search space dimension.

In MD PSO, each particle possesses two sets of components (for positional and dimensional search), which are subjected to two independent and consecutive update processes. The first process corresponds to the traditional velocity updates of the regular PSO (equation (2.4)), while the second one updates the particle dimension. Accordingly, now the particles keep track of their position, velocity and personal best position in *every* dimension, so that whenever a particle revisits a particular dimension at a later time, its regular positional update process may continue using this information. The swarm, on the other hand, keeps track of the global best particle of each dimension, which are then used in the regular velocity updates performed in the corresponding dimension. Finally, each particle keeps also track of its personal best *dimension* visited so far (in which the personal best fitness score has been achieved), and the global best dimension is indicated by a corresponding dimensional social component. This means that the global best particle position in the global best dimension represents the optimum solution found so far.

Following the logic of the traditional PSO, the particle velocity update formula (2.4) changes to

$$\mathbf{v}_a^{d_a[t]}[t+1] = w[t]\mathbf{v}_a^{d_a[t]}[t] + c_1\mathbf{r}_1[t]\left(\mathbf{y}_a^{d_a[t]}[t] - \mathbf{p}_a^{d_a[t]}[t]\right) + c_2\mathbf{r}_2[t]\left(\hat{\mathbf{y}}^{d_a[t]}[t] - \mathbf{p}_a^{d_a[t]}[t]\right), \tag{2.6}$$

where the index term $d_a[t]$ is added to each term to illustrate the dimensional dependency. Similarly, the particle position is updated as

$$\mathbf{p}_a^{d_a[t]}[t+1] = \mathbf{p}_a^{d_a[t]}[t] + \mathbf{v}_a^{d_a[t]}[t+1]. \tag{2.7}$$

This means that the new particle position remains in the current dimension $d_a[t]$ after the positional update. The dimensional velocity update is thus performed separately for each particle at the end of the iteration round $t+1$ as follows:

$$vd_a[t+1] = \lfloor vd_a[t] + c_1 r_1[t]\,(yd_a[t] - d_a[t]) + c_2 r_2[t]\,(\hat{y}d[t] - d_a[t]) \rfloor, \tag{2.8}$$

where $vd_a[t]$ denotes the dimensional velocity at iteration t, $yd_a[t]$ and $\hat{y}d[t]$ are the personal and global best dimensions visited so far, respectively, and $\lfloor \cdot \rfloor$ is a floor operator. Analogous to the regular positional update, the new dimension is finally updated as

$$d_a[t+1] = d_a[t] + vd_a[t+1], \tag{2.9}$$

where the dimensional jump is allowed only if the target dimension $d_a[t+1]$ is within a set dimensional range, $d \in \{D_{min}, D_{max}\}$, where D_{min} and D_{max} correspond to the pre-specified minimum and maximum dimensions, respectively.

Similar to PF, the *fitness* (or weight) value of particle a at iteration t, $\mathcal{F}(\mathbf{p}_a^{d_a[t]}[t])$, is computed based on its current position in the search space. Note that the fitness is only evaluated within the current dimension of the particle, meaning that the positional parameters in all other dimensions remain the same for the next iteration round $t+1$, i.e., $\mathbf{p}_a^{d_a[t]}[t+1] = \mathbf{p}_a^{d_a[t]}[t]$, $\mathbf{v}_a^{d_a[t]}[t+1] = \mathbf{v}_a^{d_a[t]}[t]$, $\mathbf{y}_a^{d_a[t]}[t+1] = \mathbf{y}_a^{d_a[t]}[t]$, $\forall d \in \{D_{min}, D_{max}\} \wedge d \neq d_a[t]$. In the case of minimization, the cognitive and social components are then updated based on the fitness values as

$$\mathbf{y}_a^{d_a[t]}[t+1] = \begin{cases} \mathbf{y}_a^{d_a[t]}[t], & \text{if } \mathcal{F}(\mathbf{p}_a^{d_a[t]}[t]) > \mathcal{F}(\mathbf{y}_a^{d_a[t]}[t]) \\ \mathbf{p}_a^{d_a[t]}[t], & \text{else,} \end{cases} \tag{2.10}$$

and

$$\hat{\mathbf{y}}^d[t+1] = \begin{cases} \hat{\mathbf{y}}^d[t], & \text{if } \min\limits_a \left(\mathcal{F}(\mathbf{y}_a^d[t])\right) \geq \mathcal{F}(\hat{\mathbf{y}}^d[t]) \\ \underset{\mathbf{y}_a^d[t]}{\operatorname{argmin}} \left(\mathcal{F}(\mathbf{y}_a^d[t])\right), & \text{else,} \end{cases} \tag{2.11}$$

respectively. The term d in (2.11) refers to each dimension considered in the optimization task, i.e. $d \in \{D_{min}, \ldots, D_{max}\}$. The corresponding dimensional components are defined in a similar manner:

$$yd_a[t+1] = \begin{cases} yd_a[t], & \text{if } \mathcal{F}(\mathbf{p}_a^{d_a[t]}[t]) > \mathcal{F}(\mathbf{y}_a^{yd_a[t]}[t]) \\ d_a[t], & \text{else,} \end{cases} \tag{2.12}$$

for the cognitive and

$$\hat{y}d[t+1] = \begin{cases} \hat{y}d[t], & \text{if } \min_a \left(\mathcal{F}(\mathbf{y}_a^{yd_a[t]}[t])\right) \geq \mathcal{F}(\hat{\mathbf{y}}^{\hat{y}d[t]}[t]) \\ \underset{yd_a[t]}{\text{argmin}} \left(\mathcal{F}(\mathbf{y}_a^{yd_a[t]}[t])\right), & \text{else,} \end{cases} \tag{2.13}$$

for the social component, respectively.

For clarity, Figure 2.1 shows in a graphical form the difference between the particle structures of the MD PSO and the regular PSO algorithms. In the shown case, a particle a currently resides at dimension $d_a[t] = 2$, while its personal best dimension is $yd_a[t] = 3$. Hence, at iteration t, first a positional PSO update is performed over the positional components of $\mathbf{p}_a^2[t]$, after which the particle may move to another dimension based on the dimensional update process. A pseudo-code of the MD PSO algorithm is provided in Algorithm 2, and further details about the MD PSO algorithm are provided in [Kir09].

2.2.2 Fractional Global Best Formation

In order to avoid the so-called *premature convergence* problem occurring with (MD) PSO (which basically means converging into a local minimum in the search space), an *artificial* global best (aGB) particle can be formed. This is done by combining the most "attractable" individual particle vector *elements*, $p_{aj}[t], j \in \{1, \ldots d_a[t]\}$. The approach is called *fractional global best formation* (FGBF) [Kir10], and it requires evaluating a separate fitness value (or an estimate of it) for each particle element $p_{aj}[t]$. The idea is then to combine those particle elements with best fitness values through the entire swarm. The formed aGB particle then replaces the conventional global best whenever its overall fitness score is improved, i.e., if $\mathcal{F}(aGB[t]) \leq \mathcal{F}(\hat{\mathbf{y}}[t])$ (in the case of a minimization problem).

In the multi-dimensional case where several search spaces are considered in parallel, a separate aGB particle is defined for each dimension. With respect to this work, each aGB particle is formed by combining particle vector elements also from *different* dimensions. This is shown in Figure 2.2, where elements from three different particles of different dimensions are combined. This way the probability of obtaining a new aGB particle with improved fitness compared to the conventional global best particle is further increased. The FGBF procedure within the MD PSO algorithm is given in a pseudo-code form in Algorithm 3.

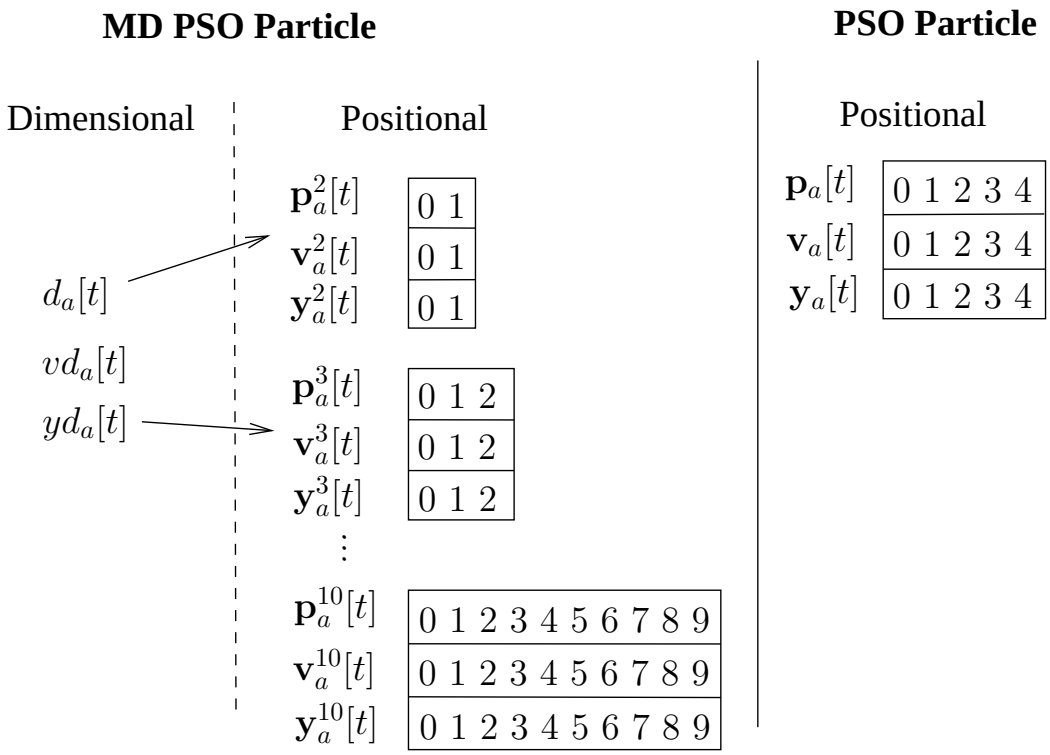

Figure 2.1: Particle structures of the MD PSO (left) and regular PSO (right) algorithms. The dimensional range of MD PSO is set to $\{D_{min} = 2, D_{max} = 10\}$, while the regular PSO dimensionality is fixed to $D = 5$. At the current iteration t, the MD PSO particle is located at dimension $d_a[t] = 2$, while its personal best dimension found so far is $yd_a[t] = 3$. Copyright© 2011 IEEE.

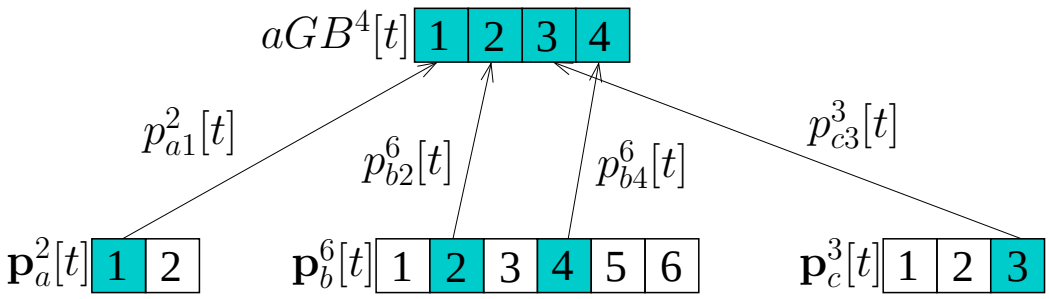

Figure 2.2: The formation of an aGB particle of dimension 4. The individual elements of three different particles, a, b, and c, having the dimensions 2, 6, and 3, respectively, are combined in the process. As shown in the figure, also several elements can be selected from a single particle.

2.2.3 Heterogeneous Particle Behaviour

Updating the particle positions in a swarm is an essential part of the PSO algorithm. Depending on the optimization problem, it may be beneficial to dy-

Input: MD PSO parameters
Output: The global best particle

Initialize the MD PSO particles according to prior distribution.
for $t \leftarrow 0$ **to** I **do**
    `// for each particle:`
    for $a \leftarrow 1$ **to** P **do**
        Compute the fitness of particle a, $\mathcal{F}(\mathbf{p}_a^{d_a[t]}[t])$
        if $\mathcal{F}(\mathbf{p}_a^{d_a[t]}[t]) \leq \mathcal{F}(\mathbf{y}_a^{d_a[t]}[t])$ **then**
            `// update the personal best position (Eq. (2.10)):`
            $\mathbf{y}_a^{d_a[t]}[t+1] = \mathbf{p}_a^{d_a[t]}[t]$
            if $\mathcal{F}(\mathbf{p}_a^{d_a[t]}[t]) < \mathcal{F}(\hat{\mathbf{y}}^{d_a[t]}[t])$ **then**
                `// update the global best particle of dimension` $d_a[t]$`:`
                $\hat{\mathbf{y}}^{d_a[t]}[t+1] = a$
            end
            if $\mathcal{F}(\mathbf{p}_a^{d_a[t]}[t]) \leq \mathcal{F}(\mathbf{y}_a^{yd_a[t]}[t])$ **then**
                `// update the personal best dimension (Eq. (2.12)):`
                $yd_a[t+1] = d_a[t]$
            end
            if $\mathcal{F}(\mathbf{p}_a^{d_a[t]}[t]) < \mathcal{F}(\hat{\mathbf{y}}^{\hat{y}d[t]}[t])$ **then**
                `// update the global best dimension:`
                $\hat{y}d[t+1] = d_a[t]$
            end
        end
    end
    for $a \leftarrow 1$ **to** P **do**
        Update the velocity of particle a using (2.6)
        Update the position of particle a using (2.7)
        Update the dimensional velocity of particle a using (2.8)
        Update the dimension of particle a using (2.9)
    end
end

Algorithm 2: The MD PSO algorithm

namically vary the *behaviour* of the particles to better adapt to the underlying search space. Such an idea was first proposed by Engelbrecht in [Eng10], where, in addition to the regular particle behaviour shown in (2.4), *four* optional update procedures are considered. Applying these behaviour models to MD PSO has not been proposed earlier in the literature.

Input: MD PSO particles
Output: Artificial global best (aGB) particle

Define $[j] = \text{argmin}_{a \in \{1,P\}}(\mathcal{F}(p_{aj}^{d_a[t]}[t]))$.
Among all the particles $a \in \{1, P\}$, select the best particle index b for each element j, $b[j]$. ($j \in \{1, D_{max}\}$)
for $d \leftarrow D_{min}$ **to** D_{max} **do**
    `// Assign the best elements into the aGB solution:`
    for $j \leftarrow 1$ **to** d **do**
        $aGB_j^d[t] = p_{b[j]j}^{d_{b[j]}[t]}[t]$
    end
    if $\mathcal{F}(aGB^d[t]) < \mathcal{F}(\hat{\mathbf{y}}^d[t])$ **then**
        $\hat{\mathbf{y}}^d[t] = aGB^d[t]$
    end
end
`// Re-evaluate:`
$\hat{y}d[t] = \text{argmin}_d(\mathcal{F}(\hat{\mathbf{y}}^d[t]))$

Algorithm 3: The FGBF algorithm in MD PSO

Cognitive-only Model

The first two optional behaviour models restrict the conventional particle update procedure by concentrating solely on either the cognitive or the social particle component. Thus, in the cognitive-only model, the social terms $\hat{\mathbf{y}}^{d_a[t]}[t]$ and $\hat{y}d[t]$ are removed from (2.6) and (2.8) (meaning the latter summation terms). Such a modification leads to a broader particle exploration due to the loss of interaction between them. This also causes every particle to become an independent *hill-climber* in the search space, meaning that they essentially perform local stochastic search in the search space. This, instead, means that the artificial global best particle discussed in section 2.2.2 is not applied with this model.

Social-only Model

Following the logic of the previous model, the social-only model removes the cognitive terms $\mathbf{y}_a^{d_a[t]}[t]$ and $yd_a[t]$ from (2.6) and (2.8) (the first summation terms). This causes faster particle exploitation, as now the whole swarm becomes a single stochastic hill-climber.

Barebones Model

The Barebones model [Ken03] changes the particle velocity as

$$\mathbf{v}_a^{d_a[t]}[t+1] \sim \mathcal{N}\left(\frac{\mathbf{y}_a^{d_a[t]}[t] + \hat{\mathbf{y}}^{d_a[t]}[t]}{2}, \sigma\right), \tag{2.14}$$

where $\sigma = \left|\mathbf{y}_a^{d_a[t]}[t] - \hat{\mathbf{y}}^{d_a[t]}[t]\right|$. The particle position update (equation (2.7)) is also changed to $\mathbf{p}_a^{d_a[t]}[t+1] = \mathbf{v}_a^{d_a[t]}[t+1]$, meaning that the updated velocity sampled from the provided Gaussian distribution is assigned as such as the new particle position. As proposed in [P6], the behaviour is applied to the dimensional velocity analogously as

$$vd_a[t+1] \sim \mathcal{N}\left(\frac{yd_a[t] + \hat{y}d[t]}{2}, \sigma\right), \tag{2.15}$$

where $\sigma = |yd_a[t] - \hat{y}d[t]|$. The obtained dimensional velocity is considered as the new dimension, i.e. $d_a[t+1] = vd_a[t+1]$.

At the beginning of the MD PSO algorithm the personal best particle positions are usually far away from the global best solution, causing a large deviation to the above Gaussian distribution. Thus, this behaviour model facilitates an *initial* particle exploration throughout the search space, whereas – after more PSO iterations are performed – the deviation converges to zero. This, instead, causes the particles to focus on exploitation of the personal and global best average position as the algorithm proceeds.

Modified Barebones Model

The final behaviour model modifies the Barebones model by introducing an additional probability-based exploration to it [Ken03]. This is shown in the following equations, where the velocity is updated as

$$\mathbf{v}_a^{d_a[t]}[t+1] = \begin{cases} \mathbf{y}_a^{d_a[t]}[t], & \text{if } \mathcal{U}(0,1) < 0.5 \\ \mathcal{N}\left(\frac{\mathbf{y}_a^{d_a[t]}[t] + \hat{\mathbf{y}}^{d_a[t]}[t]}{2}, \sigma\right), & \text{else,} \end{cases} \tag{2.16}$$

and similarly for dimensional update ([P6]):

$$vd_a[t+1] = \begin{cases} yd_a[t], & \text{if } \mathcal{U}(0,1) < 0.5 \\ \mathcal{N}\left(\frac{yd_a[t] + \hat{y}d[t]}{2}, \sigma\right), & \text{else.} \end{cases} \tag{2.17}$$

The exploration is increased in the modified version at the initial stages, as 50% of the time the focus is on the personal best position. Once the process converges, the behaviour turns to exploitation due to the convergence of the personal best positions towards the global best solution.

The initial behaviour of each particle is obtained *randomly* among the shown behaviour models. However, whenever a specific particle is not able to improve its fitness value within 10 consecutive iteration rounds, a new behaviour is again assigned for it in a random manner. This new behaviour model may then help

the particle in getting out from potential local optima and eventually provide improvements to the particle fitness value. Thus, in addition to FGBF, the heterogeneous PSO (HPSO) is another solution for avoiding the premature convergence phenomenon of regular PSO.

The techniques introduced in this section are applied in the publications as follows: MD PSO is applied in [P3] – [P6], where the FGBF is applied in [P5] and [P6], and HPSO in [P6].

2.3 Simulated Annealing

The third stochastic optimization method considered in this thesis is simulated annealing (SA) [vL87]. The method is named after the annealing process encountered in metallurgy, which is a technique involving controlled *cooling* of a material. The method is particularly applicable to optimization problems [Kir83], for which it possesses a specific decreasing "temperature" term T and a somewhat modified version of the Metropolis algorithm applied with particle filtering.

Being an iterative optimization method, each iteration in the SA algorithm proposes a new random *neighbour* solution to replace the current best one. Usually the proposed solution is chosen somewhere near the best solution – hence the name – and is accepted with a certain probability. The probability is dependent on both the *difference* between the likelihood/fitness values of the best and the suggested solutions, and the (gradually decreasing) global temperature parameter T. The idea is to allow frequent solution updates at the beginning of the process (even to positions with lower fitness value), and then "cool down" the system by decreasing the parameter T and the update probability accordingly.

Simulated annealing is used in [P2] to optimize (with two separate optimization criteria) the parameters for a bullet state likelihood function. Furthermore, in [P5], SA is applied *together* with the MD PSO algorithm, such that the global best particle position is further optimized by applying SA at the end of each PSO iteration. Such a combination of PSO and SA has been proposed earlier e.g. in [Zha05], whereas the multi-dimensional aspect provided by the MD PSO has not been considered earlier in the field. Thus, a pseudo-code representing the combined MD PSO and SA algorithm is provided in Algorithm 4. The terms are defined as follows: I_{PSO} and I_{SA} stand for the number of iterations applied for MD PSO and SA, respectively, P is the total number particles, u stands for a problem-specific update constant, $randn(d)$ is a d-dimensional Gaussian random vector, and C denotes a cooling constant, $C < 1$. As discussed above, it can be noticed that at the beginning of each SA process, the parameter T has a high value (T_0). This results to a nearly random selection between the current and the neighbour solution during the first iterations. However, as T decreases the selection begins to favour the better solution (converging towards "downhill" in a case of minimization problem). The additional allowance of "uphill" move-

Input: Required system parameters
Output: The best obtained solution for the given optimization task

Initialize the MD PSO particles according to prior distribution.

`// MD PSO loop:`
for $t \leftarrow 0$ **to** I_{PSO} **do**
    `// for each particle:`
    for $a \leftarrow 1$ **to** P **do**
        Update the cognitive and social particle components as shown in Algorithm 2
    end
    `// SA loop (for each dimension` d `in a pre-defined range):`
    for $d \leftarrow D_{min}$ **to** D_{max} **do**
        Initialize the temperature T_0; Set $k = 0$;
        while $k < I_{SA}$ **do**
            Generate a neighbour solution: $nGB^d = \hat{\mathbf{y}}^d[t] + randn(d) \times u$;
            Evaluate $\mathcal{F}(nGB^d)$;
            `// difference between the solutions:`
            Compute $\Delta = \mathcal{F}(nGB^d) - \mathcal{F}(\hat{\mathbf{y}}^d[t])$;
            `// Metropolis:`
            if $\min(1, \exp(-\Delta/T_k)) > rand(0,1)$ **then**
                $\hat{\mathbf{y}}^d[t] = nGB^d$;
            end
            Set $T_{k+1} = CT_k$;
        end
    end
    Update the particle positions and dimensions using (2.7) and (2.9) (or other behaviour model).
end

Algorithm 4: Simulated annealing combined with the MD PSO algorithm

ments – adopted from the Metropolis algorithm – potentially avoids the method in becoming stuck at any local optima during the process [vL87].

As a final statement of this chapter, the summarized stochastic optimization methods and their derivatives represent a rather broad spectrum of the state-of-the-art approaches available in the area. Genetic algorithms are not applied in this thesis due to their rather common existing usage in the problems considered in this work. As shown in the following chapters, the introduced methods can be successfully applied to such optimization tasks where analytical deterministic solutions are not either available or feasible. For an interested reader, simulation-based comparisons of deterministic and stochastic optimization algorithms are performed and discussed in [Wet04].

Chapter 3

Spatial Audio Analysis and Firing Event Estimation

SPATIAL audio analysis as a term includes, but not limits to, space-, direction- and location-related audio analysis using single or several microphones. Usually microphone *arrays* with several microphones are required for decent spatial analysis. This chapter introduces some of the most commonly known methods applied to sound direction of arrival estimation, which is an essential subtopic in the spatial audio area. Also the practical application of shooter localization and bullet trajectory, caliber and speed estimation is gone through in detail, beginning by describing a general gunshot event scene with the related audible sounds and geometrical measures. A gunshot detection and recognition algorithm is also required prior the actual estimation procedure, which is discussed and detailed in the chapter.

3.1 Directional Audio Analysis

Selecting and implementing a suitable DOA estimation algorithm is an essential step in a spatial audio analysis. From a psychoacoustics point of view, there are two primary sound perception mechanisms or cues used by humans in detecting sound directions. These involve the time or phase differences of the sound signal between our two ears, and the amplitude or spectral differences between the ears. The time and phase difference is mainly caused by the different sound propagation distances from the sound source to the ears, whereas the spectral cues result from the filtering effects of ears, head and the whole body. Although also some monaural cues have been shown to exist, in most cases it is the *differences* in the received signal that matter the most [Rum12].

3.1.1 Background and General Methods

This section shortly introduces two commonly applied DOA estimation methods in the literature.

Beamforming

Beamforming is a spatial filtering technique that can be used for DOA estimation by forming directional spatial *beams* with desired patterns. The technique is applied over an array of sensors, which are usually put in line with respect to each other. The principal idea is based on weighting and *combining* the signals of each sensor with different phases, such that signals at particular angles experience constructive interference while others are suppressed. This allows computing intensity measures towards different directions, which can be used to analyse the potential signal DOA.

The most common methods used to create the directional beams are based on time delay (time shift) and phase shift [Mon04]. Time domain beamforming is achieved by delaying the microphone signals, adjusting their amplitude, and summing the signals over the array of microphones to steer the beam towards a particular location where potential sources might be present. This approach is called *delay and sum*. Correspondingly, beamforming in the frequency domain is achieved by applying these phase shifts and amplitude adjustments to the microphone signals. However, such *phase shift* beamformer is only suitable for well-defined narrowband signals [Liu10].

A more sophisticated version of delay and sum introduces applying filters to the sensor channels. This approach is called – accordingly – *filter and sum* and it is discussed in more detail in [Liu12].

Time-difference-based DOA estimation

Because of its accuracy, fairly simple implementation and popularity in the field [Bal10], this thesis concentrates solely on time-difference-based DOA estimation. Assuming an array of two microphones and an external sound source with a different distance to the two microphones, a deterministic delay occurs between received microphone signals. This delay can be measured using a *cross-correlation* between the microphone channels. Usually a *generalized* cross correlation (GCC) [Kna76] is applied for the task, defined as

$$C(k) = \mathrm{IDFT}\left(\Psi_{12} X_1 X_2^*\right), \tag{3.1}$$

where IDFT stands for an inverse discrete Fourier-transform, X_1 and X_2 are Fourier-transforms of the observed signals $x_1(k)$ and $x_2(k)$ of microphone channels 1 and 2, respectively, and $(\cdot)^*$ stands for a complex conjugate. The term Ψ_{12}

corresponds to frequency-dependent *weighting* of the cross-spectral magnitudes, for which several values can be applied. The most commonly used is the phase transform (PHAT) frequency weighting [Kna76], defined as

$$C(k) = \mathrm{IDFT}\left(\frac{X_1 X_2^*}{|X_1||X_2^*|}\right). \tag{3.2}$$

This weighting provides a *flat* magnitude spectrum for the signals, meaning that only the phase information is utilized in computing the correlation coefficients $C(k)$. The delay τ between the channels, also known as a *time-difference of arrival* (TDOA) measure, is then obtained as

$$\tau = \underset{k}{\mathrm{argmax}}\left(C(k)\right), \tag{3.3}$$

in which the values $\tau \in (-\tau_{max}, \tau_{max})$ are of interest, τ_{max} being a geometry-based maximum delay between the channels.

3.1.2 Limitations of Time Delay Estimation

Certain fundamental limitations exist in the scope of time delay estimation. Especially when cross correlation-based methods are applied, *signal-to-noise ratio* (SNR) plays a key role considering the estimation accuracy. In case the SNR decreases below a specific limiting value, a dramatic loss of accuracy occurs in the delay estimation. This value is called *threshold* SNR, and it is widely known in the field [Ash05]. The phenomenon follows from the fact that in high noise levels the cross correlation function becomes more or less flat with no distinguishable maximum peak. The threshold value can be evaluated numerically by using the source signal bandwidth B, duration N, and central frequency F_c as follows:

$$SNR_{th} = \frac{6}{\pi^2(BN)}\left(\frac{F_c}{B}\right)^2\left[\phi^{-1}\left(\frac{B^2}{24F_c^2}\right)\right]^2, \tag{3.4}$$

where $\phi(y) = 1/\sqrt{2\pi}\int_y^\infty e^{-k^2/2}dk$ [Ash05]. This means that the threshold is proportional to the inverse of the product BN and the relation B/F_c. The occurring phenomenon can be clearly noticed in the simulation sections of the publications [P1] and [P2], where Gaussian noise is gradually increased to simulated gunshot events; at some point the estimates degenerate rapidly as a function of decreasing SNR level.

A so-called *Cramér-Rao lower bound* (CRLB) [Wei84] can be used to predict a lower limit for the time delay estimation error. It provides the value that is asymptotically *approached* by a maximum likelihood (ML) estimator with specific SNR and BN values. In [Sad06], the lower bound is defined as

$$CRLB = \frac{1}{SNR \times \beta^2}, \tag{3.5}$$

where β^2 is a *mean square bandwidth* of the source signal $x(k)$, defined as

$$\beta^2 = \frac{\int_0^N \left(\frac{dx(k)}{dk}\right)^2 dk}{\int_0^N x^2(k)dk} = \frac{\int_{-\infty}^{\infty}(2\pi f)^2|X(f)|^2df}{\int_{-\infty}^{\infty}|X(f)|^2df}, \tag{3.6}$$

in which $X(f)$ is the Fourier transform of signal $x(k)$ and f stands for frequency. It can be seen that the CRLB is lowered, i.e. the estimation accuracy is increased, by increasing the SNR or the signal bandwidth. The signal waveform or central frequency are not affecting the lower bound significantly. The CRLB works reliably only when $SNR > SNR_{th}$, which is why also other lower bounds have been proposed in the literature. These are discussed in more detail in [Sad06].

3.1.3 Direction of Arrival Estimation

Once the time delays between different microphones are obtained, the direction of the received sound wave can be estimated. Assume we have a microphone array with *four* microphones at known positions in Cartesian coordinates, $\mathbf{m}_n = [m_{nx}, m_{ny}, m_{nz}], n = \{1, \dots 4\}$, meaning that six different microphone pairs can be formed. Defining then a *sensor vector* from microphone 1 to microphone 2 as $\mathbf{x}_{12} = \mathbf{m}_2 - \mathbf{m}_1$ and the corresponding time delay as $\tau_{12} = \mathbf{x}_{12}^T\mathbf{k}$, where $\mathbf{k}$ is the sound wave propagation vector, the following relation can be formed between the terms:

$$\mathbf{X}\mathbf{k} = \mathbf{t}, \tag{3.7}$$

where

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_{12} \\ \mathbf{x}_{13} \\ \mathbf{x}_{14} \\ \mathbf{x}_{23} \\ \mathbf{x}_{24} \\ \mathbf{x}_{34} \end{bmatrix}, \quad \mathbf{t} = \begin{bmatrix} \tau_{12} \\ \tau_{13} \\ \tau_{14} \\ \tau_{23} \\ \tau_{24} \\ \tau_{34} \end{bmatrix}. \tag{3.8}$$

The equation (3.7) can be solved by applying a *least squares method* as ([YH96])

$$\mathbf{k} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{t}, \qquad \left\| \frac{1}{\hat{\mathbf{k}}} \right\| = c, \tag{3.9}$$

Table 3.1: Estimated gunshot parameters corresponding to different observation combinations.

Observation	**Shooter direction**	**Bullet trajectory, speed, and caliber**	**Shooter location**
Muzzle blast	X		
Shock wave		X	
Muzzle blast and shock wave	X	X	X

where c stands for the speed of the arriving sound wave. In this thesis, the speed of sound in air is defined as $c = 342$ m/s. Note that the solution assumes *far-field* conditions [Ars99], meaning that the arriving wave is assumed as plane wave. Additional details and explanation related to TDOA and DOA estimation is provided in e.g. [Per09].

3.2 Gunshot Events and Geometry

The different sound sources occurring in a firing event and their corresponding geometric layout are considered in this section. Unless a silencer is used, an obvious sound source is the *muzzle blast* sound generated at the gun barrel. In addition, whenever the fired projectile exceeds the speed of sound in the air, an acoustic *shock wave* is formed, causing another very sharp and sudden wave form. In the case of sniper weapons, such as precision-rifles, a great majority of them fires supersonic bullets [Wor01]. A third potential sound source is the aimed target: depending on the material and position of the target, it may also cause audible sounds to the array.

In this work, the muzzle blast and shock wave signatures are applied in estimating the different gunshot parameters. As shown in Table 3.1, different parameters can be estimated based on the amount of observed signatures.

3.2.1 Muzzle Blast

The muzzle blast signal is produced by an explosive inside a cartridge. The explosive causes a sudden increase to the gas volume inside the gun barrel, forming thus high-pressured waves that send the projectile into flight and cause the actual muzzle blast. The muzzle blast signal is highly directional, producing the highest sound level in front of the gun [ISO05]. As is commonly known, geometrical spreading, atmospheric absorption and reflections attenuate the sound as a function of distance. These factors define a certain maximum distance from how

far the muzzle blast can still be heard.

3.2.2 Shock Wave

The shock wave signature has a well-recognizable shape, looking like a letter of 'N'. Hence it follows that the shock wave is also known as "N-wave". The wave is caused by *compressed* air molecules that bend around a supersonic bullet. This happens as the molecules have a certain maximum speed – corresponding to the speed of sound in the corresponding medium – such that they have no time to "move out" in front of the bullet. The phenomenon is illustrated in Figure 3.1, where a *cone*-shaped shock wave pattern is formed on both sides of a propagating bullet. The angle between the bullet trajectory vector, $\mathbf{h}$, and the shock wave front is called *Mach-angle*, θ_M, and it is proportional to the bullet speed v by

$$\theta_M = \sin^{-1}\left(\frac{1}{M}\right), \tag{3.10}$$

where M is a *Mach number*, defined as $M = v/c$ [Mah06].

3.2.3 Firing Event Geometry

Assume we have a microphone array mounted on a soldier helmet at known position. In order to derive the trajectory of an observed supersonic bullet, an estimate of a so-called *closest point of approach* (CPA), marked as $\mathbf{a}$, is needed. The CPA is the point in the bullet trajectory from which the distance to the center of the microphone array is the shortest among all the trajectory points. This is illustrated in Figure 3.2, where the firing event geometry is shown from above. The term $\mathbf{m}_n$ stands for the position of microphone n in the array, $\mathbf{g}$ is the position of the shooter, $\mathbf{h}$ is the *heading* of the bullet trajectory with respect to $\mathbf{g}$, and $\mathbf{s}_n$ is a so-called *S-point*, defining the position where the formed shock wave front originates to microphone $\mathbf{m}_n$. Due to the different microphone positions, both $\mathbf{a}_n$ and $\mathbf{s}_n$ are defined separately for each microphone $\mathbf{m}_n$ in the array. As the line drawn from $\mathbf{a}_n$ to $\mathbf{m}_n$ is perpendicular to the bullet trajectory, estimating $\mathbf{a}$ is enough to define the trajectory unambiguously.

3.3 Gunshot Detection and Recognition

In a real-world application, gunshot event detection and recognition plays a key role in the overall estimation process. A continuous estimation process would be impractical at least because of power consumption and an unnecessary computational workload (due to a vast amount of invalid estimation data received). Transient detection methods can be applied for the task, and in this thesis the gradient-based detection approach introduced in [Kla06, Chapter 4] is chosen

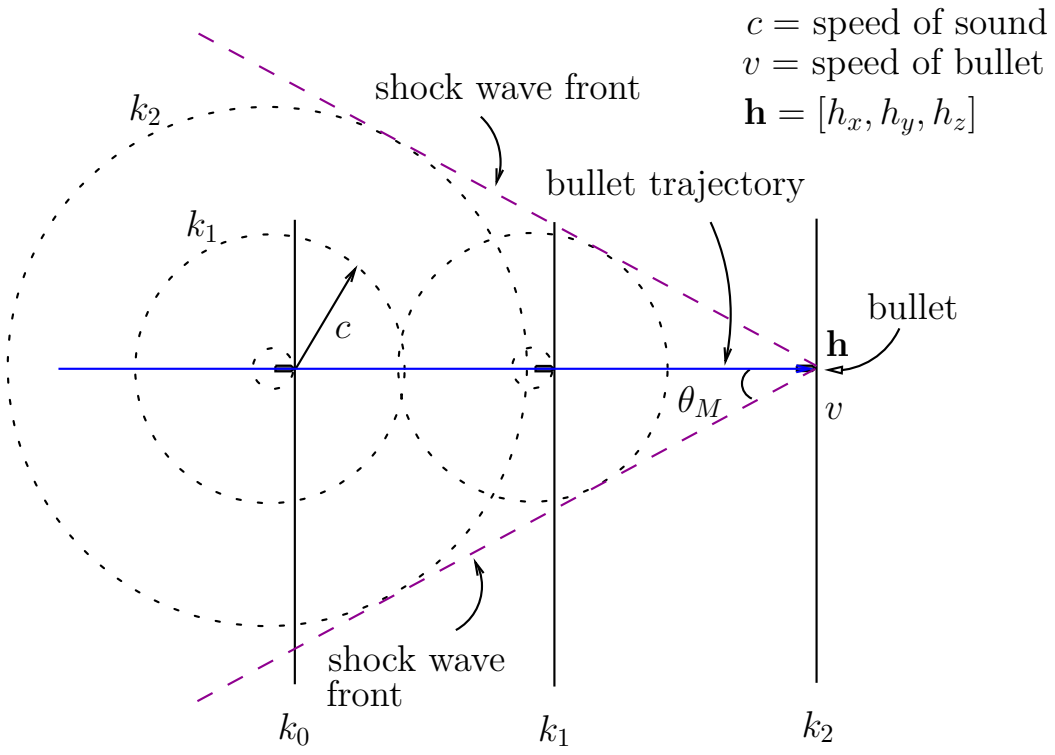

Figure 3.1: As a bullet propagates from left to right, a shock wave cone is formed behind the bullet, marked with a purple dashed line. The shock wave front is propagating at the speed of sound c, while the bullet has a speed v. Vector $\mathbf{h}$ determines the *heading* of the bullet trajectory, whereas the different time indices are marked by k_n, where n is running from 0 to 2.

among potential solutions, see e.g. [Sad98, Che01, Duf00]. The shock wave and muzzle blast recognition considered after the detection was first proposed in [P2].

3.3.1 Transient Detection

The detection process begins by forming a so-called *energy envelope* function from the input signal $x(k)$. This is done by summing the signal frequency components from a specific range over short time windows, stated as

$$E(k) = \sum_{f \epsilon \kappa} |\mathrm{STFT}(x(k), f)|^2, \tag{3.11}$$

where $\mathrm{STFT}(x(k), f)$ is a *short-time* Fourier transform of the signal $x(k)$ – windowed with a rectangular window – and f stands for the frequency index. Typically a window length of 20 ms with a $50 - 75\%$ overlap between consecutive windows is used. The frequency range κ can be specified according to the sound source at stake.

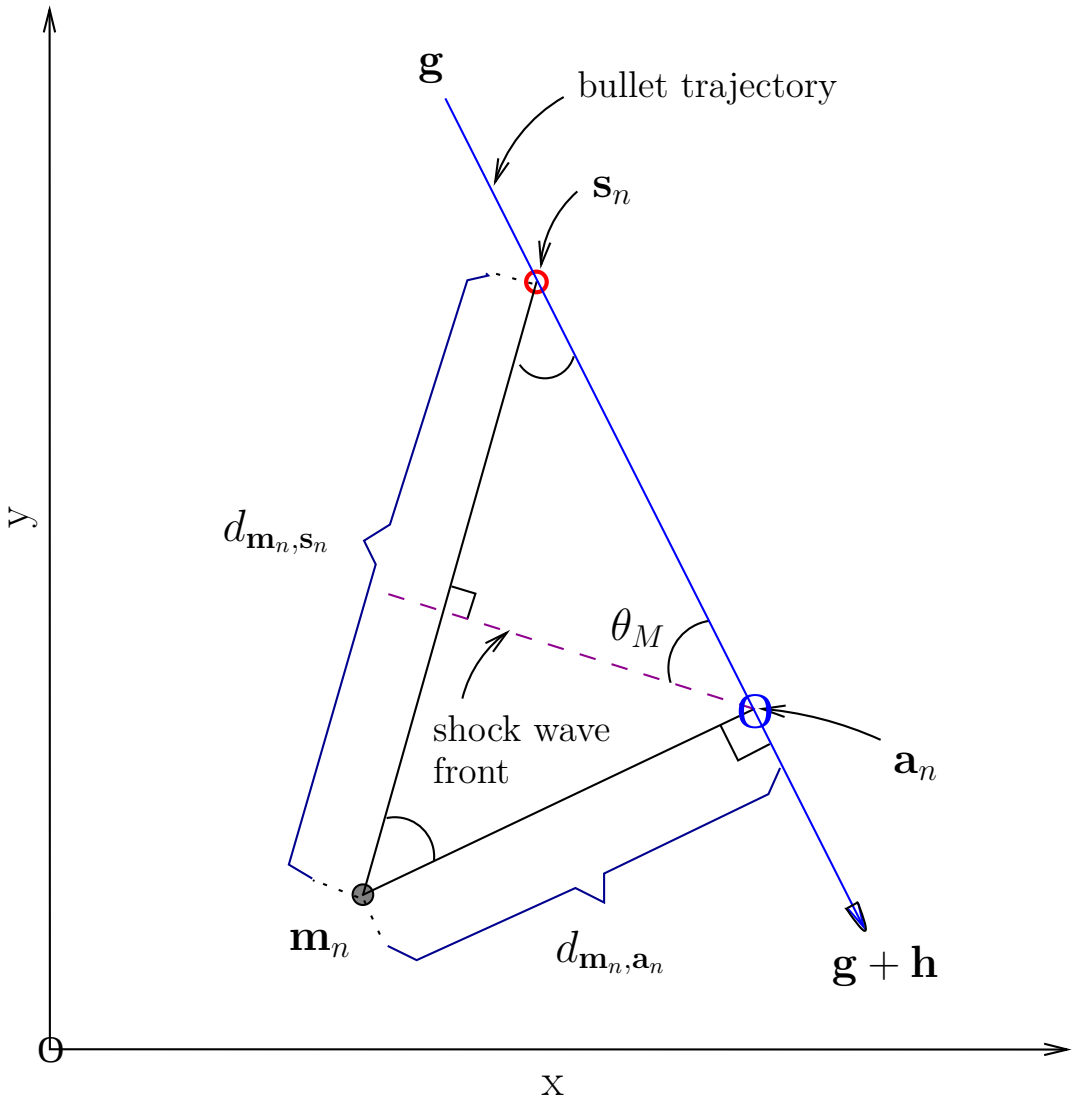

Figure 3.2: The geometry of a firing event viewed from above. The shock wave front is shown with respect to the CPA - point ($\mathbf{a}_n$) and S-point ($\mathbf{s}_n$).

The energy envelope function $E(k)$ is not as such optimal for detection. Instead, its temporal derivative, $D(k)$, can be used to obtain a more reliable detection performance. Using linear regression, $D(k)$ is defined as

$$D(k) = \frac{E(k+1) - E(k-1)}{3}. \tag{3.12}$$

For simplicity, here only the differences between data samples in the previous and next time frame (from the current time index k) are considered. Next, the biggest *peaks* of the detection function $D(k)$ are searched, as those correspond to the most significant energy level changes of $E(k)$. Note that, due to the need

of comparing against the $k+1$ time frame, the peak is actually detected with a delay of one frame step. However, such a delay ($\sim$ 10 ms) has no practical impact to the overall processing time.

Specific heuristics are applied in determining a proper *threshold* for finding the detection function peaks. One possibility is to take a local average of $E(k)$ over 1.5 s and multiply it with a suitable weight (as is done in [P2]). Considering a real-time application, the threshold needs to set through a calibration phase to adapt to the environment at stake, i.e., the background noise level. For this, adaptive thresholds have been proposed, as in [Che01].

3.3.2 Transient Classification

As discussed earlier, audio content-based classification is as such one of the main application areas considered in this thesis, but here it is included at some extend to the shooter localization scheme. To be more precise, a three-class audio (transient) classification problem is encountered after the transient detection phase. The classes are labelled as "muzzle blast", "shock wave" and "invalid detection", and the classification approach is based on a two-phase *unsupervised* (rule-based) classifier. Experimentally defined *temporal* and *spectral* thresholds are applied for the task, as shown in this section.

At first, the temporal threshold of 187.5 ms (corresponding to 9000 samples at sampling frequency $F_s = 48000$ kHz) is applied to see whether a new transient $S(T+1)$ – where S is the sample number and T stands for the transient index – occurs within the set temporal limit. The threshold is based on the fact that, whenever a shock wave is formed, the shock wave and muzzle blast wave fronts should occur temporally very close to each other. Next, the spectrum of the time frame in which a transient has been detected is analysed. By applying a spectral threshold of 1.6 kHz, the *relation* between the energy levels of $E(k)$ below and above the threshold is computed using (3.11). The motivation relies upon the fact that the shock wave is extremely sharp by nature, which results in high energy at the high frequencies of the spectrum. Finally, the obtained analysis results are combined and the classification output is obtained based on the logic described in Figure 3.3. For an interested reader, also other types of gunshot classification approaches have been proposed in the literature, such as the one in [Par12].

3.4 Bullet Parameter Estimation and Shooter Localization

The bullet trajectory, speed and caliber can be estimated based on mathematical modelling of shock waves. For this, the shock wave model and the overall estimation process applied in this work are introduced in this section. The section begins with a brief literature review to the sniper localization topic in general.

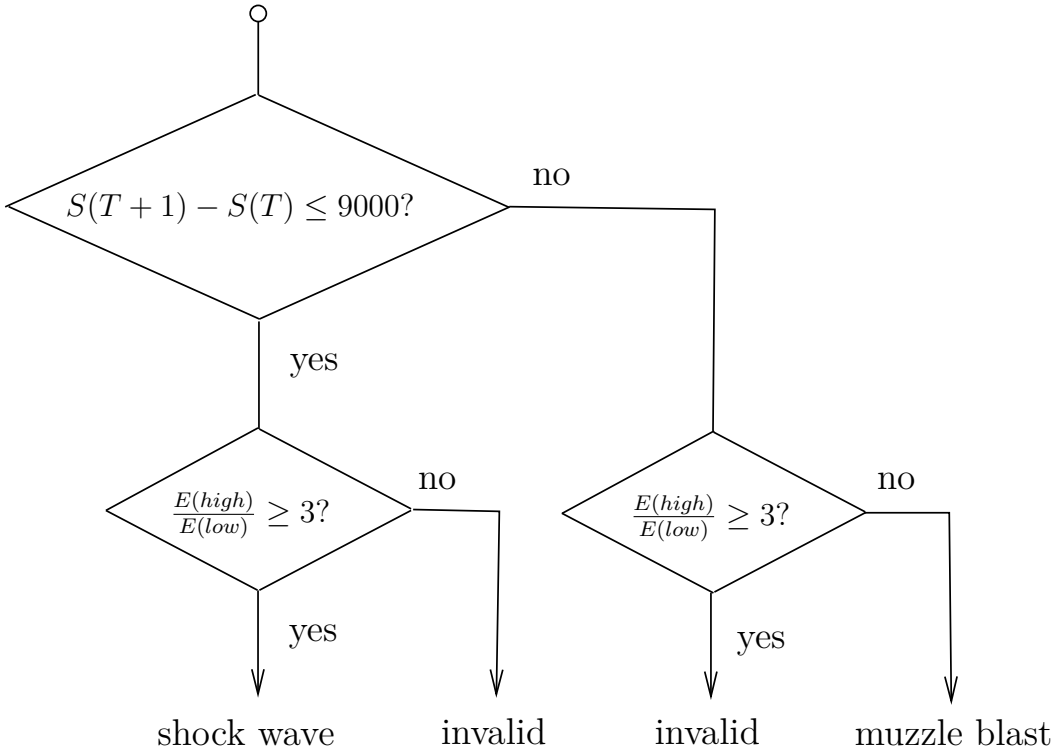

Figure 3.3: A flow chart of the transient classification algorithm. The classification of detected transients into shock wave and muzzle blast categories is based on two phases. The terms $E(low)$ and $E(high)$ correspond to the signal energies between $(0 - 1600)$ Hz and $(1600 - 16000)$ Hz, respectively, whereas $S(T)$ is the sample value of the Tth detected transient.

3.4.1 Preliminary Work

Due to its importance, several sniper localization approaches have been proposed in the literature [Duc97, Sto97, Vol07, Nas06, Hen07], on top of which at least few "counter sniper" systems with non-published technical details exist around the world [BBN09, SST09]. The mentioned articles consider both the shock wave and muzzle blast signatures in their estimation processes, whereas methods relying solely upon shock waves are discussed e.g. in [Dan06, McN93]. The approach in [Dan06] applies an iterative numerical solution to nine non-linear equations for the estimation, providing a successful approach for trajectory estimation. However, the method requires nine microphones (hence the number of equations) and – more importantly – it only works with *fixed*, ad hoc microphone distributions. This means that it cannot be considered as a soldier-wearable solution. Instead, in [McN93] a unit vector-based method with three sensors (each consisting of three separate transducers for generating signals in response to an obtained shock wave) is considered. This approach can estimate the bullet trajectory with wearable sensors, but the sensors themselves are unique and not easily available.

A major challenge regarding to shooter localization and bullet trajectory estimation relates to generalizing the estimation over several different weapon types and environments. The approach presented in this thesis applies the stochastic optimization algorithms discussed in Chapter 2 – namely particle filtering and simulated annealing – to address this challenge. Such stochastic processes provide means to discover alternative, possibly non-linear, estimation solutions compared to conventional and somewhat restricted deterministic approaches. Stochastic approaches were first studied for this topic in [Léd05, Bar08], both in which genetic algorithms are considered in the estimation. However, as in [Dan06], the approach presented in [Léd05] is designed for ad hoc sensor distributions over a sufficiently large area. In [Bar08], an array of acoustic sensors (microphones) is considered, whereas the number of sensors required for the approach is at least five, preferably seven. The method proposes combining the muzzle blast and shock wave observations for estimating the shooter distance to the sensors, from which a similar principal idea is also adopted to the publication [P2] of this thesis.

3.4.2 Mathematical Shock Wave Modelling

The mathematical shock wave model applied to this work was first introduced by Whitman in [Whi52]. The model simulates the time-domain shock wave waveform as a function of projectile *diameter* ϕ, velocity v, and the *miss distance* $d_{\mathbf{m}_n,\mathbf{a}_n}$ (see Figure 3.2). As defined by the model, the atmospheric signal peak level A_n and length L_n observed at microphone $\mathbf{m}_n$ are given as

$$A_n = \frac{0.53P_0(M^2-1)^{1/8}\phi}{d^{3/4}_{\mathbf{m}_n,\mathbf{a}_n} l^{1/4}} \ \text{[Pa]}, \tag{3.13}$$

$$L_n = \frac{1.82Md^{1/4}_{\mathbf{m}_n,\mathbf{a}_n}\phi}{c(M^2-1)^{3/8} l^{1/4}} \ \text{[s]}, \tag{3.14}$$

where P_0 is the atmospheric air pressure, M is the Mach number defined in section 3.2.2, and l is the length of the bullet, related to its diameter by: [Fer07]

$$l \approx 4.35\phi \ \text{[m]}. \tag{3.15}$$

Also the shock wave *time of arrival (TOA)*, τ_n, to a given microphone $\mathbf{m}_n$ is needed for modelling. The TOA is measured from the S-point $\mathbf{s}_n$ to the microphone position, and it is defined as

$$\tau_n = \frac{||\mathbf{m}_n - \mathbf{s}_n||}{c} = \frac{d_{\mathbf{m}_n,\mathbf{s}_n}}{c} \ \text{[s]}, \tag{3.16}$$

where $d_{\mathbf{m}_n,\mathbf{s}_n}$ is the distance between the microphone $\mathbf{m}_n$ and the S-point $\mathbf{s}_n$.

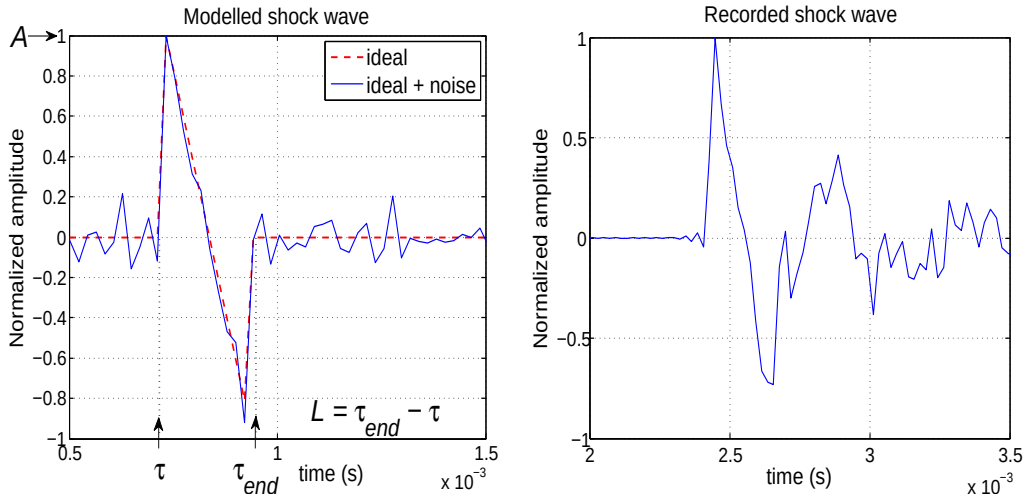

Figure 3.4: An example of a modelled and recorded shock wave signatures together with the model parameters, τ, A and L. Red dashed line on the left image represents an ideal shock wave form with zero noise level.

Finally, the shock wave time domain waveform can be generated as [Sad98]

$$y_n(k) = \begin{cases} A_n \left(1 - 2^{\frac{k-\tau_n}{L_n}}\right), & \tau_n \leq k \leq \tau_n + L_n \\ 0, & \text{otherwise,} \end{cases} \tag{3.17}$$

where k stands for a discrete time index. The equations (3.13) and (3.14) reveal that the shock wave amplitude A_n increases and the wave length L_n becomes shorter as the miss distance decreases. To provide an example of a regular shock wave form, Figure 3.4 shows both a modelled and a (normalized) *recorded* shock wave together with all the modelling parameters. The first rising edge of "N-wave" results from the highly compressed air in front of the projectile, whereas the second rise at the end is formed as the air molecules fill up the generated void again. Note that the recorded signal (right-hand side) shows some oscillations which do not exist in the simulated model (left-hand side), most probably caused by ground reflections. Also note that a real world shock wave might oscillate somewhat even without reverberation, as can be noticed e.g. from the figures shown in [Mah06]. However, despite such non-ideal ripple, the shock wave model is able to mimic the most essential parts of the real signal.

3.4.3 Spatial Likelihood Function

A probabilistic *inference* approach is considered in this work to estimate the interested bullet state parameters, i.e. the trajectory, speed and caliber. For this, a spatial *likelihood function* needs to be constructed. The function provides as an output the likelihood of observing a specific shock wave form, given the bullet

state parameters. In practice, the likelihood value is computed by *comparing* the shock waves actually observed by the microphones $\mathbf{m}_n$, $n = \{1, \ldots, 4\}$, to those modelled by randomized parameters. As discussed above, the modelled shock waves are dependent on the varying input parameters $\mathbf{a}$, ϕ and v.

The shock wave comparison can be made in several ways, as long as the computation remains feasible. The computation may become a problem, since the comparison needs to be performed separately for all the microphone channels. This thesis considers both time and frequency domain comparison criteria: mean squared error (MSE) and the GCC-PHAT discussed in section 3.1.1. Because of the Fourier transform needed with GCC-PHAT at each comparison, computationally the MSE is somewhat more attractive, whereas it lacks in accuracy in some cases as shown in publication [P2].

Applying the MSE criterion, the likelihood is computed as

$$L(\mathbf{s}[t]|\mathbf{O})_{\mathrm{MSE}} = \exp\left(-\frac{1}{\sigma}\sum_{n=1}^{M}(\mathbf{x}_n - \mathbf{O}_n)^2\right), \tag{3.18}$$

where $\mathbf{x}_n$ is a vector containing the modelled time domain shock wave signal of length N to microphone $\mathbf{m}_n$, σ is a parameter defining the *variance* of the resulting likelihood distribution, $\mathbf{O}$ is a $M \times N$ matrix containing the actual observed signals of length N from M different microphone channels, and $\mathbf{s}[t]$ defines the bullet state vector at iteration t, i.e.,

$$\mathbf{s}[t] = \left[\mathbf{a}^T,\ \phi,\ v\right]^T. \tag{3.19}$$

In short, (3.18) states that the smaller the difference between the observed and the modelled time domain waveforms, the higher the likelihood value. The GCC-based comparison, instead, is performed by computing the PHAT-weighted GCC between the Fourier-transforms of the observed and modelled shock waves (using (3.2)). The likelihood value is then obtained by

$$L(\mathbf{s}[t]|\mathbf{O})_{\mathrm{GCC}} = \max_k \left(\prod_{n=1}^{M} C_n(k)\right),\ k \in (-\tau_{max}, \tau_{max}). \tag{3.20}$$

As can be seen, the obtained cross-correlation functions C_n from each channel n are multiplied together, such that whenever the observed and modelled signals are similar in content, high values should result with the correct lag k.

An illustrative example of a spatial likelihood function obtained with the MSE criterion is shown in Figure 3.5. Real recorded gunshot data from an outdoor shooting range is used in computing the function. The recorded data are utilized in publications [P1] and [P2], whereas more descriptive details of the actual data recordings are provided in [Mäk08]. As the state vector $\mathbf{s}[t]$ to be optimized is actually 5-dimensional - consisting of the Cartesian $\mathbf{a}$ coordinates, ϕ and v - for

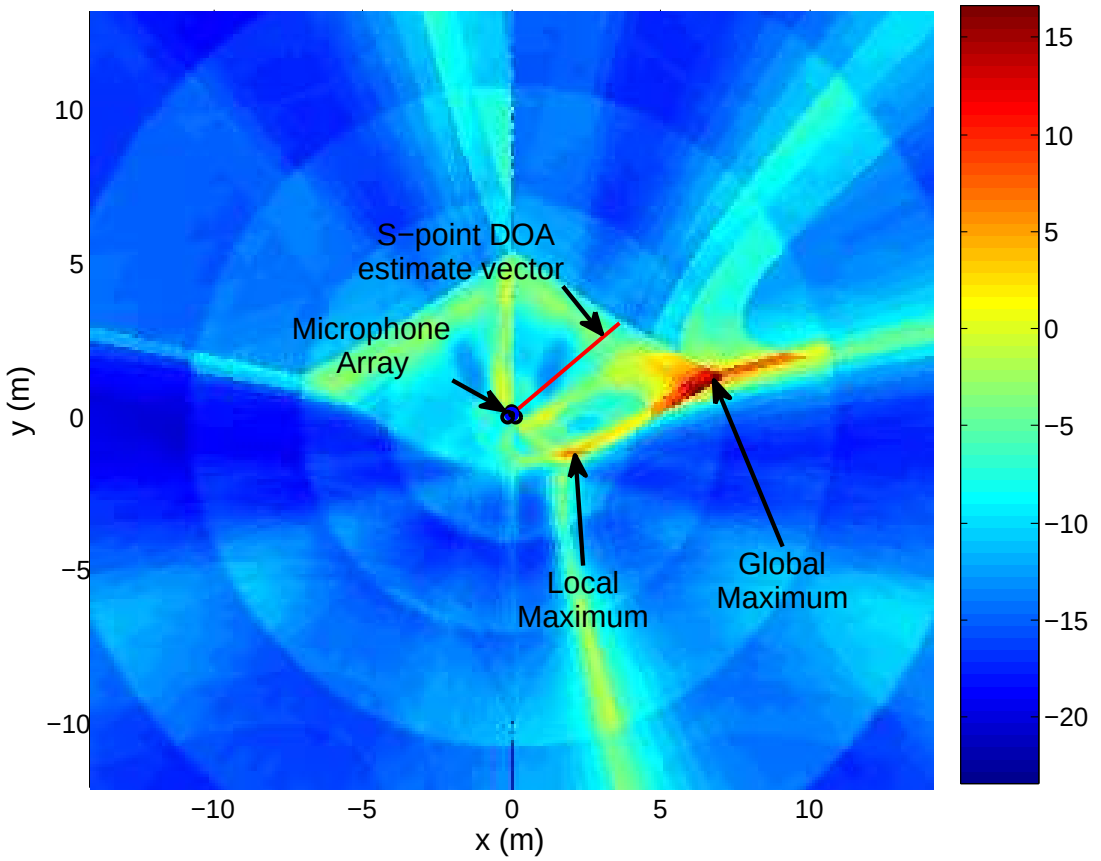

Figure 3.5: An illustration of the likelihood function (Eq. (3.18)) with respect to x- and y-coordinates of the CPA point. The parameters a_z, ϕ and v are fixed for illustration purposes. Four microphones are used in the microphone array located at the origin.

illustration purposes ϕ, v and a_z (the z-coordinate of $\mathbf{a}$) are *fixed* a priori in the figure. Thus, the illustrated likelihood values correspond to the variations of the $\mathbf{a}$ x- and y-coordinates a_x and a_y.

The blue areas of Figure 3.5 correspond to the low-probability spatial areas of $\mathbf{a}$, whereas the dark red spot is the global maximum of the likelihood function, i.e. the solution to which the applied stochastic optimization algorithm should converge. Microphone array is located at the origin, and the solid red line originating from it illustrates the DOA estimate of the actually observed shock wave front. Thus, the geometry is rather similar to that shown earlier in Figure 3.2. The negative likelihood values follow from the fact that *logarithmic* scale is used in the figure. Clearly, the shown likelihood distribution contains "clutter" and several local maxima, severely restricting the usage of traditional gradient-based optimization methods. Multi-modal and complex search spaces in general raise needs for applying stochastic optimization algorithms for such

optimization problems.

3.4.4 Discussion

Estimating the bullet parameters and the shooter position requires solving a non-linear inverse problem by optimizing the parameters of the shown shock wave model with respect to a proper fitness criterion. The both criteria considered in this thesis (and the estimation approach itself) are proven to provide successful estimation results in publications [P1] and [P2]. The achieved caliber estimation results are in well comparison with the results provided earlier in the literature, although no universal criterion for a "correct" caliber classification is defined. The estimates of the bullet speed can be considered as satisfyingly accurate, as the estimation error of 10% reported in [Sto97] is well met also in the results of this work. In general, the speed estimation errors less than this have no appreciable effect on the trajectory estimation, as those have no major effect to the Mach-angle θ_M.

The results regarding to the shooter position show that the estimation can be performed with an accuracy of few percents of the actual shooting distance. As demonstrated in [P2], in overall the estimated bullet trajectories and shooter positions are close to the actual ones. The achieved detection and classification rates for both the muzzle blast and shock wave signals are above 80% in all the cases experimented in [P2]. In most of the cases, the shock waves are detected slightly better than the muzzle blast signals. This can been seen as a positive result, as the shock wave detection and classification has an essential role in the overall estimation procedure.

Chapter 4

Content-based Audio Management

Following the dramatic increase on the computational capacity of microprocessors and the size of storage spaces, content-based multimedia analysis has become a widely studied topic. Several such methods and solutions that could barely be dreamt of only few decades ago are now available and used around the world. Completely new types of scientific fields have emerged due to the development, such as *data mining* [Han11], *machine learning* [Alp10] and *pattern recognition* [Dud01], which are contiguously concentrated in this chapter. To be more precise, here the main focus is put on audio content-based *classification* via stochastic classifier parameter optimization. Thus, the chapter discusses the background and reasoning of the research performed in publications [P3] and [P4].

4.1 Machine Learning Principles

Machine learning as a term can be roughly defined as a *prediction* of some output quantity (such as an audio class label) based on *learned* properties of the (training) data seen so far by a learning algorithm. A machine is said to be learning whenever it changes its structure, program, or data (based on its inputs or in response to external information) in such a manner that the expected future performance of the machine improves [Nil98, page 1]. The process consists of estimating the unknown parameters of a model – or a *classifier* – based on the available training patterns. In practise, the "learning" phase refers to some form of algorithm for reducing the error in predicting the labels of a training data set [Dud01]. Several application areas, such as regression and classification, can be performed using machine learning methods, whereas in the scope of this work the main focus is kept on content-based classification and retrieval.

In general, the learning process can be divided into two categories: supervised and unsupervised learning. In supervised learning, a class label is provided for each pattern belonging to the training set, meaning that the aim is to learn a

mapping from the input to the output. In unsupervised learning, there is no such "supervisor" and we only have the input data with no labels [Alp10, page 11].

4.1.1 Supervised Learning

In the sense of classification, supervised learning can be understood as learning a class from examples belonging to that class. For this, several classifier models have been proposed in the literature, among which the ones considered in this work are introduced in this section.

Artificial neural networks

Inspired by the observations made in the study of biological systems, such as human brains, *artificial neural networks* (ANNs) [Gur97] seek to loosely mimic the actual biology in mapping an input space to an output space. In the context of classification, the input space corresponds to extracted *features* (also called as *attributes*) from the data, whereas the output space corresponds to the class labels of the data. An ANN consists of several artificial neurons, which are basically mathematical computing engines providing a single output from several *weighted* inputs. This can be stated mathematically as

$$y = f_a \left(\sum_{i=0}^{n} w_i x_i + \theta \right), \tag{4.1}$$

where n is the number of inputs x_i, w is the the weight coefficient, θ stands for an external *bias* value added to the neuron, and f_a denotes an *activation* function, for which sigmoid or logistic function is commonly applied [Pri05]. Depending on the number of neuron *layers* in the network, the inputs x_i may be either the original ANN inputs or outputs of the neurons from a previous layer. This is clarified in Figure 4.1, where a principal structure of a multilayer feed-forward neural network, also known as *multilayer perceptron* (MLP), is shown.

Training of ANNs – i.e. optimizing the parameters w_i and θ – is performed by providing a desired output (the ground truth class label) for a given input vector. A learning algorithm is then applied to minimize the error (usually MSE) between the network output and the ground truth class label. This is traditionally done using the well-known *back-propagation* (BP) [Cha95] algorithm.

Support vector machines

Also known as kernel machines, the *support vector machines* (SVMs) have become widely used in machine learning tasks. By definition, SVMs are binary classifiers, in which a *hyperplane* is searched that separates two classes in a feature space. The training algorithm strives for maximizing the distance to the nearest samples of both classes from the hyperplane. These nearest samples are called "support

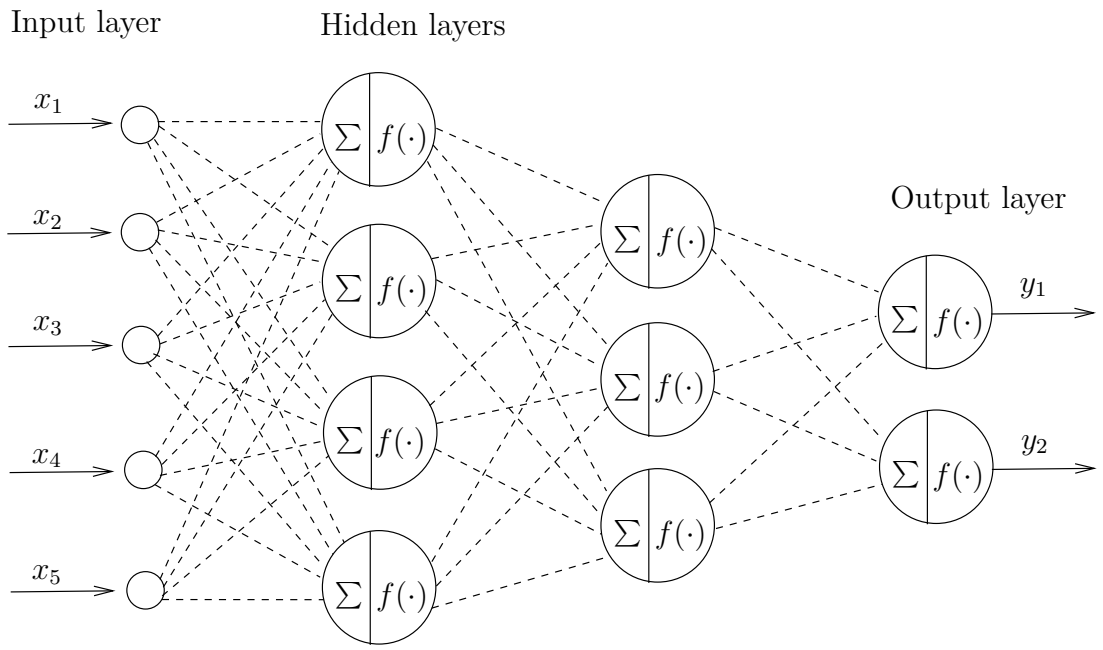

Figure 4.1: Principal structure of a feed-forward artificial neural network with 5 input neurons, 2 output neurons, and 4 layers. The two middle layers are said to be "hidden".

vectors"; hence the name support vector machine. The hyperplane shape is dependent on the underlying kernel function, for which linear, polynomial, sigmoid, or radial basis functions are typically applied. The most suitable kernel function depends on the case: for example, linear kernel is not suitable for a non-linear classification problem, where a transformation to a higher feature space dimension (i.e., the kernel trick) should be applied instead. [Wan05]

In a multi-class classification problem, multiple SVM binary outputs need to be combined. Two common methods exist for this, namely *one-against-all* (distinguish one class against the rest) and *one-against-one* (distinguish between every pair of classes). In the former the classification of new instances is done based on a "winner-takes-all" strategy, in which the binary classifier with the highest output score assigns the class. In the latter case, classification is made based on *voting*, in which every classifier assigns the instance to one of its two classes, giving hence a vote for that class. At the end, the classified instance is labelled with the class gaining most votes. [Hsu02]

Random forests

An *ensemble classifier* [Pol06] called *random forest* (RF) consists of several decision tree structures. Decision tree is a graphical predictive model for performing the attribute/feature mapping discussed above. The tree structure consists of *leaves* and *branches*, representing the class labels and the conjunctions of spe-

cific characteristics which lead to these classes. In practise this means that each branch possesses a certain criterion for some extracted characteristic of the input sample. If the criterion is met, the classification process will proceed to that branch, whereas otherwise some other branch (in which the criterion is fulfilled) is selected. At the end of the last branch of the tree, there is a leaf stating the class label for the input sample at stake. Finally, the *mode* of the classification results of each decision tree in the forest is taken as the actual output class of the RF classifier.

In general, ensemble classifiers are based on an idea of not training a single classifier, but a *set* of classifiers (hence the name "ensemble"). The outputs of these several classifiers are then combined to form the overall output of the ensemble [Pol06]. The discussed random forest classifier (ensemble of trees) is applied for comparison in publication [P4], and a more detailed mathematical description of random forests is provided in [Pav00].

4.1.2 Unsupervised Learning

In unsupervised learning, the input data contains no class labels or other additional knowledge about the data content, meaning that there is no explicit supervisor available for learning. The goal of unsupervised learning is hence to extract an efficient internal (and statistical) representation of the input data structure. In this thesis, unsupervised learning is only considered in the form of data clustering and *k-means* [Dud01, Chapter 10] classification (in publications [P5] and [P6]). In general, clustering describes a process where extracted data features are assigned to compact and distinct groups (clusters). The k-means algorithm, instead, classifies any extracted feature to its *nearest* cluster in the feature space, based on the cluster *centroids* (mean values). Unsupervised learning is discussed more e.g. in [Hin99].

4.2 Content-based Audio Classification

This section concentrates on describing the audio feature extraction process and the evaluation metrics required for content-based audio classification. At first, some preliminary work of the field is discussed.

4.2.1 Preliminary Work

The content-based audio classification and retrieval research made by Wold et al. [Wol96] is widely acknowledged as the first of its kind in the field. The article provides means for utilizing pitch, loudness, brightness and bandwidth audio features in classifying the used "Muscle Fish" audio database. Statistical measures of the features (mean and covariance) are used to form a model of each class, and

the classification is performed by computing the distance in feature space between the trained models and the features of a signal to be classified. Later, Li [Li00] applied a nearest feature line (NFL) method for the same database. The approach is based on multiple prototypes obtained from each class, which are separated by the feature lines. The distances between distinct feature points are obtained using projection techniques between a classified point and the constructed feature lines. Since these preliminary works, also SVMs have been applied to the Muscle Fish database with promising results [Guo03].

The SVM classifiers were also applied in [Che06, Zhu07] for classifying audio segments. The achieved results favoured SVMs over the other experimented classifiers (k-nearest neighbour, artificial neural networks and naive Bayes). An alternative approach was proposed by Ravelli et al. [Rav10], where a new audio signal representation was proposed. This transform-domain approach considers three separate applications, e.g. music genre classification. The proposed novel audio codec allows extracting modified audio features, which can be used for receiving more information about the audio content. Mitra and Wang [Mit08] applied a *parallel* ANN architecture for music genre classification. In their approach, extracted feature vectors are fed to the separate branches of the parallel ANN architecture, and the final classification result is obtained by vector-summing the individual branches. Moreover, perceptual aspects of the human auditory system were modelled in [Har07] for a similar classification problem. In this approach, ANNs were concluded as the most convenient among the tested classifiers for performing the classification.

In [Dog09], SVMs were applied for classification along with *hidden Markov models* (HMM) [Rab93]. In this approach, a unique HMM is trained for each audio class (six considered) using the features of the MPEG-7 [Man02] standard. In addition to HMMs, also *Gaussian mixture models* (GMMs) are commonly used with audio classification, see e.g. [Tza02]. In both the HMM- and GMM-based approaches, the idea is to estimate a probability density function for the feature vectors of each predefined audio class. The actual classification is then performed using statistical pattern recognition (SPR) classifiers, such as the GMM classifier applied in [Tza02]. Peeters [Pee07] used GMMs together with HMMs for modelling individual audio classes for music genre classification. The classification is performed using statistical models, where principal component analysis (PCA) [Dun89] and linear discriminant analysis (LDA) [McL04] are also applied to lower the feature space dimensionality. However, as a general problem in content-based audio classification, also in [Pee07] the selection of the best classifier configuration remains still unsolved. Note that, in audio retrieval problems, which are closely related to classification, GMM and HMM modelling are also commonly used [Wic10], [Hel10].

More recently, audio classification has been broadened to more specific problems, such as context recognition [Hei10] [Ero09]. Especially with such challenging classification tasks, it should be noted that the performance of the approaches

varies considerably depending on the extracted features and the classifier parameters. This is a widely acknowledged challenge in the field, see e.g. [Alp10]. As a reference for an interested reader, several classifier types and the effects of their configurations are studied in [Liu04]. A selection of different audio features is introduced in the next section, and some of their effects are brought into discussion later in Chapters 5 and 6.

4.2.2 Audio Feature Extraction

The feature extraction process has a crucial role in the overall learning process: whenever the extracted features are incapable of discriminating audio classes from each other, no classifier can achieve a satisfactory classification performance. Audio features consist of both time and frequency domain attributes, designed to characterize the signal shape and frequency content. Most of the audio features applied in this thesis are described and discussed in this section. As a common convention in audio signal processing, in this work all the features are extracted within short time frames of 40 ms. The features are extracted using the *MUVIS* framework for content-based indexing and retrieval in multimedia databases [Kir12], developed at Tampere University of Technology. In addition, a *LibXtract* [Bul07] audio feature extraction library is utilized in the extraction.

Mel-frequency cepstral coefficients

Especially in speech recognition, Mel-frequency cepstral coefficients (MFCCs) – and their derivatives – are widely applied [Qua08]. The extraction of the MFCCs is based on a perceptually motivated frequency scale called *Mel-scale*, which is defined as

$$Mel(f) = 2595 \log_{10}(1 + \frac{f}{700}), \tag{4.2}$$

where f is frequency in Hz [Qua08]. The Mel-scale is approximately linear up to 1000 Hz, after which the scale turns logarithmic. At first, the input signal is transferred to frequency domain using a discrete Fourier transform (DFT), after which its *power spectrum* is computed by squaring the DFT absolute values. Next, a filter bank of triangular filters, distributed according to the Mel-scale, is constructed and multiplied with the obtained power spectrum. A principal illustration of a such Mel-scale filter bank is provided in Figure 4.2, where it is shown that the filter magnitudes are set to unity. Spectral energy values of the Mel bands are then obtained by *summing* the resulting spectrum over each band. These energy levels are still compressed by taking a logarithm of the band energy levels. Finally, the actual cepstral coefficients are computed using a discrete cosine transform (DCT) over the obtained log filter bank energies.

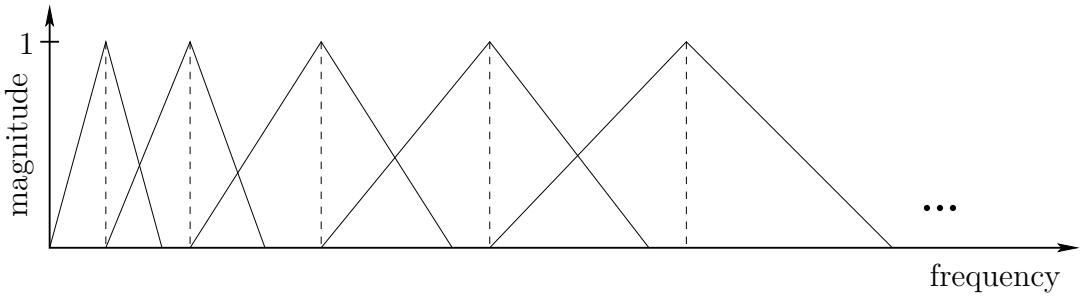

Figure 4.2: A principal illustration of a Mel-scale filter bank with respect to frequency.

In general, the lower cepstral coefficients describe the overall spectral shape, whereas its fine structure is included in the higher coefficients. The spectral dynamics may be characterized using *delta* cepstral coefficients, i.e. the temporal derivatives of the MFCCs. Usually these are obtained based on the coefficient trajectories over finite time segments, for example as: [Ye04]

$$\Delta m_j(k) = \frac{\sum_{n=-F}^{F} n m_j(k-n)}{\sum_{n=-F}^{F} n^2}, \tag{4.3}$$

where $m_j(k)$ denotes a cepstral coefficient of index $j \in \{0, \ldots, M\}$ at time k, F is the number of frames considered in the time segment before and after the current time index k, and M stands for the MFCC order. The acceleration coefficients $\Delta\Delta m_j(k)$ are obtained by applying (4.3) to the $\Delta m_j(k)$ coefficients.

Linear prediction

The audio spectrum can be also approximated by linear prediction (LP) analysis [Rab07, Chapter 6], where the spectrum is modelled by an all-pole filter concentrating on the spectral peaks. The approach is especially suitable for speech signals (for formant estimation), but may be also applied to other signals such as music [Sch77].

Predicting the next time-domain output sample of a linear discrete-time system can be performed by forming a linear combination of p previous output samples as

$$\hat{y}(k) = \sum_{n=1}^{p} a_n y(k-n), \tag{4.4}$$

where a_n are the linear prediction coefficients. These correspond to the poles of the predictor all-pole filter, for which the transfer function is defined as

$$H(z) = \frac{1}{A(z)} = \frac{1}{1 - \sum_{n=1}^{p} a_n z^{-n}}. \tag{4.5}$$

The prediction coefficients $\{a_1, a_2, \ldots, a_p\}$ can be solved using a *Levinson-Durbin* recursion [Rab07, Chapter 6]. The approach includes computing the signal *auto-correlation* coefficients $r(k), k \in \{1, \ldots p\}$, where p is the LP model order, such that the estimated linear prediction coefficients (LPCs), $\mathbf{a}$, satisfy the following matrix-form equation:

$$\mathbf{Ra} = \mathbf{y}, \tag{4.6}$$

where $\mathbf{R}$ corresponds to a $p \times p$ Toeplitz autocorrelation matrix and $\mathbf{y}$ is the output sample vector. The linear prediction cepstral coefficients (LPCCs) can be derived from the LPCs as

$$c_n = -a_n - \frac{1}{n}\sum_{k=1}^{n-1} k c_k a_{n-k}, \quad \text{for } n > 0, \tag{4.7}$$

where $a_0 = 1$ and $a_k = 0$ for $k > p$.

Spectral centroid

Spectral centroid (SC) represents the spectrum "mass" center point. It correlates with the subjective measures of "brightness" and "sharpness" also presented in the literature. Denoting $X(b)$ as the bth DFT frequency bin of an input signal $x(k)$, the SC is defined as

$$SC = \frac{\sum_{b=0}^{B} b|X(b)|}{\sum_{b=0}^{B} |X(b)|}, \tag{4.8}$$

where B is the index of the highest DFT frequency bin.

Spectral spread

The variance of the spectral centroid feature is called spectral spread (SS), and it is defined as

$$SS = \frac{\sum_{b=0}^{B} (b - SC)^2 |X(b)|}{\sum_{b=0}^{B} |X(b)|}. \tag{4.9}$$

This feature can be used for separating tone-like signals from noise signals, as it measures how concentrated the energy is around the spectrum centroid.

Spectral flux

Spectral flux (SF) measures the spectral variation by comparing two consecutive time frames. It is defined as

$$SF = \sum_{b=0}^{B} ||X_k(b)| - |X_{k-1}(b)||, \tag{4.10}$$

where the notation X_k represent the DFT output of the kth time frame.

Spectral roll-off

Spectral roll-off (SR) is a feature used for obtaining the spectral "skewness". The feature represents the frequency below which a specified amount of the spectral energy resides, and it is computed as

$$SR = \underset{f}{\text{argmax}} \left(\sum_{b=0}^{f} |X(b)|^2 \leq TH \sum_{b=0}^{B} |X(b)^2| \right), \tag{4.11}$$

where TH is a threshold value set between 0 and 1.

Short-time average frame energy

The short average energy (SAE) is a simple feature computed as a sum of squared amplitudes within a single frame. The feature is useful in detecting potential silent sections occurring in an audio signal.

Subband energy

A subband energy (BE) of nth spectral band is computed as

$$BE = \frac{\sum_{b \in B_n} |X(b)|^2}{\sum_{b=0}^{B} |X(b)|^2}, \tag{4.12}$$

where B_n denotes the set of frequency bins belonging to the nth frequency band. The whole frequency range can be divided into as many subbands as required to a particular application. For example, in publications [P3] and [P4] *four* bands are considered. It should be noted that in addition to energy calculation, also several other subband features may be extracted, such as the band *means* as realized in publications [P5] and [P6].

Band energy ratio

The ratio of the energies of two frequency bands, the band energy ratio (BER), can be computed as

$$BER(b_c) = \frac{\sum_{b=0}^{b_c} |X(b)|^2}{\sum_{b=b_c}^{B} |X(b)|^2}, \tag{4.13}$$

where b_c is a *cut-off* frequency bin from which the two bands are separated. The two subbands fully cover the whole range of the input signal spectrum.

Bandwidth

The effective bandwidth (BW) of an audio signal spectrum (or its subband) can be computed as

$$BW = \sqrt{\frac{\sum_{b=0}^{B} (b - SC)^2 |X(b)|^2}{\sum_{b=0}^{B} |X(b)|^2}}. \tag{4.14}$$

Zero-crossing rate

As the name suggests, zero-crossing rate (ZCR) defines the number of zero voltage crossings within a specified time interval, i.e., within a single time frame or a segment of several frames.

4.2.3 Feature Post-processing

Once the aforementioned audio features are extracted frame-wise from an input audio signal, in this work the extracted features are further post-processed in two optional ways. The post-processing aims to reduce the overall amount of features (which is huge for large databases) by means of clustering and statistical analysis. The first approach is called *key-frame*-based feature vector (FV) formation, in which a *minimum spanning tree* (MST) [Mot95] clustering algorithm is applied. The other one is based on audio *segmentation*, meaning that the extracted audio features are *averaged* over segments of several frames. The both approaches are described in detail below.

Key-frame feature vector representation

Key-frame-based audio FV formation takes advantage of the *redundancy* between different time frames in case of similar types of audio instances. Such instances include repeated vowels in speech signals or identical notes played by specific instruments in music pieces. The approach originates from video analysis, whereas it has been also used with audio signals already in [Kir06a]. The basic idea is

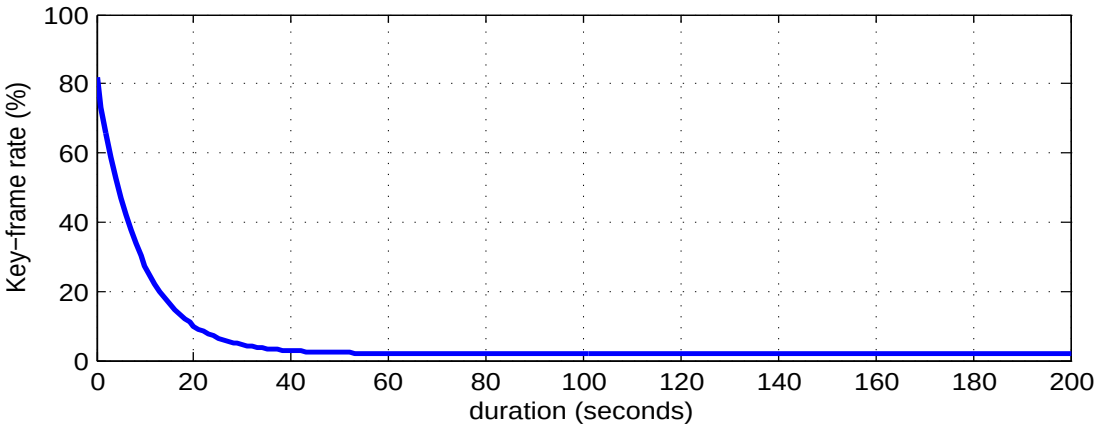

Figure 4.3: The applied key-frame selection rate in publications [P3] – [P6] for each cluster obtained with the MST clustering approach. As can be seen, the amount of selected fey frames decreases quickly as a function of overall signal duration to keep the overall amount of features feasible.

to cluster the audio frames based on their similarity between each other (corresponding to *distance* in the feature space). Once the clusters are formed, only one or few key-frames are selected from each cluster to represent the others with similar content. In publications [P3] – [P6], an experimentally determined key frame selection *rate* is applied, which is drawn as a function of signal length in Figure 4.3. A detailed usage of the MST clustering algorithm for the task is given in [Kir06a].

Audio segment feature vector representation

The second FV formation approach applies the segmentation method proposed in [Kir06b]. At first, the energy levels of each frame of the input signal are computed and compared to the average energy level of the whole signal. This is done to detect the silent sections from the signal. Next, the found non-silent consecutive frames are merged, resulting to distinct audio segments separated by the silent frames. Hence, theoretically, an audio signal with no silent sections would be considered as a single long segment. In practise, occasional non-silent frames may also occur in the middle of a silent section due to noise etc., which is why in this work an empirically determined threshold of *five* frames is applied as a minimum duration for a segment. Finally, the actual segment features are obtained by computing the *mean* (μ) and *standard deviation* (σ) statistics of each originally extracted feature over the formed segments. These segment features are ordered in a suitable way to form a feature vector, and the number of these new segment feature vectors (SFVs) corresponds to the number of formed segments, S. For clarity, the overall procedure is illustrated in Figure 4.4.

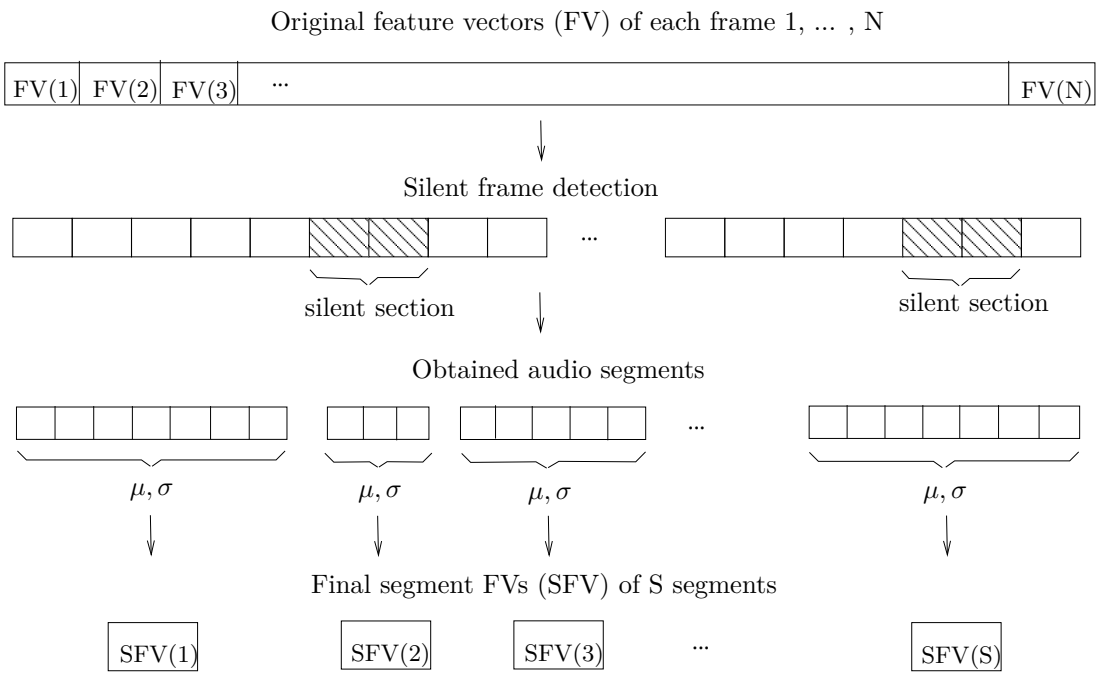

Figure 4.4: An illustration of the SFV formation.

As a common notice, statistical modelling of the extracted feature *distributions* is also a widely applied approach for feature representation. The HMM and GMM algorithms mentioned in section 4.2.1 are two commonly used examples of such approaches, see [Wic10], [Hel10]

4.2.4 Classification Evaluation Metrics

In order to evaluate the classification performance of a particular classifier, a new set of previously unseen data samples are classified with it. These new data samples are commonly referred as a *test set*, and the obtained classification results are compared to the ground truth class labels. The resulting classification error (CE) can be then computed as

$$CE = \frac{fp + fn}{N} \times 100, \tag{4.15}$$

where fp and fn stand for the number of "false positive" and "false negative" classifications and N is the total amount of tested samples. Other generally applied metrics for classification evaluation are *precision* (P) and *recall* (R), for which the mean values over all class labels are defined as [Dav06]

$$P = \frac{1}{L} \sum_{l=1}^{L} \frac{tp_l}{tp_l + fp_l} \tag{4.16}$$

$$R = \frac{1}{L} \sum_{l=1}^{L} \frac{tp_l}{tp_l + fn_l}, \tag{4.17}$$

where tp denotes the "true positive" classifications and L is the total amount of class labels. Either all or some of these metrics are applied in publications [P3] – [P6].

4.3 Collective Network of Binary Classifiers

The content-based audio classification framework applied in this thesis is detailed in this section. The section begins by introducing the topology of the framework, after which the classifier training and classification process are discussed.

4.3.1 Topology of the Classifier Network

The audio classification approach applied in this work consists of several *networks* of individual *binary* classifiers (NBC). A binary classifier (BC) is defined here as an any type of a classifier with *two* outputs. Essentially the purpose of such classifiers is to distinguish a particular audio class from any other. The binary classifier networks, NBCs, are designed in a way that the final output of each network is also binary. Thus, in order to solve an L-class classification problem, L separate NBCs are required, and the corresponding binary outputs are compared to derive the classification result. These L NBC "units" form a single entirety, which is called collective network of binary classifiers (CNBC).

The number of inputs for each classifier at each NBC is defined based on the total amount of extracted features. An extracted (and post-processed) feature vector is *divided* into F smaller feature subsets based on some feasible logic. For example, in publication [P4], $F = 6$ feature subsets are considered (such as MFCCs, LPCs, LPCCs, and spectral audio features). A separate binary classifier is then assigned for each of these feature subsets at each NBC. This allows applying a large amount of different types of audio features for classification problems, which may be very beneficial; in many cases a specific feature set is especially suitable for distinguishing certain types of audio classes, whereas not at all for others. On the other hand, applying a single huge feature vector (FV) for classification may become computationally infeasible due to the high number of classifier inputs.

The CNBC framework is illustrated in Figure 4.5 for F input feature subsets and L classes. The *fuser* BCs shown in the figure provide as their outputs the actual classification results in a binary form, which state the credibility of the classified item belonging to the class of their corresponding NBC. The fusers

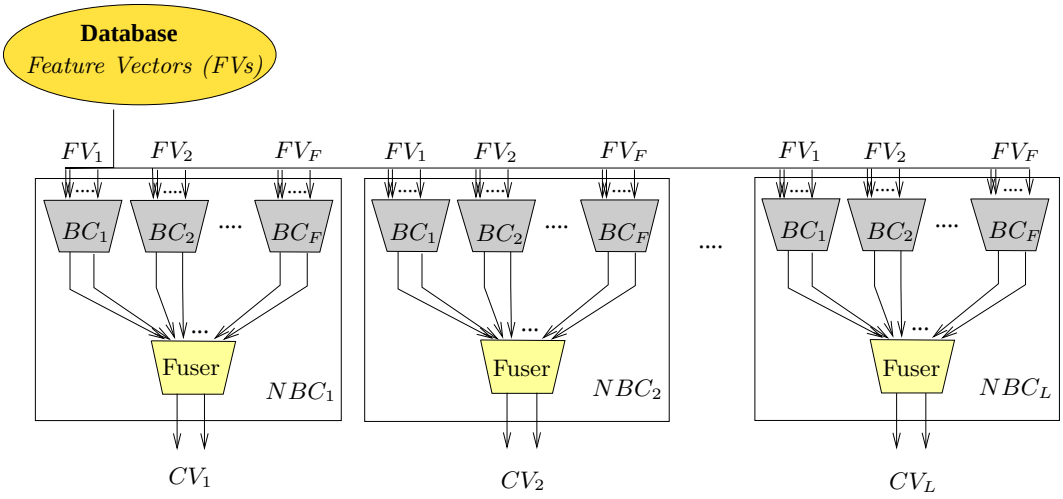

Figure 4.5: The structure of the CNBC framework for F input feature sets and L classes. Each network of binary classifiers provides a single binary output, called as *class vector* (CV). Copyright© 2011 IEEE.

are named after the "fusion" they perform to the outputs of the BCs on the NBC "input layer". The fundamental idea of the CNBC approach is to divide complex classification problems into several smaller and easier ones in order to avoid overly complex classifiers. Thus, the topology can be also stated as a "divide and conquer" - type of approach.

Based on their networked structures, each NBC in the CNBC topology can be compared to ensemble based systems [Pol06, Abb03]. In these, the strategy is to create many classifiers and combine their outputs such that the combination improves upon the performance of a single classifier (as in the case of random forest discussed earlier). Here the classifier *diversity* plays a key role, meaning that the individual classifier decision boundaries should be adequately different from each other. The diversity is achievable in several ways, for example by partitioning the input data or varying the classifier parameters or types [Pol06]. In each NBC, the diversity is achieved by using the different feature subsets for the classifiers and – more importantly – by applying different (optimized) classifier parameters to them. In this work, artificial neural network (ANN) classifiers are used in each NBC.

Structure-wise, a single NBC within the CNBC topology is analogous to a "stacked generalization" approach described in [Pol06]. The whole CNBC approach, however, is a novel one with its *evolving* classifiers and the fusion operation performed over the binary classifiers with different feature subsets. Moreover, devoting a separate NBC for each class represents a new and uniquely motivated topology with an attractive property of class and feature *scalability* discussed in section 4.3.4. In the following, the term "evolution" is used instead of "training" to emphasize the fact that, in addition to the conventional ANN parameters w

and θ, also the network structures of the individual ANN classifiers are iteratively optimized via the underlying MD PSO algorithm in an evolutionary manner.

4.3.2 Evolving Binary Classifiers

Any of the classifier types introduced in section 4.1.1 may be applied as binary classifiers in the CNBC framework. In the case of feed-forward ANNs – or multi-layer perceptrons – considered in this work, a specific *architecture* search space is defined, which consists of several indexed MLP structures. The indices correspond to the MD PSO dimensions, beginning from the most simple structure (the lowest dimension) and ending to the most complex one (the highest dimension). The particle positions in each dimension correspond to the MLP parameters (connections, weights, and biases) for the corresponding MLP structure. For each structure, the number of hidden layers and number of neurons in each layer is defined by using a vector R of the following form:

$$R = \{N_f, N_1, \ldots, N_h, \ldots, N_H, 2\}, \tag{4.18}$$

where N_f is the dimension of the fth input feature subset, FV_f (the input layer), N_h stand for the number neurons in the hidden layer h, and H is the maximum number of hidden layers included in the search space. Note that, since binary classifiers are applied, the number of neurons at the output layer is fixed to 2.

The MD PSO algorithm is applied to optimize the MLP parameters by defining the positional component of a particle a at iteration t, $\mathbf{p}_a^{d_a[t]}[t]$, as

$$\mathbf{p}_a^{d_a[t]}[t] = \Psi^{d_a[t]}\left\{\{w_{mn}^0\}, \{w_{mn}^1\}, \{\theta_m^1\}, \{w_{mn}^2\}, \{\theta_m^2\}, \ldots, \{w_{mn}^H\}, \{\theta_m^H\}, \{\theta_m^O\}\right\}, \tag{4.19}$$

where $\{w_{mn}^h\}$ represent a set of weights from mth neuron at hidden layer h to the nth neuron at hidden layer $h+1$, and $\{\theta_m^h\}$ is a set of biases for the mth neuron at layer h of a corresponding MLP structure $\Psi^{d_a[t]}$, where $m \in \{1, \ldots, N_h\}$ and $n \in \{1, \ldots, N_{h+1}\}$. Note that the input layer contains only weights, whereas the output layer ($h = O$) has only biases. In order to optimize these parameters, the MD PSO particles should converge to the global optimum in an optimal dimension. Mean squared error between the desired and obtained output can be stated in a form of fitness/measure function as

$$\mathcal{F}_{MSE} = \frac{1}{T}\sum_{p=1}^{T}\left(\mathbf{d}_p - \mathbf{y}_p\right)^2, \tag{4.20}$$

where $\mathbf{d}_p$ and $\mathbf{y}_p$ are the desired and obtained two-element class vectors (CV) for pattern p in a train set of size T. The term "class vector" is used to describe

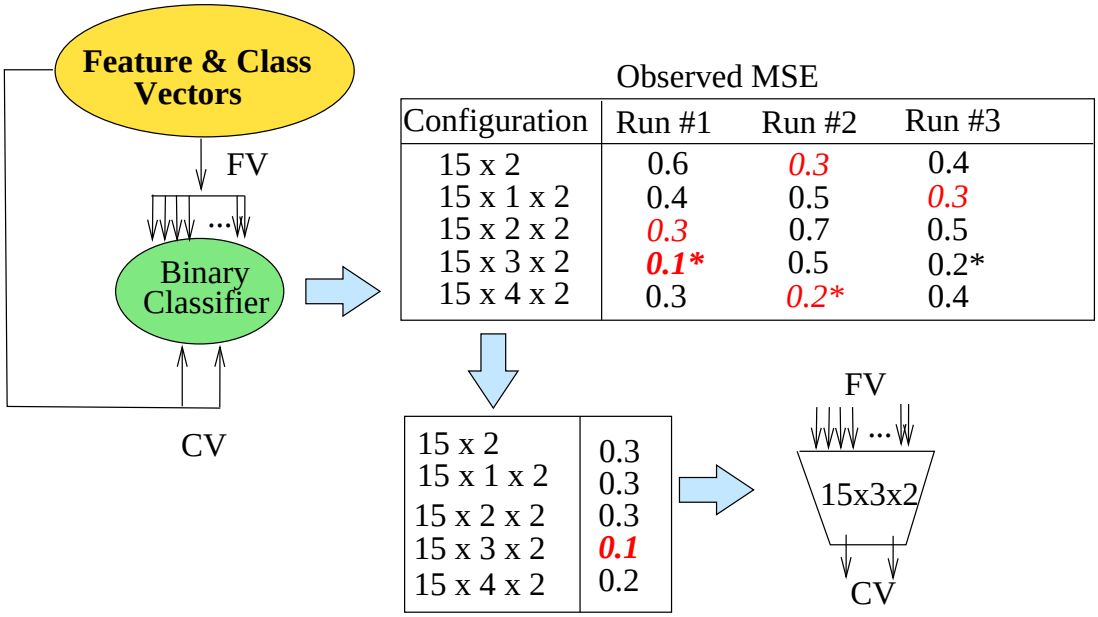

Figure 4.6: An example of an evolutionary MLP configuration update process. The upper table shows the obtained fitness values, where the best runs for each MLP configuration in the architecture space are highlighted. The best configuration in each run is marked with "*". In this case, the overall best configuration has 3 neurons in one hidden layer, and it is thus assigned for the (ANN) binary classifier. Copyright© 2011 IEEE.

whether or not a particular audio sample belongs to the class at stake. Obviously, the evolving process falls into the category of supervised learning, as training data with ground truth class vector are required, i.e. the desired output for (4.20) needs to be available.

Due to the stochastic nature of PSO, the evolutionary update of each classifier may be performed in several runs to improve the probability of converging to the global optimum. This is demonstrated in the case of MLPs in Figure 4.6, where a 5-dimensional architecture space with configuration vectors $R_1 = \{15, 2\}$, $R_2 = \{15, 1, 2\}$, $R_3 = \{15, 2, 2\}$, $R_4 = \{15, 3, 2\}$ and $R_5 = \{15, 4, 2\}$ are considered. Three separate parameter optimization runs are performed, and the obtained MSE fitness values are shown in the upper table. The best results are considered from *each* dimension, among which the optimal network structure and parameters are chosen for the binary classifier.

Considering the whole CNBC topology, the classifier evolution process is performed in two phases. In the first phase, the BCs in the NBC "input layer" with different feature subsets are evolved, while the fuser BC – since it uses the outputs of the BCs evolved in the first phase – is left for the second phase. The fuser BC can be thought of as a certain type of feature selection object, because it may emphasize those features which have the highest capability in discrimi-

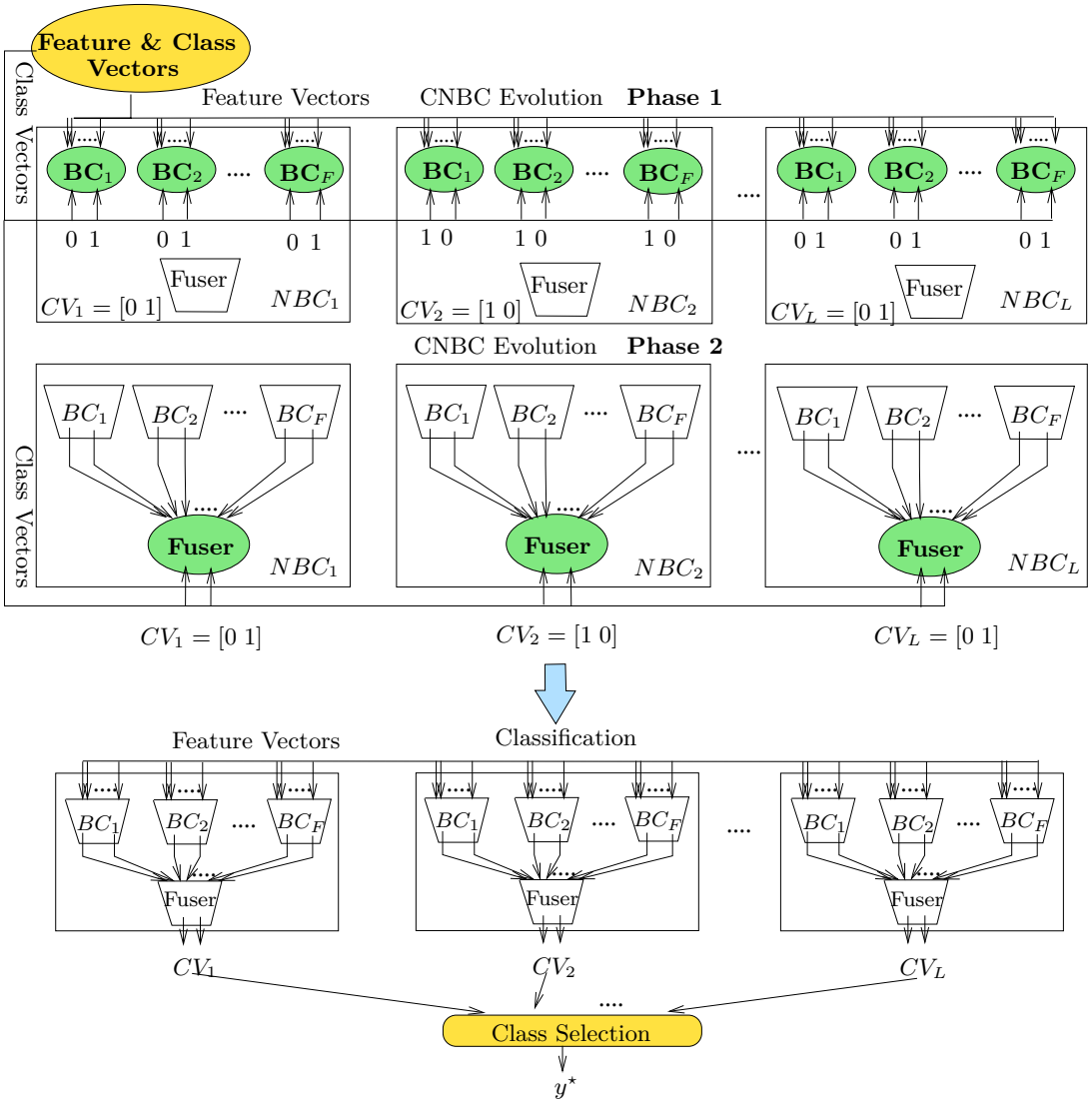

Figure 4.7: A flowchart of the two-phase classifier evolution and the final output class selection. The classifiers within elliptic shapes are being evolved by applying a specific subset of features as an input and the corresponding ground truth class vector as an output. In this example, the classified input FV belongs to the class number 2. Copyright© 2011 IEEE.

nating the class of the corresponding NBC from the others. Naturally, this is done by scaling the BC outputs according to the fuser BC parameters (i.e., the connection weights of an MLP), meaning that no hard selection is performed. An overview of the evolution process in shown in Figure 4.7, where the lower part illustrates the actual output class selection phase, which is described in detail in section 4.3.3.

4.3.3 Classification with CNBC

The overall output of the CNBC framework is obtained by combining the individual classifier outputs, i.e., the outputs of each NBC. The combination can be based on e.g. majority voting, whereas in CNBC a dedicated class selection technique is applied over the two-element class vector outputs of each NBC. Considering a uni-class database (in which each database item belongs to exactly one class), a "1-of-m" encoding with $m = 2$ is first applied to *every* binary classifier in the network. The encoding is done by comparing the output class vector elements and stating the output as

$$y = \begin{cases} positive, & \text{if } CV_{f1} > CV_{f2} \\ negative, & \text{else,} \end{cases} \tag{4.21}$$

where CV_{fj} corresponds to the jth element of the class vector of the fth binary classifier at a particular NBC. A similar encoding is also performed for the fuser BCs at each NBC. The overall classification result (class selection) is then obtained by applying a "winner-takes-all" strategy on all the individual NBC outputs. Here the "winner" class is defined as

$$y^{\star} = \underset{l \in \{1,\ldots,L\}}{\operatorname{argmax}} \left(CV_{l1} - CV_{l2} \right), \tag{4.22}$$

stating that the class label l with "most positive" outcome is selected. This means selecting the class label which provides the biggest positive difference between its two output vector values CV_{l1} and CV_{l2}. Because of the comparison between each class vector, the approach enables handling also cases with several positive class vector differences (or none) unambiguously.

4.3.4 Incremental Evolution of the CNBC

Due to its element-wise design, the CNBC can be dynamically scaled to a higher or lower number of classes in a feasible manner. For example, in the case of *incremental learning* [Pol06], where new data samples with a new class label are added to the database at hand, a new NBC may be added and evolved without necessarily changing the existing classifiers. This is a significant computational advantage and provides notable *scalability* to the CNBC structure. Whenever necessary, also the existing NBCs may be *incrementally* evolved by performing new MD PSO runs using either the updated or the original training dataset. The procedure is detailed and experimented in publication [P4] by adding new classes to the applied audio database. In short, the idea is that a new MD PSO run begins from the previous solution converged by the earlier MD PSO run(s), such that the swarm experience achieved so far is fully utilized.

In addition to the class scalability considered in [Pol06], the CNBC topology is also scalable with respect to the amount of extracted *features*: whenever a new feature subset is added, a corresponding new BC is inserted to each NBC in the framework. Feature scalability is not supported by the ensemble of classifier methods, such as Bagging, Boosting or Stacked Generalization [Pol06] [Dud01, Chapter 9], meaning that a new feature extraction will eventually make the classifier ensemble obsolete and require retraining it from scratch. Note that in CNBC the scaling can be performed also downwards by simply removing NBCs (or the BCs within) from the framework according to the class or feature changes.

Incremental learning is often problematic for traditional classifiers, causing a phenomenon known as *catastrophic forgetting* [Fre99]. This means that all the previously acquired information is lost, which is not only costly in computational sense; the original dataset may be already unavailable due to corruption or access restrictions. Besides of CNBC, only very few attempts have been proposed to address this problem. The *Learn*$^{++}$ algorithm has shown to accomplish incremental learning on several applications [Pol01]. However, the algorithm creates an ensemble of ensemble classifiers, one ensemble for each updated database, meaning that learning new classes requires an increasingly large number of individual classifiers. Such a structure is not as scalable as the presented CNBC topology.

4.3.5 Discussion

Both in publications [P3] and [P4], the CNBC classification performance is compared to several SVMs with different kernels and parameter settings. The one against one - approach is applied to enable multi-class SVM classification. These comparisons reveal the high potential of the CNBC considering complex classification tasks with high number of distinct audio classes and features. It should be noted, however, that unless the applied training dataset is representative enough, potential over-learning may occur when optimizing the classifier parameters with static train data. Nonetheless, as discussed above, the CNBC is also directly applicable to dynamic databases with both varying number of classes and features.

The computational complexity of the CNBC is dependent upon several factors. First of all, the training method applied for evolving the classifiers affects the computational time by some varying amount, which is dependent on the training method parameters. In [P3] and [P4], two methods are employed in evolving the artificial neural networks applied in the CNBC, namely the sequential back-propagation (BP) and the multi-dimensional PSO (MD PSO) algorithms. The sequential BP algorithm performs an exhaustive search over all the network architectures defined in the architecture space to find the optimal ANN structure and parameters. It can be easily understood that the computational time is thus directly proportional to the size of the architecture space, i.e., the number of possible network configurations, N_c. Another affecting parameter is the number of *epochs*, E, considered in the BP training, which stands for the number of times

the training dataset is fed to the network during the parameter training process. Hence, an approximate order of $O(EN_c)$ computational time can be defined for the sequential BP algorithm. The MD PSO algorithm, instead, does not perform such an exhaustive search, but strives for converging to the global best solution of the architecture space in a stochastic manner. The computational time required by MD PSO is dependent on the size of the training dataset, T, the number of particles, P, and the number of iterations, I. All of these define the total number of forward propagations performed during the process, such that, by denoting μ_t as an abstract time required for computing the Eq. (4.20) over an *average* network in the architecture space, an approximate computational complexity of order $O(TPI\mu_t)$ can be determined for the process. It should be noted that, for the both methods, the parameter selection is a trade-off between the required computational time and the accuracy of the results.

As a concrete comparative example, the training process performed in [P3] for an 8-class database with the BP method took approximately 1 - 1.5 h with the specified computer. However, as one would imagine, the processing time was significantly lowered by decreasing the number of epochs. Considering the MD PSO approach, with the defined architecture space the processing took somewhat longer than with BP when comparable parameter settings were used. Unfortunately, no precise and comprehensive measurements regarding to the computational times are documented from the experiments for deeper analysis. However, it can be roughly said that for relatively small architecture spaces the BP method performs faster than MD PSO, but the situation should change to the opposite rather rapidly as the number of network configurations, N_c, is increased e.g. to those experimented in [Kir09].

Other factors affecting the complexity of the CNBC framework itself are the applied binary classifier types, as well as the number of feature vectors and audio classes considered in a particular problem. However, due to the fact that the individual NBCs are independent of each other, the CNBC structure is directly applicable to parallel grid computing. This allows evolving each NBC with an own CPU (central processing unit), which can – in the current implementation – speed up the optimization process by the amount NBCs in the structure. In the scope of this thesis, the *Techila* computation grid [Tec] is utilized in the evolutionary optimization processes performed in publications [P4] – [P6].

Chapter 5

Evolutionary Audio Feature Generation

Conventional audio features are required for estimating several types of characteristics and attributes from audio signals, but occasions may occur in which the obtained properties are not descriptive enough for decent and accurate audio content analysis. Especially when the database size and the number of audio classes increase beyond a certain magnitude, the discrimination between individual classes becomes more challenging. This is why feature selection and feature synthesis (also called generation/construction/transformation) has gained an increasing amount of attention in the literature in the context of data mining and machine learning. This chapter describes a stochastic optimization-based approach for combined feature selection and generation in the field of content-based audio classification and retrieval.

5.1 System Overview and Preliminary Work

Considering an ideal feature generation case, a feature generation system (also called here as a feature *synthesizer*) receives as its input a specific set of (low-level) audio features, selects the most representative and appropriate subset among them, and finally combines and modifies the selected features by applying a proper set of transformation *operators* and feature *weights*. As its output, the system generates a set of new and more descriptive (artificial) audio features with respect to a designed fitness function or other quality measure assigned for the operation. Such an ideal feature synthesis operation is demonstrated in Figure 5.1 for the purpose of clustering, where two-dimensional features of a 3-class dataset are successfully synthesized into new, clearly distinct clusters. In many cases, such an operation enables obtaining significantly improved classification and/or retrieval results compared to the original feature distribution. In the next two sections, some of the earlier contributions and important studies made in the field are discussed.

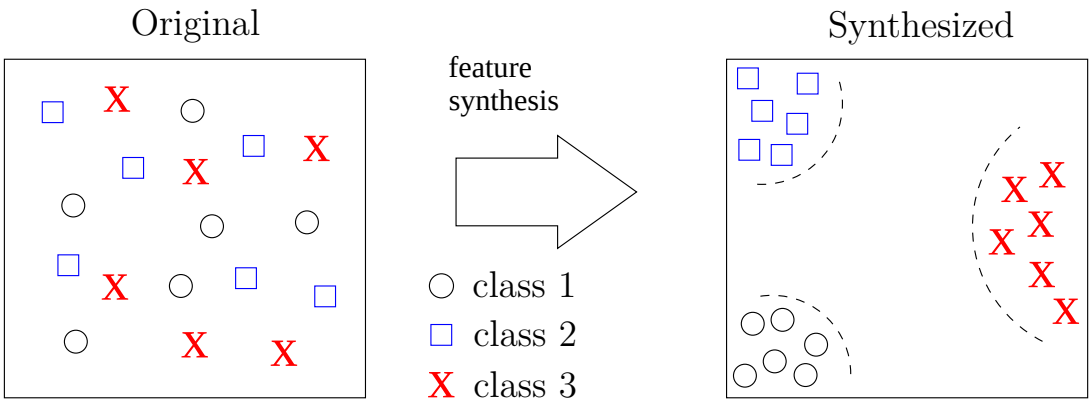

Figure 5.1: An overview of an ideal feature synthesis process.

5.1.1 Feature Selection

Feature selection [Liu08] is beneficial whenever the feature space dimensionality is too high for a reliable statistical or structural analysis. This so-called *curse of dimensionality* phenomenon [Bel61] is caused by the fact that the available data becomes too sparse at high dimensions. Feature selection is applied for choosing an expressive and compact subset of features among the original ones. Genetic algorithms (GAs) [Gol89] and genetic programming (GP) [Koz92] are traditionally applied for the task [Hua06], [Zhu08].

More recently, PSO was also applied for selecting features which were originally extracted by using DCT and discrete wavelet transform [Ram09]. Comparable face recognition results to a GA-based feature selection were obtained with a benefit of fewer features needed. Also in [Lin08], a PSO-based feature selection was tested with SVMs to various classification problems. Moreover, a survey of several feature selection approaches is presented in [Guy03], where the authors conclude that feature "construction" should be applied together with feature selection to obtain improved performance and more compact feature sets. Hence, in this work the main attention is drawn to synthesizing new audio features from the conventional ones by the means of stochastic optimization.

5.1.2 Feature Generation

One of the earliest forms of feature generation are based on *grammars* of feature construction functions [Mar02]. However, such approaches cannot generalize to more realistic data mining cases with raw signals instead of symbolic data [Pac09]. Stochastic optimization, instead, is well suitable for searching artificial feature combinations and representations that go beyond human imagination. Thus – in accordance with the general scope of this thesis – "trial and error" type of approaches are generally applied for the task. In [Rit02], for example, a combination of both feature selection and generation is proposed based on a modified GA. The

research basically verifies that generated features can significantly improve the learning performance when compared to direct data observations. In [Lin05], a co-evolutionary GP-based approach is proposed for synthetic aperture radar (SAR) image classification. The approach allows applying several sub-populations to GP, whose synthesis outputs are then combined in forming the final synthesized FVs. With less number of synthesized features needed, the classification results were comparable to (or better than) those obtained with the original features.

Probably the first audio feature generation system was the one proposed by Pachet and Zils [Pac03]. They applied GP in an extractor discovery system (EDS) framework to explore a large operator function space for automatically discovering new high-level audio features. More recently, Pachet and Roy [Pac09] applied the EDS framework with what they call *analytical features* (AF). These represent a large subset of all audio signal processing functions and are expressed as a functional term consisting of basic operators. The main idea is to apply genetic transformations to improve the current population of the (initially random) AFs, while the fitness of each AF is evaluated using an SVM classifier.

Generally in audio signal processing, *ad hoc* and domain-specific audio features have recently gained a considerable amount of attention. These are mainly applied to some very specific audio classification problems. For example, Mörchen et al. [Moe06] constructed a large set of features by applying cross products between several existing feature functions, resulting to approximately 40 000 audio features in total. A similar example was presented by Mierswa and Morik [Mie05], who introduced method trees consisting of ad hoc features for a given audio signal. The trees are automatically generated using GP in combining elementary feature extraction *methods*. The authors report improvements in music genre classification accuracy over approaches with traditional audio features.

5.2 Particle Swarm Optimization for Feature Synthesis

In this work, the feature synthesis is performed via an MD PSO-based parameter optimization approach. The method was initially applied in [Kir11], where the performance was tentatively verified in the context of image retrieval. To motivate the selection of PSO over the traditionally applied GP, the two algorithms are compared in Table 5.1 [Chu08]. The main advantage of PSO over GAs is that the algorithm provides more *profound* intelligent background, and it can be performed more easily than GAs [Shi05]. Also, the computation time of PSO is usually less than for GAs, as all the particles in PSO tend to converge to the best solution rather quickly [Ebe98].

In the following, the MD PSO-based feature synthesis process is discussed in detail. The discussion includes the system overview, particle encoding, and the

Table 5.1: Comparison of GA and PSO as search algorithms

Property	**GA**	**PSO**
Genetic operators	included	excluded
Key functions	-crossover -mutation	-social particle interaction -particle velocity updates
Information source	all the chromosomes (the whole population moves in one group)	the global best particle (evolution only looks for the best solution)
Update occurrence	probabilistic (crossover and mutation rates)	all particles are updated after each iteration
Local optimum	can become easily trapped	can avoid well the local optima

fitness metrics applied for evaluating the synthesized audio features.

5.2.1 Evolutionary Feature Synthesis Overview

The audio feature synthesis approach described in this work provides an attractive property of searching for both the optimal synthesis parameters and the optimal *dimensionality* of the feature search space. This is achieved by applying MD PSO along with the FGBF algorithm described in section 2.2.2 to the optimization. In order to perform successful feature synthesis, the proposed system, with a specified synthesis *depth* value K,

1. Selects $K+1$ original features $f_0, \ldots, f_K$,
2. Scales the selected features with proper weights $w_0, \ldots, w_K$,
3. Selects operators $\theta_1, \ldots, \theta_K$ among a large collection to be applied over the selected and scaled features, and,
4. Binds the output with a non-linear operator (such as tangent hyperbolic).

Stating that $\theta_n(f_a, f_b)$, where $n \in \{1, K\}$, denotes performing an operator θ_n over features f_a and f_b, a formula for synthesizing a new feature s_j is defined as

$$s_j = \tanh\left(\theta_K(\theta_{K-1}(\ldots, \theta_2(\theta_1(w_0 f_0, w_1 f_1), w_2 f_2), \ldots), w_K f_K\right). \qquad (5.1)$$

As shown in the formula, first the operator θ_1 is applied to the weighted features f_0 and f_1, after which the operator θ_2 is performed over the result of this first operation and the weighted feature f_2. The process continues similarly, until

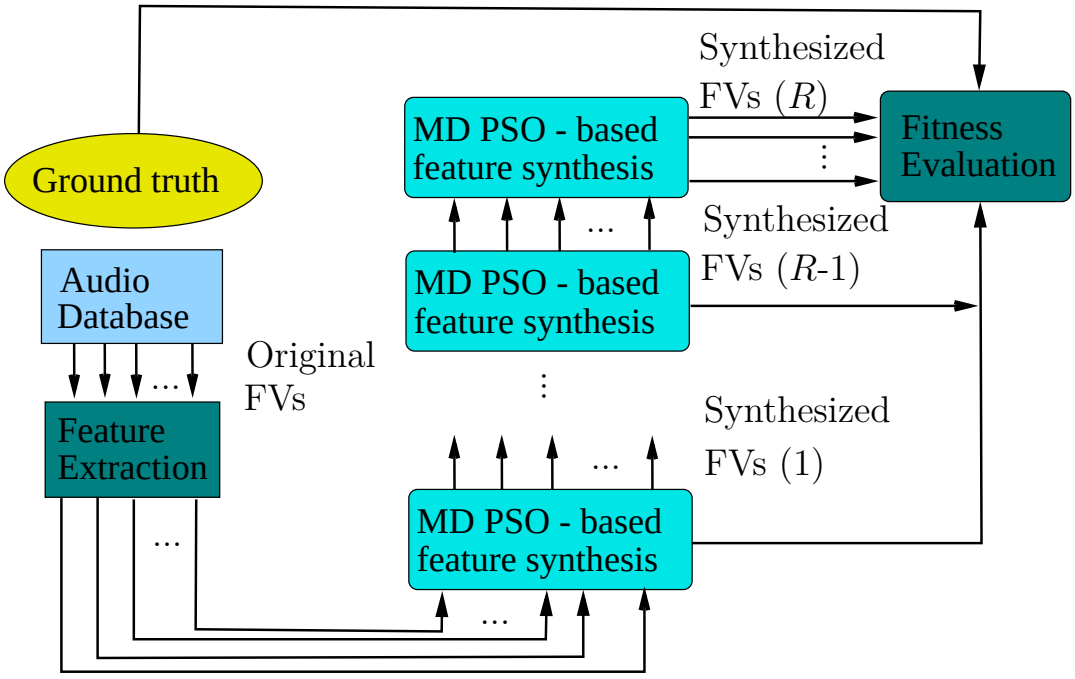

Figure 5.2: A flowchart of an evolutionary feature synthesis process performed over R runs.

finally the operator θ_K is applied to the results of *all* the previous operations and the weighted feature f_K.

The optimization process of the parameters included in (5.1) is a high-dimensional and multi-modal problem requiring stochastic optimization methods. In publications [P5] and [P6], also the fractional global best formation (FGBF) and simulated annealing (SA) algorithms described in sections 2.2.2 and 2.3 are co-employed with MD PSO for the task. The process is called "evolutionary" due to the evolving nature of the PSO algorithm and the fact that the process itself can be performed over several consecutive *runs*, in an evolving manner. This means that the process adopts the synthesized feature vectors from a previous run as its input for the next one, as demonstrated in Figure 5.2 with R synthesis runs. The arrows in the figure correspond to individual feature subsets (such as MFCCs, LPCs, etc), each of which an individual feature synthesizer is evolved. The approach is hence directly applicable to parallel processing, in which a separate CPU is assigned to each individual feature set. As shown in the figure, the fitness evaluation of each synthesized feature vector – corresponding to one feature subset – is performed after every synthesis run. In this work, the retrieval metrics discussed in section 5.2.3 are applied for the task.

5.2.2 Encoding of the Particles

The positions of the MD PSO particles in the parameter space are encoded such that, once decoded, each of them represents a potential solution on performing the feature synthesis operation. In other words, the position vector of each particle can be seen as a potential feature synthesizer. Every particle encapsulates a complete set of synthesis parameters: the indices of the selected input features, f_n, the corresponding feature weights, w_n, and the selected mathematical operators, θ_n. Since the feature space dimensionality corresponds to the dimension (length) of the synthesized output feature vector, the jth output vector element is synthesized as described by the jth positional element of a particle a at iteration t, $p_{aj}^{d_a[t]}[t]$, where $j \in \{1, \ldots, d_a[t]\}$.

Referring to the four system steps shown in section 5.2.1, each particle *element* $p_{aj}^{d_a[t]}[t]$ must contain $K+1$ indices for selecting the original features, $K+1$ feature weights, and K operators for synthesizing the corresponding jth output feature. For this, the positional elements of each particle in the swarm are encoded in a vector form of length $2K+1$, including $K+1$ *A-type* and K *B-type* components. These define the synthesis parameters as

$$\begin{aligned} f_n &= \lfloor A_n \rfloor + 1, \qquad n \in \{0, \ldots, K\} \\ w_n &= A_n - \lfloor A_n \rfloor, \quad n \in \{0, \ldots, K\} \\ \theta_n &= \lceil B_n \rceil, \qquad\quad n \in \{1, \ldots, K\}, \end{aligned} \tag{5.2}$$

where $\lfloor \cdot \rfloor$ and $\lceil \cdot \rceil$ denote the floor and ceiling operators, respectively. Denoting the input FV dimension as F and the total number of available operators as Θ, the allowed values for the components are defined as $A_n \in [0, F[$ and $B_n \in]0, \Theta]$.

As an illustrative example, Figure 5.3 shows a feature synthesis procedure performed over an 8-dimensional input feature vector at run r, $FV(r)$. At iteration t, the particle a is located at position $\mathbf{p}_a^{d_a[t]}[t]$ in a dimension $d_a[t] = 6$, which corresponds to the length of the synthesized vector $FV(r+1)$. For the sake of simplicity, the synthesis depth value is set to $K = 3$, meaning that only $K + 1 = 4$ features are selected from the input vector $FV(r)$. Correspondingly, each particle vector element $\mathbf{p}_{aj}^6[t]$, $j \in \{1, \ldots, 6\}$, includes thus $2K + 1 = 7$ encoded synthesis components ($A_0, \ldots, A_3$ and $B_1, \ldots, B_3$), and each element j is applied to synthesize the corresponding output feature for $FV(r+1)$ (although only the synthesis process of the first element is shown in detail in Figure 5.3). Based on the dimensionality of the input FV ($F = 8$) and the total number of mathematical operators ($\Theta = 5$), the value ranges for the A and B components are derived as $A_n \in \{0, \ldots, 7\}$ and $B_n \in \{1, \ldots, 5\}$. For synthesizing the first output feature, the current particle position suggests selecting the 7th, 3rd, 1st, and again the 3rd input feature from $FV(r)$, whereas the corresponding selected

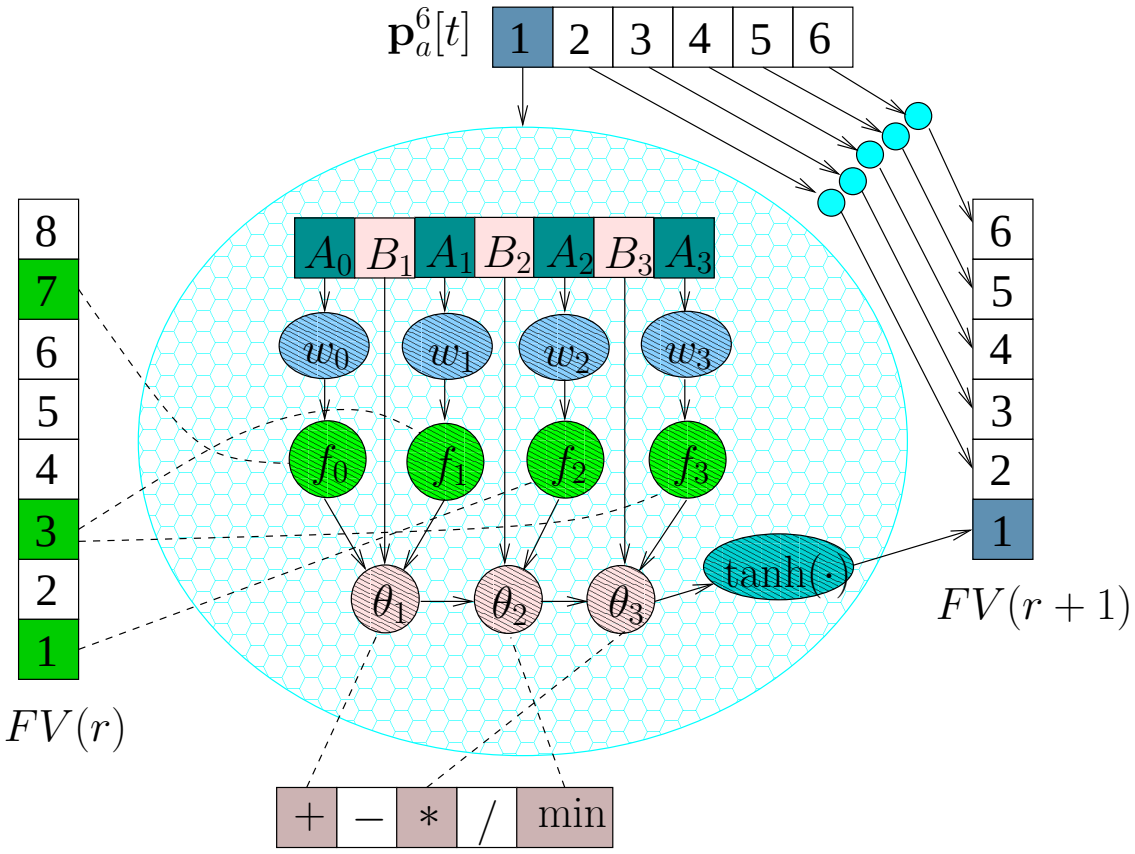

Figure 5.3: Illustration of a synthesis process of a 6-dimensional FV.

operators are "+", "min" and "∗". Based on (5.1), the first output feature is thus synthesized as

$$s_1 = \tanh\Big(\min((w_0 f_{[7]} + w_1 f_{[3]}), w_2 f_{[1]}) * w_3 f_3\Big), \tag{5.3}$$

where $f_{[n]}$ denotes the nth element of the input FV.

5.2.3 Retrieval Evaluation Metrics

Traditionally in audio content-based retrieval, certain (dis-)similarity measures, such as Euclidean distance, are applied to measure the distances between the FVs of a *queried* database item and each item belonging to the database. Similarity measures have been studied and compared e.q. in [Hel10], whereas in this work the Euclidean distance is used for demonstrating the performance improvement achieved via the evolutionary feature synthesis (EFS) process. In this work, the retrieval performance is evaluated using the average precision (AP) metric and the so-called average normalized modified retrieval rank (ANMRR), defined in

MPEG-7 standard as [Man02]

$$ANMRR = \frac{\sum_{q=1}^{Q} NMRR(q)}{Q} \leq 1, \tag{5.4}$$

where

$$\begin{aligned} NMRR(q) &= \frac{2AVR(q) - N(q) - 1}{2W - N(q) + 1} \leq 1, \\ AVR(q) &= \frac{\sum_{k=1}^{W}(q)R(k)}{N(q)}, \\ W &= 2N(q). \end{aligned} \tag{5.5}$$

The term q in (5.4) and (5.5) denotes a single query, $N(q)$ is the number of relevant samples included in a set of Q retrieval experiments, and $R(k)$ corresponds to the rank of the kth relevant retrieval within a window of W retrieval results (window "length" W meaning the number of retrieval results taken into account for one query). An average rank obtained from a query q is then denoted as $AVR(q)$, from which the first retrieval result is excluded, due to the fact that the highest similarity value is always obtained with the queried item itself. This latter procedure is performed in the normalization phase, yielding thus the normalized modified retrieval rank ($NMRR$) measure. In an ideal case, the first $N(q)$ retrieval results are all relevant to the query item, resulting to the best possible retrieval performance, i.e. $NMRR(q) = 0$, whereas in the worst case (where no relevant items are retrieved within the first $N(q)$ retrievals), the $NMRR$ results to $NMRR(q) = 1$. The $ANMRR$ value is computed in a *batch query* mode, that is, by querying all the database items one by one.

In order to improve the audio retrieval performance by the means of feature synthesis, an intuitive approach for constructing a proper fitness function would involve computing either the AP or the ANMMR measure. However, these would be computationally infeasible for large databases, as the both measures require conducting a separate batch query for every fitness evaluation during the synthesis process (i.e., selecting each item in the database as a query item, performing a separate query for each of them, and finally taking the average of the obtained retrieval results). Alternative approaches are therefore needed, and in this work the effort is put on providing a maximal separation between the features of different classes in the feature space, as demonstrated earlier in Figure 5.1. This should, in turn, result to improved retrieval performance. The two fitness measures experimented in this work are discussed in the next section.

5.2.4 Fitness Measures

The optimization of the EFS parameters can be performed with respect to any fitness function that suits to the problem at hand. The first fitness function applied in this thesis – called as *discrimination measure* (DM) – computes the absolute distance between the synthesized feature vectors in the feature space. The second one utilizes the approach employed in artificial neural networks, where a unique target vector is assigned for each audio class in the problem. The second measure is hence called *target vector assignment*.

Discrimination measure

This measure is based on clustering, in which the compactness and separation of different clusters are the main criteria for the design. Denoting the class labels of an L-class database as $l_1, \ldots, l_L$ and the corresponding class centroids as $\mu_1, \ldots, \mu_L$, the DM can be defined over a set of synthesized feature vectors, $S = \{s\}$ as,

$$DM(S) = fp(S) + \delta_{mean}(l_n)/\delta_{min}(l_n, l_m), \tag{5.6}$$

where $fp(S)$ is the number of false positive FVs occurring among S, stating that these FVs are located in a closer proximity to some other class centroid than their own. The terms $\delta_{mean}(l_n)$ and $\delta_{min}(l_n, l_m)$ correspond to average intra-class distance and the minimum centroid distance among the classes, respectively, and are defined as

$$\begin{aligned} \delta_{mean}(l_n) &= \frac{1}{L}\sum_{n=1}^{L}\frac{\sum_{\forall s \in l_n}\|\mu_n - S\|}{|l_n|}, \\ \delta_{min}(l_n, l_m) &= \min_{n \neq m}(\|\mu_n - \mu_m\|). \end{aligned} \tag{5.7}$$

This means that the $DM(S)$ strives for minimizing the intra-cluster distances and maximizing the shortest inter-cluster distance. As shown in Figure 5.1, in an ideal case every synthesized FV lies in the closest proximity of its own cluster centroid, leading thus to a high class discrimination.

Considering audio retrieval, minimizing the DM, however, does not always lead to improved retrieval results. This is because those query samples located at the outskirt of their own class centroid may be actually in a closer proximity to some other feature vector located at the outskirt of its corresponding class. This results in irrelevant retrieval results, which is the main reason for applying also another fitness measure for the synthesis task.

Target vector assignment

The second fitness measure is based on an idea of finding the synthesis parameters that produce a desired unique *binary* target vector for each audio class. As mentioned above, this is the same logic that is utilized in ANNs for finding the proper network parameters. The fitness value of this measure is computed by taking the MSE between the obtained and the targeted output vectors, in a similar fashion to bullet state estimation in section 3.4.3 or evolutionary classifiers in section 4.3.2. However, since the output dimensionality of EFS is not fixed in advance, a separate target vector needs to be generated for each dimension $d = \{D_{min}, \ldots, D_{max}\}$. Also, in order to provide a clear distinction of the desired target vectors between each class, the binary target vectors need to be set as different from each other as possible. For this, an error-correcting output code (ECOC) [Die94] analogy is applied, in which two criteria are considered for generating proper target vectors:

1. Row separation: The target vectors of each dimension should be well separated from each other

2. Column separation: The individual elements of each target vector should be well separated from each other,

both in the sense of Hamming distance. A large vector separation allows minor fluctuations to occur between the synthesized and desired output vectors, without losing the discrimination between different classes. In other words, the higher the separation between individual target vectors, the more *inaccuracy* is allowed for the synthesis parameters. Moreover, note that the target vector *elements* of each dimension can be thought of as individual binary classification tasks as such (between the vectors having a value "1" and those having a value "-1" in a specific dimension d). Now, due to the fact that two arbitrarily chosen audio classes have a varying similarity between each other, some of these classifications are significantly easier than others. Therefore – since the pre-determined target vectors are applied independently to the corresponding audio classes – it is also beneficial to keep these binary classification tasks as different from each other as possible by maximizing the column separation.

Constructing the binary target vectors with the aforementioned constraints is performed using the following procedure:

1. Compute the number of bits, N_b, needed to present the total number of classes L.

2. Construct L empty target vectors.

3. For each target vector, assign a binary representation of the class label index l_{n-1}, $n \in \{1, \ldots, L\}$ as the first (or next) N_b target vector values. For

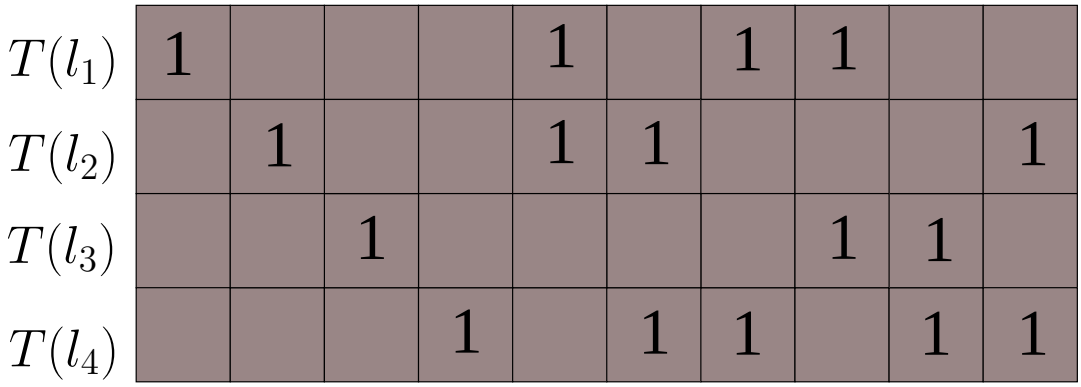

Figure 5.4: A set of binary target vectors assigned for 4 audio classes based on ECOC encoding.

example, if $n = 3$ and $N_b = 4$, assign "-1 -1 1 -1" as the first (or next) 4 vector values.

4. Move the target vector of class label l_1 to class label l_L and shift the other target vectors "upwards" by one class ($l_n \Rightarrow l_{n-1}$, $n \in \{2, \ldots, L\}$). That is, move the target vector of class l_8 to class l_7, the vector of l_7 to l_6, etc.

5. Repeat the steps 3 and 4 until D_{max} target vector values are assigned.

6. Replace the first L vector values of each target vector with a 1-of-L coded [Bai03] section.

For further clarification, the target vectors assigned for a 4-class database are demonstrated in Figure 5.4. The maximum dimension is set to $D_{max} = 10$ and the number bits needed to present the four classes is $N_b = 2$. For clarity, the "-1" values in the target vector table are presented by empty entries. The term $T(l_n)$ stands for a target vector for audio class l_n. Corresponding to the presented target vector generation procedure, this example demonstrates the 1-of-L coding section for the first $L = 4$ elements of each target vector, after which the shifted 2-bit representations of the class indices follow. For the vector dimensions $d < D_{max}$, only the d first elements of the target vectors are considered. This way the FGBF algorithm discussed in section 2.2.2 can be applied to combine particle position elements, as the common elements of target vectors with different lengths are always identical. This is done in publications [P5] and [P6].

The fitness values for the jth vector elements of all the synthesized FVs are finally computed as

$$\mathcal{F}(S_j) = \sum_{n=1}^{L} \sum_{\forall s \in l_n} (T(l_n)_j - s_j)^2, \tag{5.8}$$

where $T(l_n)_j$ is the jth element of the target vector of class l_n and s_j denotes the jth element of a single synthesized FV belonging to class l_n. These *fractional* fitness values $\mathcal{F}(S_j)$ are then added together and normalized with respect to the number of dimensions to form the overall fitness value $\mathcal{F}(S)$, as follows:

$$\mathcal{F}(S) = \frac{1}{L}\sum_{j=1}^{L}\sum_{n=1}^{L}\sum_{\forall s\in l_n}(T(l_n)_j - s_j)^2 + \frac{1}{(D_{max}-L)^{\alpha}}\sum_{j=L+1}^{D_{max}}\sum_{n=1}^{L}\sum_{\forall s\in l_n}(T(l_n)_j - s_j)^2. \tag{5.9}$$

In (5.9), the probability of converging to higher dimension is moderately increased by an additional power parameter α. This is because the first L vector elements with the 1-of-L encoding are usually the easiest ones to synthesize (and thus mainly favoured in the dimension search by the MD PSO algorithm). It is also assumed in (5.9) that $D_{min} > L$. The DM metric is applied as a fitness function in publication [P5] for a 16-class database, whereas in [P6] the both fitness measures are used with two databases of 6 and 12 audio classes.

5.3 Comparison to Artificial Neural Networks

In a sense, the described EFS technique can be seen as a generalized form of artificial neural networks. Considering the four algorithmic steps listed in section 5.2.1, a single-layer perceptron (SLP) classifier acts by performing the steps 2 and 4. This is because neither feature nor operator selection is performed in an SLP classifier. Instead, SLP adds a bias value to its weighted features, which – whenever found beneficial – can be mimicked also in the EFS approach by inserting an additional constant value of 1 at the end of each original input feature vector. This constant can be then selected and scaled by the synthesizer among the other selected features. Considering the feature synthesis procedure shown in equations (5.1) and (5.3), it can be noticed that by setting the synthesis depth as $K + 1 = F$, discarding the feature selection as $f_{[n]} = f_n$, and setting each operator θ_n to '+', the EFS becomes identical to SLP. However, note that the dimensionality of the SLP output layer is fixed, whereas in the EFS approach the output dimension is (as mentioned) optimized within a specified range.

Correspondingly, performing several consecutive EFS runs – as shown in Figure 5.2 – is comparable to a multi-layer perceptron topology, or, in fact, any feed-forward ANN. Similar to SLP, an MLP does not involve feature selection, and it also performs with a fixed (summing) operator and fixed output dimension. Interestingly, performing EFS with the above parameter settings corresponds to an MLP approach with a one-to-one analogy between the number of hidden layers and the number of runs performed. In this sense, it can be stated that a regular

feed-forward ANN is a *special case* of the EFS technique. This also means that regular ANNs may be considered and applied for feature generation purposes, as they do perform mapping from their original feature space to a new one with either the same or different (pre-determined and fixed) dimension. An important difference between the MLP and the EFS technique is that in EFS the fitness of the synthesized feature vector is evaluated after each run, whereas in MLPs only the final fitness value occurring at the output layer is shown and the intermediate network layers are hidden.

Comparative evaluations between EFS and SLP/MLP "feature synthesizers" are performed in publication [P6] in the context of audio retrieval. The main conclusions of these evaluations are that, due to its extensions compared to ANNs, the EFS is able to synthesize notably more suitable audio features for retrieval purposes than the compared evolutionary ANNs, both in the sense of retrieval accuracy and output FV dimensionality. However, despite the possibility of efficient distributed computing with EFS, the increase on computational complexity with EFS compared to ANNs may still favour applying a regular ANN with a fixed configuration in cases where relatively small and simple (in the sense of class separability) audio databases are considered.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

In this thesis, several stochastic optimization techniques and their extensions were described, discussed, combined and experimented with selected practical applications. The main effort was put in providing a solid and versatile foundation of algorithms for solving – in a novel manner – high-dimensional, multi-modal and (possibly) non-linear optimization problems with considerable accuracy. As shown in the results of this thesis, these algorithms have a strong potential in providing solutions to such complex and possibly non-linear optimization problems that cannot be solved in an analytical and deterministic manner. Analytical optimization methods, such as deterministic Kalman filter, are only feasible to linear and Gaussian problems. Considering non-convex or large-scale optimization problems, deterministic methods may not be able to derive a globally optimal solution within reasonable time [Lin12]. Therefore, stochastic solutions were applied to the problems considered in this thesis.

Two main application areas were covered in this thesis, namely sniper positioning with bullet trajectory, speed and caliber estimation ([P1], [P2]) and applications related to content-based audio classification and retrieval ([P3] – [P6]). In the first area, Bayesian-estimation based particle filtering and simulated annealing were applied in finding proper shock wave modelling parameters based on the real shock wave signals observed with a microphone array. The sound wave direction of arrival estimation used for "pinpointing" the sniper was based on time differences between the microphones on the array. The second application area was mainly covered by the MD PSO-based optimization of individual audio classifiers used in the described CNBC topology. Simulated annealing was also applied together with MD PSO in the proposed feature synthesis approach, in which the synthesis parameters were optimized such that the overall audio retrieval performance could be improved by using the new synthesized features.

In publication [P1], particle filtering was shown to provide reliable bullet

state estimation results with a moderate computational effort. The estimation approach was able to solve the multidimensional inference problem encountered in the process. Depending on the problem, the relation between the speed and accuracy of the method could be adjusted by setting the number of particles accordingly. The GCC-PHAT criterion used for the estimation in [P1] was also used and extended with a MSE criterion in publication [P2]. In [P2] the bullet state estimation was based on simulated annealing. A separate gunshot detection algorithm was also applied and the actual estimation process was set as dependent on the detected gunshot events. The estimation results in overall are applicable considering the military and police applications, especially the shooter position estimates, which are found with an accuracy of few meters. It should be noted, however, that challenging weather conditions, such as strong turbulent wind, may degrade the localization performance. Other potential error sources are simultaneous gunshots, which can confuse the direction estimation if not detected correctly, as well as some urban environments where buildings may cause strong echoes. Nonetheless, in the made experiments the occasional difficulties encountered in converging to the global maximum could be often handled by utilizing muzzle blast-based heuristics, such as the shooter position estimates.

The CNBC framework was first proposed for audio content-based classification in publication [P3]. Using an 8-class audio database with varying types of audio classes, the achieved classification results were better than those obtained with an SVM classifier. The applied audio classes were selected to demonstrate the generalization ability of the CNBC, by showing that it can successfully handle such unconstrained and general types audio classes. Although the overall accuracy improvement was not that prominent with the tested database, it should be mentioned that the compared SVM classifier – which is a powerful classification algorithm as such – was applied with its best kernel and parameter settings found. No separate statistical analysis of the results was performed, which may question the true statistical significance of the achieved accuracy improvement. However, the framework was also tested over dynamic databases by gradually increasing the number of classes from 4 to 8, in which it was shown that the dynamic CNBC design could cope up with such incremental evolution without necessarily needing to re-evolve the existing classifiers between the class additions. The extended evaluations performed in publication [P4] with 12- and 20-class databases verified the results of [P3] with large databases. The properties of scalability and dynamic adaptability of the CNBC negate the necessity of configuring the classifier parameters and configurations strictly for some specific audio dataset only. Despite such generalisation, the classification results showed that the CNBC can compete and also surpass other powerful classifiers such as SVMs, and RFs with their best kernels and parameters set *a priori*.

The EFS research presented in publications [P5] and [P6] provided notable improvements in terms of audio content-based retrieval performances. The feature extraction procedure was changed somewhat compared to publications [P3]

and [P4]. This was to increase the number of different types of feature vectors and also to ease the repeatability of the work, as now all the features were extracted using the publicly available LibXtract [Bul07] library. In [P5], the MD PSO algorithm (with FGBF and SA) was able to find the (near-)optimal synthesis parameters from the defined parameter space. In some of the performed experiments, the synthesized features improved the retrieval precisions by over 10% with consecutive EFS runs. In the case of classification, the classification errors were improved when comparing to those obtained with an SVM classifier with several different parameter settings. Especially with the "audio statistics" feature set, the achieved improvement was notable. In [P6], where the heterogeneous particle behaviour was applied with the MD PSO algorithm, clear improvements were achieved compared to the original low-level audio features. This improvement was also the main interest of the work, and not the absolute classification or retrieval performance as such. In other words, the purpose was to provide new and powerful (artificial) audio features, more than providing solutions for particular classification problems, such as music genre classification. This is why the selection of the audio class types may seem somewhat arbitrary if considered as a pure audio classification problem.

As shown by Tables 10 and 12 in [P6], in the 12-class case the results were better than those obtained using evolutionary artificial neural networks as feature synthesizers. This demonstrates the importance of applying several operators and feature selection to the synthesis process. However, the results achieved with a 6-class database shown in Table 10 suggest (when compared to Table 12) that when it comes to discriminating features of a relatively small and distinguishable database, the EFS technique may not provide much improvement. Nonetheless, as discovered from Table 11, in most cases the EFS technique was able to provide significant retrieval improvements with an additional property of finding also the optimal output vector dimensionality, which is rarely considered in the feature generation literature in general. It should be also noted that the results shown in Table 9 in [P6] should not be confused to the actual performance of the EFS method, as their purpose was to act as a reference for the performance improvement achieved once all the operators and the dynamic output vector dimensionality were enabled (in Table 10). Finally, another important foundation was the performance level achieved with the baseline MFCC features, which was only beaten after applying several consecutive EFS runs. This indicates the efficiency of these features and reveals that once modified, the effective outcome is more likely to be negative than positive.

In a sense, all the optimization problems considered in this thesis can be though of as inverse problems. This is most obvious in the case of sniper positioning, in which the observed shock waves represent the "desired" output for the applied shock wave model. This desired output is then applied to adjust the underlying model parameters through using the particle filtering and simulated annealing approaches. In the classifier parameter optimization, the ground truth

class labels of the training data represent the desired output, and the similar logic applies also with the binary target vector - based feature synthesis approach. Thus, although the actual application areas are rather different from each other, the stochastic optimization algorithms presented in this thesis were proven applicable to them with fundamentally similar approaches. This demonstrates the high range of applicability of such methods to various application areas and optimization tasks.

6.2 Future Work

Excluding the few discussions and verbal comparisons, a significant branch of the stochastic optimization algorithms family, the genetic algorithms, were not considered in this thesis. The potential of these algorithms, e.g. the mutation property, could be researched more in the sense of utilizing the advantages of these algorithms together with those already considered in the work. Another way of proceeding could include performing and studying analytical parameter setting for the optimization algorithms. Experimental parameter adjustment is often time consuming and non-optimal, not to mention the "wasted" computational effort in performing several experimental optimization processes. As an algorithmic development, combining the CNBC framework with the EFS technique is a potential way to proceed with the work, as this would enable optimizing both the structures/parameters of the applied classifiers and the features themselves simultaneously. Also a recent and emerging area of machine learning called *deep learning* [Are10] should be included to the research in the future.

Considering the applications presented in this thesis, new gunshot recordings with calibrated microphones could be performed to allow including the absolute sound pressure information to the data. This would provide valuable information regarding to estimating the bullet miss distances. Furthermore, this could give additional performance enhancement considering a real prototype, where a soldier helmet is used as a microphone array. Although no real prototype helmet was applied in the published studies, the made research forms a solid ground for potential future attempts in bringing such a sniper detection system to life. With respect to content-based classification, new perceptually motivated audio features and several types of classifiers could be applied to the CNBC framework. The EFS technique could be experimented with other types of fitness functions and databases, and the list of available operators could be optimized or updated during the synthesis process based on the performed operator selections. At least in theory, the application of each operator could be also expanded to several features at a time, instead of the feature pair - based approach mimicked from artificial neural networks. Finally, the considered directional and contextual audio estimation approaches could be combined in a novel and natural way to bring more value and new experiences for the end user.

Bibliography

[Abb03] H. Abbass, Pareto neuro-evolution: Constructing ensemble of neural networks using multi-objective optimization. In *the 2003 Congress on Evolutionary Computation*, vol. 3, pp. 2074–2080, 2003, Canberra, ACT, Australia.

[Alp10] E. Alpaydin, *Introduction to Machine Learning*. Adaptive Computation and Machine Learning, MIT Press, 2010, 2nd edition.

[Are10] I. Arel, D. Rose, and T. Karnowski, Deep machine learning - a new frontier in artificial intelligence research. *IEEE Computational Intelligence Magazine*, 5(4):pp. 13–18, November 2010.

[Ars99] G. Arslan, F. Sakarya, and B. Evans, Speaker localization for far-field and near-field wideband sources using neural networks. In *Proceedings of the IEEE Workshop on Nonlinear Signal and Image Processing (NSIP)*, pp. 528–532, June 1999.

[Aru02] M. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking. *IEEE Transactions on Signal Processing*, 50(2):pp. 174–188, February 2002.

[Ash05] J. Ash and R. Moses, Acoustic time delay estimation and sensor network self-localization: Experimental results. *J. Acoust. Soc. Am.*, 118(2):pp. 841–850, May 2005.

[Bai03] W. Bainbridge, W. Toms, D. Edwards, and S. Furber, Delay-insensitive, point-to-point interconnect using M-of-N codes. In *Proceedings of the 9th International Symposium of Asynchronous Circuits and Systems*, pp. 132–140, May 2003, Vancouver, BC, Canada.

[Bal10] T. Ballal and C. Bleakley, DOA estimation for a multi-frequency signal using widely-spaced sensors. In *Proceedings of the 18th European Signal Processing Conference (EUSIPCO)*, pp. 691–695, August 2010.

[Bar08] J. Barger, S. Milligan, M. Brinn, and R. Mullen, Systems and methods for determining shooter locations with weak muzzle detection. United States Patent US 7359285, BBN Technologies Corp. Solutions LLC, April 2008.

[BBN09] BBN Technologies, Boomerang. 2009, URL `http://boomerang.bbn.com/`, read 6–January–2013.

[Bel61] R. Bellman, *Adaptive control processes: a guided tour*. Rand Corporation Research studies, Princeton University Press, 1961.

[Ben08] J. Benesty, J. Chen, and Y. Huang, *Microphone Array Signal Processing*. Springer Topics in Signal Processing, Springer-Verlag, 2008.

[Ber04] B. Berg, *Markov Chain Monte Carlo Simulations and Their Statistical Analysis: With Web-based Fortran Code*. World Scientific, 2004.

[Bis06] C. Bishop, *Pattern Recognition and Machnine Learning*. Information Science and Statistics, Springer Science+Business Media, 2006.

[Bul07] J. Bullock, Libxtract: A lightweight library for audio feature extraction. In *Proceedings of the International Computer Music Conference*, pp. 25–28, 2007.

[Can11] J. Candy, *Bayesian Signal Processing: Classical, Modern and Particle Filtering Methods*. Adaptive and Learning Systems for Signal Processing, Communications and Control, John Wiley & Sons, 2011.

[Cha95] Y. Chauvin and D. Rumelhart, *Backpropagation: Theory, Architectures, and Applications*. Developments in Connectionist Theory, Lawrence Erlbaum Associates, Inc., 1995.

[Che01] T. Chen and H. Wu, Adaptive impulse detection using center-weighted median filters. *IEEE Signal Processing Letters*, 8(1):pp. 1–3, January 2001.

[Che06] L. Chen, S. Gündüz, and M. Özsu, Mixed type audio classification with support vector machine. In *Proceedings of the IEEE International Conference on Multimedia and Expo*, pp. 781–784, July 2006.

[Chu08] L. Chuang, H. Chang, C. Tu, and C. Yang, Improved binary pso for feature selection using gene expression data. *Computational Biology and Chemistry*, 32(1):pp. 29–37, February 2008.

[Dan06] E. Danicki, The shock wave-based acoustic sniper localization. *Nonlinear Analysis: Theory, Methods & Applications*, 65(5):pp. 956–962, September 2006.

[Dav06] J. Davis and M. Goadrich, The relationship between precision-recall and ROC curves. In *Proceedings of the 23rd international conference on Machine learning (ICML '06)*, pp. 233–240, 2006.

[Die94] T. Dietterich and G. Bakiri, Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, 2(1):pp. 263–286, 1994.

[Dog09] E. Dogan, M. Sert, and A. Yazici, Content-based classification and segmentation of mixed-type audio by using MPEG-7 features. In *First International Conference on Advances in Multimedia*, pp. 152–157, July 2009.

[Duc97] G. Duckworth, D. Gilbert, and J. Barger, Acoustic counter-sniper system. In *Proceedings of SPIE, Command, Control, Communications, and Intelligence Systems for Law Enforcement*, vol. 2938, pp. 262–275, February 1997.

[Dud01] R. Duda, P. Hart, and D. Stork, *Pattern classification*. Pattern Classification and Scene Analysis: Pattern Classification, Wiley, 2001, 2nd edition.

[Duf00] A. Dufaux, L. Besacier, M. Ansorge, and F. Pellandini, Automatic sound detection and recognition for noisy environment. In *Proceedings of IEEE European Signal Processing Conference (EUSIPCO) X*, pp. 1033–1036, September 2000.

[Dun89] G. Dunteman, *Principal Component Analysis*. Sage University Paper series on Quantitative Applications in the Social Sciences, SAGE, 1989.

[Ebe98] R. Eberhart and Y. Shi, Comparison between genetic algorithms and particle swarm optimization. In *Evolutionary Programming VII*, vol. 1447 of *Lecture Notes in Computer Science*, pp. 611–616, 1998.

[Eng10] A. Engelbrecht, Heterogeneous particle swarm optimization. In *Proceedings of 7th International Conference on Swarm Intelligence*, pp. 191–202, September 2010, Berlin, Germany.

[Ero09] A. Eronen, *Signal Processing Methods for Audio Classification and Music Content Analysis*. Ph.D. thesis, Tampere University of Technology, 2009.

[Fer07] B. Ferguson, K. Lo, and R. Wyber, Acoustic sensing of direct and indirect weapon fire. In *the 3rd International Conference on Intelligent Sensors, Sensor Networks and Information*, pp. 167–172, December 2007, Sydney, Australia.

[Fre99] R. French, Catastrophic forgetting in connectionist networks: Causes, consequences and solutions. *Trends in Cognitive Sciences*, 3(4):pp. 128–135, April 1999.

[Gol89] D. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Artificial Intelligence, Addison-Wesley Longman Publishing Co., 1989.

[Gri97] C. Grinstead and J. Snell, *Introduction to probability*. 2nd ed, American Mathematical Society, 1997.

[Guo03] G. Guo and S. Li, Content-based audio classification and retrieval by support vector machines. *IEEE Transactions on Neural Networks*, 14(1):pp. 209–215, 2003.

[Gur97] K. Gurney, *An Introduction to Neural Networks*. UCL Press, 1997.

[Guy03] I. Guyon and A. Elisseeff, An introduction to variable and feature selection. *The Journal of Machine Learning Research*, 3:pp. 1157–1182, January 2003.

[Han11] J. Han, M. Kamber, and J. Pei, *Data Mining: Concepts and Techniques*. The Morgan Kaufmann Series in Data Management Systems, Elsevier, 2011.

[Har07] H. Harb and L. Chen, A general audio classifier based on human perception motivated model. *Multimedia Tools and Applications*, 34(3):pp. 375–395, September 2007.

[Hei10] T. Heittola, A. Mesaros, A. Eronen, and T. Virtanen, Audio context recognition using audio event histograms. In *Proceedings of the 18th European Signal Processing Conference (EUSIPCO)*, pp. 1272–1276, August 2010, Aalborg, Denmark.

[Hel10] M. Helén and T. Virtanen, Audio query by example using similarity measures between probability density functions of features. *EURASIP Journal on Audio, Speech, and Music Processing*, 2010(1):pp. 1–12, 2010.

[Hen07] S. Hengy, S. Demezzo, and P. Hamery, Sniper detection using a helmet array: First tests in urban environment. In *SPIE Defense and Security Symposium*, April 2007, Orlando, US.

[Hin99] G. Hinton and T. Sejnowski, (Eds.) *Unsupervised Learning - Foundations of Neural Computation*. Computational Neuroscience series, Massachusetts Institute of Technology (MIT) Press, 1999.

[Hsu02] C. Hsu and C. Lin, A comparison of methods for multiclass support vector machines. *IEEE Transactions on Neural Networks*, 13(2):pp. 415–425, 2002.

[Hua06] C. Huang and C. Wang, A GA-based feature selection and parameters optimization for support vector machines. *Expert Systems with Applications*, 31(2):pp. 231–240, August 2006.

[ISO05] Acoustics - Noise from shooting ranges - Part 1: Determination of muzzle blast by measurement (ISO 17201-1:2005(e)). 2005.

[Kal60] R. Kalman, A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):pp. 35–45, 1960.

[Kal09] M. Kalos and P. Whitlock, *Monte Carlo Methods - Second, Revised and Enlarged Edition*. John Wiley & Sons, 2009.

[Ken95] J. Kennedy and R. Eberhart, Particle swarm optimization. In *Proceedings of IEEE International Conference on Neural Networks*, vol. 4, pp. 1942–1948, November/December 1995.

[Ken03] J. Kennedy, Bare bones particle swarms. In *Proceedings of the 2003 IEEE Swarm Intelligence Symposium (SIS)*, pp. 80–87, April 2003, Washington DC, USA.

[Kir83] S. Kirkpatrick, C. Gelatt, and P. Vecchi, Optimization by simulated annealing. *Science, New Series*, 220(4598):pp. 671–680, May 1983.

[Kir06a] S. Kiranyaz and M. Gabbouj, Generic content-based audio indexing and retrieval framework. *IEE Proceedings - Vision, Image and Signal Processing*, 153(3):pp. 285–297, June 2006.

[Kir06b] S. Kiranyaz, A. Qureshi, and M. Gabbouj, A generic audio classification and segmentation approach for multimedia indexing and retrieval. *IEEE Transactions on Audio, Speech, and Language Processing*, 14(3):pp. 1062–1081, May 2006.

[Kir09] S. Kiranyaz, T. Ince, A. Yildirim, and M. Gabbouj, Evolutionary artificial neural networks by multi-dimensional particle swarm optimization. *Neural Networks*, 22(10):pp. 1448–1462, December 2009.

[Kir10] S. Kiranyaz, T. Ince, A. Yildirim, and M. Gabbouj, Fractional particle swarm optimization in multi-dimensional search space. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 40(2):pp. 298–319, April 2010.

[Kir11] S. Kiranyaz, J. Pulkkinen, T. Ince, and M. Gabbouj, Multi-dimensional evolutionary feature synthesis for content-based image retrieval. In *Proceedings of the 18th IEEE International Conference on Image Processing (ICIP)*, pp. 3645–3648, 2011.

[Kir12] S. Kiranyaz and M. Gabbouj, *Content-Based Management of Multimedia Databases: Advanced Techniques for Multimedia Analysis and Retrieval*. Lambert Academic Publishing, 2012.

[Kla06] A. Klapuri and M. Davy, *Signal Processing Methods for Music Transcription*. Springer Science+Business Media LLC, 2006.

[Kna76] C. Knapp and G. Carter, The generalized correlation method for estimation of time delay. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 24(4):pp. 320–327, August 1976.

[Koz92] J. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. Complex Adaptive Systems series, MIT Press, 1992.

[Kri08] M. Kristan, D. Skočaj, and A. Leonardis, Incremental learning with gaussian mixture models. In *Proceedings of the Computer Vision Winter Workshop*, pp. 25–32, February 2008.

[Léd05] A. Lédeczi, A. Nádas, P. Völgyesi, G. Balogh, B. Kusy, J. Sallai, G. Pap, S. Dóra, K. Molnár, M. Maróti, and G. Simon, Countersniper system for urban warfare. *ACM Transactions on Sensor Networks (TOSN)*, 1(2):pp. 153–177, November 2005.

[Li00] S. Li, Content-based classification and retrieval of audio using the nearest feature line mathod. *IEEE Transactions on Speech and Audio Processing*, 8(5):pp. 619–625, September 2000.

[Lin05] Y. Lin and B. Bhanu, Evolutionary feature synthesis for object recognition. *IEEE Transactions on Systems, Man, and Cybernetics - Part C: Applications and Reviews*, 35(2):pp. 156–171, May 2005.

[Lin08] S. Lin, K. Ying, S. Chen, and Z. Lee, Particle swarm optimization for parameter determination and feature selection of support vector machines. *Expert Systems with Applications*, 35(4):pp. 1817–1824, November 2008.

[Lin12] M. Lin, J. Tsai, and C. Yu, A review of deterministic optimization methods in engineering and management. *Mathematical Problems in Engineering*, 2012:pp. 1–15, April 2012.

[Liu04] C. Liu, H. Sako, and H. Fujisawa, Effects of classifier structures and training regimes on integrated segmentation and recognition of handwritten numeral strings. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(11):pp. 1395–1407, November 2004.

[Liu08] H. Liu and H. Motoda, *Computational Methods of Feature Selection*. Chapman & Hall/CRC Data Mining and Knowledge Discovery Series, Taylor & Francis Group, 2008.

[Liu10] W. Liu and S. Weiss, *Wideband Beamforming: Concepts and Techniques*. Wireless Communications and Mobile Computing, John Wiley & Sons, 2010.

[Liu12] Q. Liu, *Wideband Digital Filter-and-Sum Beamforming with Simultaneous Correction of Dispersive Cable and Antenna Effects*. Ph.D. thesis, Faculty of the Virginia Polytechnic Institute and State University, 2012.

[Mah06] R. Maher, Modeling and signal processing of acoustic gunshot recordings. In *Proceedings of the IEEE Signal Processing Society 12th DSP Workshop*, pp. 257–261, September 2006, Jackson Lake, WY.

[Mäk08] T. Mäkinen, *Detection and direction estimation of impulsive sound sources using a microphone array*. Master's thesis, Tampere University of Technology, 2008.

[Man02] B. Manjunath, P. Salembier, and T. Sikora, *Introduction to MPEG-7: Multimedia Content Description Interface*. Wiley, 2002.

[Man10] M. Mandel, S. Bressler, B. Shinn-Cunningham, and D. Ellis, Evaluating source separation algorithm with reverberant speech. *IEEE Transactions on Audio, Speech, and Language Processing*, 18(7):pp. 1872–1883, September 2010.

[Mar02] S. Markovitch and D. Rosenstein, Feature generation using general constructor functions. *Machine Learning*, 49(1):pp. 59–98, October 2002.

[Mar10] R. Marxer and H. Purwins, Unsupervised incremental learning and prediction of audio signals. In *Proceedings of the International Symposium on Music Acoustics (ISMA)*, August 2010.

[McL04] G. McLachlan, *Discriminant Analysis and Statistical Pattern Recognition*. Wiley Series in Probability and Mathematical Statistics, John Wiley & Sons, 2004.

[McN93] N. McNelis and N. Conner, Methods and apparatus for determining the trajectory of a supersonic projectile. United States Patent 5241518, AAI Corporation, August 1993.

[Mie05] I. Mierswa and K. Morik, Automatic feature extraction for classifying audio data. *Machine Learning*, 58(2–3):pp. 127–149, February 2005.

[Mik08] S. Mikki and A. Kishk, *Particle Swarm Optimization: A Physics-Based Approach*. Syntheis Lectures on Computational Electromagnetic Series, Morgan & Claypool Publishers, 2008.

[Mit08] V. Mitra and C. Wang, Content based audio classification: a neural network approach. *Soft Computing - A Fusion of Foundations, Methodologies and Applications - Special issue on neural networks for pattern recognition and data mining*, 12(7):pp. 639–646, February 2008.

[Moe06] F. Moerchen, I. Mierswa, and A. Ultsch, Understandable models of music collections based on exhaustive feature generation with temporal statistics. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 882–891, 2006, philadelphia, Pa, USA.

[Mon04] S. Montebugnoli, G. Bianchi, A. Cattani, F. Ghelfi, A. Maccaferri, and F. Perini, Some notes on beamforming. Tech. Rep. IRA N. 353/04, INAF - Istituto di Radioastronomia, 2004, the Medicina IRA-SKA Engineering Group.

[Mör10] P. Mörters and Y. Peres, *Brownian Motion*. Cambridge Series in Statistical and Probabilistic Mathematics, Cambridge University Press, 2010.

[Mot95] R. Motwani and P. Raghavan, *Randomized algorithms*. Cambridge University Press, 1995.

[Nas06] C. Nash, *Sniper Detection and Weapon Classification Using Shockwave and Muzzle Blast Transients*. Vanderbilt University, 2006.

[Nea93] R. Neal, Probabilistic inference using markov chain monte carlo methods. Tech. rep., Department of Computer Science, University of Toronto, September 1993.

[Nil98] N. Nilsson, *Introduction to Machine Learning*. 1998, URL `http://robotics.stanford.edu/~nilsson/MLBOOK.pdf`, An early draft of a proposed text book.

[Pac03] F. Pachet and A. Zils, Evolving automatically high-level music descriptors from acoustic signals. In *the 1st International Symposium on Computer Music Modeling and Retrieval*, vol. 2771 of *Lecture Notes in Computer Science*, pp. 42–53, Springer-Verlag LNCS, 2003.

[Pac09] F. Pachet and P. Roy, Analytical features: a knowledge-based approach to audio feature generation. *EURASIP Journal on Audio, Speech, and Music Processing*, 2009:pp. 1–23, 2009.

[Par12] I. Paraskevas and M. Rangoussi, Feature extraction for audio classification of gunshots using the Hartley transform. *Open Journal of Acoustics*, 2(3):pp. 131–142, September 2012.

[Pav00] Y. Pavlov, *Random Forests*. VSP, 2000.

[Pee07] G. Peeters, A generic system for audio indexing: application to speech/music segmentation and music genre recognition. In *Proceedings of the 10th Conference on Digital Audio Effects (DAFx)*, September 2007, Bordeaux, France.

[Per09] P. Pertilä, *Acoustic Source Localization in a Room Environment and at Moderate Distances*. Ph.D. thesis, Tampere University of Technology, 2009.

[Pol01] R. Polikar, L. Upda, S. Upda, and V. Honavar, Learn++: an incremental learning algorithm for supervised neural networks. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 31(4):pp. 497–508, November 2001.

[Pol06] R. Polikar, Ensemble based systems in decision making. *IEEE Circuits and Systems Magazine*, 6(3):pp. 21–45, 2006.

[Pri05] K. Priddy and P. Keller, *Artificial Neural Networks: An Introduction*. Tutorial Texts in Optical Engineering, SPIE Press, 2005.

[PS91] G. Piatetsky-Shapiro, *Knowledge discovery in databases*. AAAI Press Series, AAAI Press, 1991.

[Qua08] T. Quatieri, *Discrete-Time Speech Signal Processing: Principles and Practice*. Pearson Education, 2008.

[Rab93] L. Rabiner and B. Juang, *Fundamentals of Speech Recognition*. Prentice Hall, 1993.

[Rab07] L. Rabiner and R. Schafer, Introduction to digital speech processing. *Foundations and Trends in Signal Processing*, 1(1–2):pp. 1–194, 2007.

[Ram09] R. Ramadan and R. Abdel-Kader, Face recognition using particle swarm optimization-based selected features. *International Journal of Signal Processing, Image Processing and Pattern Recognition*, 2(2):pp. 51–66, 2009.

[Rav10] E. Ravelli, G. Richard, and L. Daudet, Audio signal representations for indexing in the transform domain. *IEEE Transactions on Audio, Speech, and Language Processing*, 18(3):pp. 434–446, 2010.

[Rit02] O. Ritthoff, R. Klinkenberg, S. Fischer, and I. Mierswa, A hybrid approach to feature selection and generation using an evolutionary algorithm. In *Proceedings of the UK Workshop on Computational Intelligence*, pp. 147–154, September 2002, Birmingham, UK.

[Rum12] F. Rumsey, *Spatial Audio*. Music Technology Series, Taylor & Francis, 2012.

[Sad98] B. Sadler, T. Pham, and L. Sadler, Optimal and wavelet-based shock wave detection and estimation. *Journal of the Acoustical Society of America*, 104(2):pp. 955–963, August 1998.

[Sad06] B. Sadler and R. Kozick, A survey of time delay estimation performance bounds. In *Proceedings of the Fourth IEEE Workshop on Sensor Array and Multichannel Processing*, pp. 282–288, July 2006.

[Sch77] C. Schmid, *Acoustic pattern recognition of musical instruments*. University of Washington., 1977.

[Shi98] Y. Shi and R. Eberhart, Parameter selection in particle swarm optimization. In V. Porto, N. Saravanan, D. Waagen, and A. Eiben, (Eds.) *Evolutionary Programming VII*, vol. 1447 of *Lecture notes in Computer Science*, pp. 591–600, Springer-Verlag London, UK, 1998.

[Shi05] X. Shi, Y. Liang, H. Lee, C. Lu, and L. Wang, An improved GA and a n ovel PSO-GA-based hybrid algorithm. *Information Processing Letters*, 93(5):pp. 255–261, March 2005.

[SST09] SST, ShotSpotter Gunshot Location System. 2009, URL `http://www.shotspotter.com/`, read 6–January–2013.

[Sto97] R. Stoughton, SAIC SENTINEL acoustic counter-sniper system. In *Proceedings of SPIE, Command, Control, Communications, and Intelligence Systems for Law Enforcement*, vol. 2938, pp. 276–284, February 1997.

[Tec] Techila Technologies Ltd, Techila grid. URL `http://www.techila.fi`.

[Tza02] G. Tzanetakis and P. Cook, Musical genre classification of audio signals. *IEEE Transactions on Speech and Audio Processing*, 10(5):pp. 293–302, July 2002.

[vL87] P. van Laarhoven and E. Aarts, *Simulated Annealing: Theory and Applications*. Mathematics and Its Applications, Springer, 1987.

[Vol07] P. Volgyesi, G. Balogh, A. Nadas, C. Nash, and A. Ledeczi, Shooter localization and weapon classification with soldier-wearable networked sensors. In *Proceedings of the 5th International Conference on Mobile Systems, Applications and Services*, pp. 113–126, 2007.

[Wan05] L. Wang, (Ed.) *Support Vector Machines: Theory and Applications*. Studies in Fuzziness and Soft Computing, Springer-Verlag Berlin Heidelberg, 2005.

[Wan06] Q. Wang, L. Xie, J. Liu, and Z. Xiang, Enhancing particle swarm optimization based particle filter tracker. In *Proceedings of the 2006 International Conference on Intelligent Computing: Part II*, vol. 4114 of *Lecture Notes in Computational Science*, pp. 1216–1221, 2006.

[Wei84] E. Weinstein and A. Weiss, Fundamental limitations in passive time-delay estimation — part II: Wide-band systems. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 32(5):pp. 1064–1078, October 1984.

[Wet04] M. Wetter and J. Wright, A comparison of deterministic and probabilistic optimization algorithms for nonsmooth simulation-based optimization. *Building and Environment*, 39(8):pp. 989–999, August 2004.

[Whi52] G. Whitman, The flow pattern of a supersonic projectile. *Communications on Pure and Applied Mathematics*, 5(3):pp. 301–348, August 1952.

[Wic10] G. Wichern, J. Xue, H. Thornburg, B. Mechtley, and A. Spanias, Segmentation, indexing, and retrieval for environmental and natural sounds. *IEEE Transactions on Audio, Speech, and Language Processing*, 18(3):pp. 688–707, March 2010.

[Wol96] E. Wold, T. Blum, D. Keislar, and J. Wheaton, Content-based classification, search, and retrieval of audio. *IEEE Multimedia*, 3(3):pp. 27–36, September 1996.

[Wor01] World Guns, Modern firearms – sniper rifles. 2001, URL `http://world.guns.ru/sniper`, read 4–April–2013.

[Ye04] J. Ye, *Speech Recognition Using Time Domain Features from Phase Space Reconstructions*. Master's thesis, Marquette University, 2004.

[YH96] J. Yli-Hietanen, K. Kalliojärvi, and J. Astola, Low-complexity angle of arrival estimation of wideband signals using small arrays. In *Proceedings of the 8th IEEE Signal Processing Workshop on Statistical Signal and Array Processing*, pp. 109–112, June 1996.

[Zha05] F. Zhao, Q. Zhang, D. Yu, X. Chen, and Y. Yang, A hybrid algorithm based on PSO and simulated annealing and its applications for partner selection in virtual enterprise. In *Advances in Intelligent Computing*, vol. 3644 of *Lecture Notes in Computer Science*, pp. 380–389, 2005.

[Zha08] X. Zhang, W. Hu, S. Maybank, X. Li, and M. Zhu, Sequential particle swarm optimization for visual tracking. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1–8, June 2008.

[Zhu07] Y. Zhu, Z. Ming, and Q. Huang, SVM-based audio classification for content-based multimedia retrieval. In *Proceedings of the 2007 International Conference on Multimedia Content Analysis and Mining (MCAM'07)*, pp. 474–482, 2007.

[Zhu08] L. Zhuo, J. Zheng, F. Wang, X. Li, B. Ai, and J. Qian, A genetic algorithm based wrapper feature selection method for classification of hyperspectral images using support vector machine. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XXXVII(B7):pp. 397–402, 2008.

Errata and Clarifications for the Publications

• Publication [P1]: The applied microphone array is the same one that is used and defined in publication [P2].

• Publication [P3]: In Introduction, the claim that in unsupervised learning techniques the number of classes needs to be fixed a priori is not always true. For example, in [Mar10] the number of sound classes can incrementally change.

• Publication [P3]: In Introduction, the sentence beginning with "Rather high training dataset ..." should be corrected as "Relatively high percent of the entire database samples (70%) were used as training data in achieving the results, whereas ...".

• Publication [P4]: In Introduction, after the statement beginning with "Support for any dynamic update...", it should be mentioned that some studies of Gaussian mixture models (GMMs), such as [Kri08], do support dynamic adaptation to data updates (on-line learning).

• Publication [P4]: In Section 2.2.4, the first sentence should state: "Sub-band centroid (SC) frequency is the first moment of the band spectral distribution (spectrum), and it can be estimated as the balancing frequency value for the absolute spectral values."

• Publication [P4]: In Conclusions, the first sentence should not claim the novelty of the CNBC framework itself, but the application of it to the domain of audio classification.

• Publication [P6]: At the beginning of the "Related work" section, the term "pattern recognition" should be used instead of "machine learning".

- Publication [P6]: The equation (1) should be

$$y_p^{d_p(t)}(t+1) = \begin{cases} y_p^{d_p(t)}(t), & \text{if } \mathcal{F}(x_p^{d_p(t)}(t)) > \mathcal{F}(y_p^{d_p(t)}(t)) \\ x_p^{d_p(t)}(t), & \text{else.} \end{cases} \tag{1}$$

Publication P1

Toni Mäkinen, Pasi Pertilä, and Pasi Auranen, Supersonic bullet state estimation using particle filtering. *Proceedings of the IEEE International Conference on Signal and Image Processing Applications (ICSIPA 09)*, pages 150 – 155, Kuala Lumpur, Malaysia, November 2009.

Supersonic Bullet State Estimation Using Particle Filtering

Toni Mäkinen [#1], Pasi Pertilä [#2], and Pasi Auranen [*3]

[#] *Department of Signal Processing, Tampere University of Technology*
P.O. Box 553 Tampere, Finland
[1,2] {toni.makinen, pasi.pertila}@tut.fi

[*] *Savox Communications Ltd*
Vitikka 4 02630 Espoo, Finland
[3] pasi.auranen@savox.com

***Abstract*—Due to an increasing number of sniper attacks in different crises and security threats around the world, there is a need for new technologies and applications to take place in helping to prepare against such offensives [1]. Estimation of sound wave direction of arrival (DOA) based on time differences between separate microphones is typically applied for sound source localization, and the existing research achievements of the field are utilized in the presented study. In this paper, a new method for estimating the *state* of a supersonic bullet is proposed. State is defined here to consist of bullet's trajectory, caliber, and speed. The method is based on a mathematical modeling of the bullet shock wave, and the parameter estimation procedure is built over the *Bayesian* inference. Both simulations and real shooting data are used to test and verify the performance of the proposed method. Bringing shock wave modeling and Bayesian inference together is the main focus of the study.**

I. Introduction

In security and peacekeeping it is vital to efficiently estimate the direction of a hostile shooter. Studies have been made in the field of bullet shock waves and trajectory estimation [2], [3], and [4]. In these papers, the mathematical modeling of a shooting event is identical, whereas the methods for detecting waveform signatures and the algorithms for sound source localization differs. Some anti-sniper systems ([5], [6]) are already being used in certain crisis situations, but there still exists a need for improving the reliability of the estimation.

This paper applies a mathematical model of the acoustic waveform of a bullet, that has been studied and measured by Whitman [7]. The previous studies exploit some basic principles of the shock wave signature in the estimation. In contrast, this work applies a *Bayesian approach* to bullet state estimation based on measured data and the mathematical model [7]. The bullet trajectory properties, as well as the caliber and speed, are enclosed by the mentioned bullet state, which is estimated with an inverse Bayesian method. Bayesian method gives an optimal solution in a case where prior knowledge exists. The state estimation problem is of high dimension and the underlying likelihood distribution is highly irregular. *Particle filters* are suitable for such problems [8], and are therefore applied in this work.

Acoustical bullet trajectory estimation methods can be divided into three categories: 1) methods using shock waves, 2) methods using the *muzzle blast* caused by the gun itself, and 3) methods using both of the features. The methods belonging to category 3 have more uncertainty involved in the estimation, since the number of error sources grows higher. Silencers and long shooting distances can prohibit the existence of muzzle blast, causing the category 2 methods to become useless. Nevertheless, they are used in some restless environments in the USA [9]. The proposed method here belongs to category 1, meaning that no observation of the muzzle blast is needed. However, since subsonic bullets do not cause shock waves, only the supersonic bullet states can be estimated. The category 1 methods are still suitable against snipers, who generally prefer rifles yielding supersonic bullets in order to ensure accuracy.

The structure of this paper is the following: First, in Section II the mathematical modeling of a shock wave is reviewed. The following Section III covers the main contribution of the paper: the use of particle filtering to solve the inference problems regarding to the state estimation. In Sections IV and V, the testing and performance of the developed system is covered, and finally conclusions and future work is presented in the last Section VI.

II. Modeling of a Shock Wave

As a bullet propagates in a homogeneous medium at supersonic speed, it creates omni-directionally propagating sound waves, which together form an acoustical shock wave front. As a result, a cone-shaped pattern is formed behind the bullet, see Fig. 1. The angle of the shock wave front with respect to the bullet's trajectory is proportional to the *speed* of the projectile [10]:

$$\theta_M = \arcsin\left(\frac{1}{M}\right), \tag{1}$$

where M is a *Mach number*, which is defined as $M = v/c$, where v and c are the speed of the projectile and sound, respectively.

The waveform of a shock wave is a function of projectile's *diameter* ϕ, *speed* v, and the distance between the trajectory and the receiving microphone position $\mathbf{m}_i$. This *miss distance*

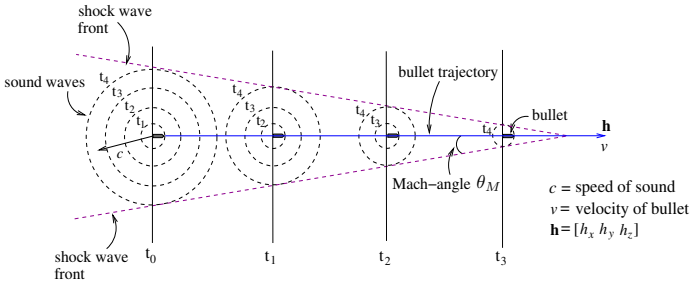

Fig. 1. As a bullet propagates from left to right, a shock wave cone is formed behind the bullet, marked with a purple dashed line. The shock wave front is propagating at the speed of sound c, while the bullet has a decreasing speed, approximated here as a constant v. Vector $\mathbf{h}$ determines the *heading* of the bullet trajectory, whereas the different time indexes are marked by t_n, where n is running from 0 to 4.

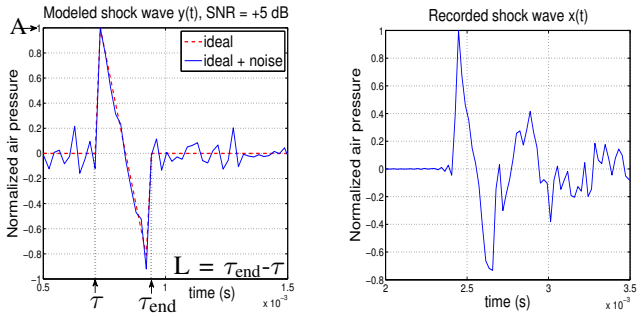

Fig. 2. Modeled and recorded shock waves and the necessary parameters, τ, A and L, for the modeling. Red dashed line on the left image represents an ideal shock wave with zero noise level.

is measured from the trajectory's *CPA-point (Closest Point of Approach)*, marked here as $\mathbf{a}$. A line drawn from this point to the microphone position $\mathbf{m}_i$ lies always perpendicular to the trajectory. The shock wave's waveform is mathematically defined using (2) and (3), which form the Whitman model: [7]

$$A_i = \frac{0.53P_0(M^2-1)^{1/8}\phi}{d_{\mathbf{m}_i,\mathbf{a}}^{3/4}l^{1/4}}\ [\text{Pa}], \tag{2}$$

$$L_i = \frac{1.82Md_{\mathbf{m}_i,\mathbf{a}}^{1/4}\phi}{c(M^2-1)^{3/8}l^{1/4}}\ [\text{m}], \tag{3}$$

where P_0 is the atmospheric air pressure, $d_{\mathbf{m}_i,\mathbf{a}}$ is the miss distance, and l is the length of the bullet, which is related to its diameter by [3]

$$l \approx 4.35\phi\ [\text{m}]. \tag{4}$$

In addition, the shock wave's time of arrival τ_i to a given microphone $\mathbf{m}_i$ from a specific point $\mathbf{s}$ of the trajectory is needed to model a shock wave caused by a bullet. The term τ_i is defined as

$$\tau_i = \frac{||\mathbf{m}_i - \mathbf{s}||}{c} = \frac{d_{\mathbf{m}_i,\mathbf{s}}}{c}\ [\text{s}], \tag{5}$$

where $d_{\mathbf{m}_i,\mathbf{s}}$ is the distance between the microphone $\mathbf{m}_i$ and the *S-point* $\mathbf{s}$, which is defined in the next Section II-A.

The actual shock wave's waveform is commonly known as *N-shaped* wave (or simply N-wave). This is because the time domain waveform has very dramatic rising and falling edges, which are literally making it to look like a letter "N". This kind of a function can be formed by as is shown in (6), where $\tilde{A}_i$ represents the normalized version of A_i of (2) [2]:

$$y_i(t) = \begin{cases} \tilde{A}_i\left(1-2\frac{t-\tau_i}{L}\right), & \tau_i \le t \le \tau_i + L_i \\ 0, & \text{otherwise.} \end{cases} \tag{6}$$

Obviously, the amplitude of the shock wave increases as the miss distance decreases. Also the shock wave length is varying as a function of distance to the trajectory. This can be seen from (3), where the waveform of the shock wave becomes longer, as the trajectory moves further away from the microphone. In Fig. 2, in the left image, an example of N-wave's waveform with all the necessary parameters can be seen. In order to compare the modeled signal with a real shock wave, also a waveform of a recorded shock wave signal is shown in the right image.

A. Shock Wave's Launching Point

The trajectory's CPA-point differs from the point where the bullet's shock wave is actually launched from, i.e. the S-point. In order to estimate the trajectory of a projectile, the derivation of the CPA-point using the S-point must be obtained — or vice versa. An illustration of the S-point with respect to the CPA is shown in Fig. 3. The blue line represents the line of fire, i.e. the trajectory to be estimated. Vector $\mathbf{g}$ is the *position vector* for the weapon, while vector $\mathbf{h}$ stands for the direction (heading) vector for the bullet. The position vectors are defined in Cartesian coordinates for a particular point, for example $\mathbf{g} = [g_x, g_y, g_z]^T$. *Mach angle* θ_M shown in Fig. 3 is estimated here between the shock wave front and the bullet trajectory. The Mach angle can vary significantly, depending on the speed

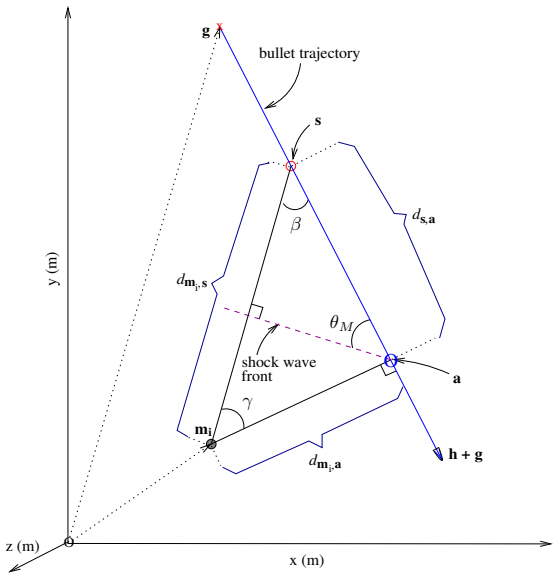

Fig. 3. The geometry of the shock wave front with respect to the CPA - point $\mathbf{a}$ and S-point $\mathbf{s}$. The view of the figure is from above.

of the bullet (1). For bullets that are only slightly faster than the speed of sound, θ_M is nearly $90°$, whereas for very fast bullets, say $v = 1000\ m/s$, θ_M can is as small as $\approx 20°$ [10]. The different angles and metrics of Fig. 3 are obtained by the means of geometry, as will be seen with the simulation in Section IV. As a resulting outcome, the relations between the S-point and CPA can be derived.

III. Estimation of the Bullet's Trajectory

It is assumed that a shock wave — caused by a certain type of a gun and a bullet — has been observed by an *array* of microphones. Furthermore, the coordinates for the microphones are assumed known. An unambiguous direction of arriving (DOA) shock wave can be then calculated, when at least *four* microphones have observed the shock wave. The DOA estimate for the shock wave is based on *time differences* between the shock wave observations of each microphone. With these assumptions, an idea of using *probabilistic inference methods* to estimate the bullet state is proposed.

In Sections III-A and III-B observed data is used to build a so-called *data model*, which is then used to draw conclusions about the unobserved quantities [11]. Here Bayes-methods are considered, since dealing with probability distributions instead of point estimates is a complete solution for the problem.

A. Estimating the Likelihood of a Trajectory

Likelihood function gives *probabilities* for different system outcomes, given that the observed outcome is formed according to some known parameters. It is possible to construct a likelihood function for estimating the probabilities for the observed shock wave *being caused* by a bullet with an arbitrary chosen trajectory, speed and caliber. Likelihood function values are here calculated using a *generalized cross correlation (GCC)* method. The GCC is calculated between the shock wave observations and the modeled shock waves based on (6). The correlation is calculated separately for all the microphones, and a so-called phase transform (PHAT) frequency weighting is also used with the GCC:

$$C_i(t) = \text{IFFT}\left(\frac{X_i Y_i^*}{|X_i||Y_i^*|}\right), \tag{7}$$

where IFFT stands for an inverse Fourier-transform, X_i and Y_i are the Fourier-transforms of the observed and generated shock waves $x_i(t)$ and $y_i(t)$ of microphone channel i, respectively, $(\cdot)^*$ stands for a complex conjugate, $|\cdot|$ is an absolute value, and $C_i(t)$ is the correlation function of microphone channel i at lag t, where t can have values $t \in (-\tau_{max}, \tau_{max})$, τ_{max} being the maximum delay between two microphones. The used phase transform makes the magnitude spectrums of the shock wave signals flat, so that only the phase information is used in calculating the correlation functions [12]. The likelihood value for a bullet state $\mathbf{p}$ is determined by

$$P(\mathbf{p}|\mathbf{O}) = \max\left(\prod_{i=1}^{n} C_i(t)\right), \quad t \in (-\tau_{max}, \tau_{max}) \tag{8}$$

where $\mathbf{O}$ is a $n \times N$ matrix containing the observed signals of length N by n different microphones, and $\mathbf{p}$ is a state vector containing the different parameter values:

$$\mathbf{p} = \left[\mathbf{a}^T,\ \phi,\ v\right]^T. \tag{9}$$

Other methods to determine the likelihood value could also be considered, such as *Mean Square Error (MSE)* between the observed and modeled signals, but here the best results were obtained by using GCC.

In Fig. 4 a outdoor shooting range is shown from above, and the likelihood function of a bullet CPA is plotted on the top by altering the CPA's coordinate. The data used to calculate the likelihood function is taken from the gunshot recordings, discussed in more detail in Section V. The blue areas in Fig. 4 are corresponding to low-probability regions, whereas the dark red color shows the position of the likelihood function's global maximum. The resulting function contains several local maximums and some clutter caused by some non-bullet objects and the background noise. This makes the use of gradient-based search methods difficult. Instead, the particle filters are studied more closely to solve the estimation problem.

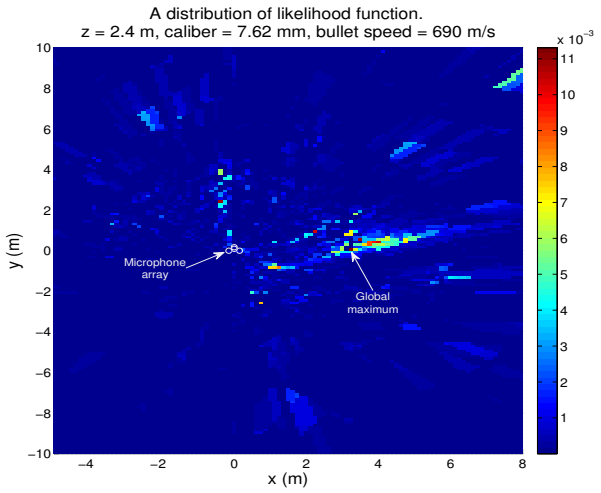

Fig. 4. Likelihood function (Eq. (8)) as a function of x- and y-coordinates of the CPA. The parameters z, ϕ and v are fixed as is described in the title. Four microphones are used in the microphone array.

B. Particle Filter

Particle filter consists of m particles $\mathbf{p}_j$ and weights $W(\mathbf{p}_j)$, where $j = 1, \ldots, m$. Particles represent a "state" of the system to be studied. Particle filters are also known as sequential *Monte Carlo-methods*, and they can be used to numerically estimate the *hidden parameters* or state of the system, based only on the observed data. This is often needed with the inference problems that are too complex to be solved analytically [8].

Since particle filtering is an approximation of Bayesian optimal solution, it follows the Bayes rule (see also (8)):

$$P(\mathbf{O}|\mathbf{p}) \propto P(\mathbf{p}|\mathbf{O})P(\mathbf{O}), \tag{10}$$

where $\propto$ stands for linear proportionality, and $P(\mathbf{O})$ means the *prior* probability distribution assigned to the observations.

Considering the state estimation problem, the functionality of particle filter is bounded to the following process: First, particles are generated to populate the state space based on the a priori information of each parameter: CPA-point is located at radius r from the array, bullet diameter is distributed according to common bullet calibers, and the velocity ranges upwards from the speed of sound. Second, the corresponding shock waves that the microphones *would observe* are generated based on the parameters. After the actual shock wave front has been detected by the used microphone array, i.e. the signals $x_i(t)$ have been received, the process continues by calculating the weight values $W(\mathbf{p}_j)$ for each particle $\mathbf{p}_j$ using (7) and (8). Note that the bullet shock wave must be detected. The detection methods vary, see e.g. [13] and [14] for details. Here the detection stage is omitted for the sake of brevity.

After all the particles are gone through, they are moved randomly in the state space by using here the *Brown's motion*. New weights are calculated and *Metropolis algorithm* is then utilized to update the particles [11]. If a new particle produces a higher weight than the previous one, the upgrading is certain to be done. However, movement towards the lower probability can also occur. The criteria for updating the particles can be mathematically derived by using a random variable α with a uniform distribution over the interval (0, 1). A new particle is then accepted if

$$P(\mathbf{p}_j^\star, \mathbf{p}_j^t) \geq \alpha, \tag{11}$$

where $\mathbf{p}_j^\star$ is the new candidate particle and $\mathbf{p}_j^t$ the particle of the current time index t. Updating is done with the probability

$$P(\mathbf{p}_j^\star, \mathbf{p}_j^t) = \min\left(1, \frac{W(\mathbf{p}_j^\star)}{W(\mathbf{p}_j^t)}\right), \tag{12}$$

where $W(\cdot)$ represents the weight of a corresponding particle, obtained from (8). Equation (12) also shows, that if the update from $\mathbf{p}_j^t$ to $\mathbf{p}_j^\star$ increases the weight, the candidate particle is certain to be kept, that is $\mathbf{p}_j^{t+1} = \mathbf{p}_j^\star$. Metropolis algorithm is a way to avoid converging into a local maximum, since the particles are allowed to move around and to search for the global maximum [11]. Also, in order to prevent the particle filter from converging into any local maximum, 5 % of the particles are randomly distributed again in the state space after each iteration round.

Another very essential concept, called *resampling*, has been developed to help the particles in converging near to the global maximum of the likelihood function. Resampling is used for replacing the particles with low weights with higher weighted ones. This is practically done to avoid the *degeneracy* problem, which can easily occur, if all but one of the particle have weights near zero. In this paper, the systematic resampling is used [8]. After the iterative particle moving, *median* of the resampled particles is taken to represent the output of the particle filter. The iterations are done in total for k times, as is seen in Alg. 1, where the main steps of the proposed procedure are roughly gone through. Because of the orthogonality, the obtained CPA contains all the necessary information to derive the bullet trajectory.

Algorithm 1: A rough algorithm for estimating the CPA.

Input: The observed shock wave by each n microphones
Output: Estimated CPA

Initialize the particles according to the prior knowledge.
for $iteration \leftarrow 1$ **to** k **do**
  for $particle \leftarrow 1$ **to** m **do**
    Derive S-point using the current CPA (sec. II-A).
    for *each microphone channel* i **do**
      Model the shock wave addressed by the current particle $\mathbf{p}_j$ (Eq. (2), (3), (5)).
      Calculate the cross correlation between the observed and modeled shock wave (Eq. (7)).
    Multiply the correlation results and take the maximum value (Eq. (8)).
    Update $\mathbf{p}_j$ using *Metropolis* algorithm (Eq. (12)).
  Resample $\mathbf{p}_{1:m}$ with the *resampling* algorithm.
$\mathbf{p}_{best} \leftarrow \text{median}\{\mathbf{p}_{1:m}\}$.
$\mathbf{a} \leftarrow \mathbf{p}_{\text{best}}(1:3)$.

IV. Simulations

For testing the capability of a particle filter to find the global maximum of a likelihood function, a simulation based on the theory of Section II was created. In the simulation, the positions of the gun and each microphone can be determined, along with the bullet's direction, speed and caliber. All of the parameters can be chosen freely, so that the testing environment can be considered as relatively diverse. The distances between the settled microphones and the trajectory are calculated by [15]

$$d_{\mathbf{m}_i,\mathbf{a}} = \frac{||(\mathbf{m_i} - \mathbf{g}) \times \mathbf{h}||}{||\mathbf{h}||}, \tag{13}$$

where $||\mathbf{h}||$ is the norm of vector $\mathbf{h}$, $\times$ represents the cross product of two vectors, and rest of the symbols are described in Section II. Considering again the metrics of Fig. 3, the relations between the CPA and the S-point can be found by solving all the geometric distances. The angle $\beta = 90° - \theta_M$, whereas the terms $d_{\mathbf{m}_i,\mathbf{s}}$ and $d_{\mathbf{s},\mathbf{a}}$ are found as is shown in (14) (using a sine rule):

$$d_{\mathbf{m}_i,\mathbf{s}} = \frac{d_{\mathbf{m}_i,\mathbf{a}}}{\sin\beta}, \qquad d_{\mathbf{s},\mathbf{a}} = \frac{d_{\mathbf{m}_i,\mathbf{a}} \sin\gamma}{\sin\beta}. \tag{14}$$

Now the observations $x_i(t)$ of the shock wave can be modeled according to the chosen simulation settings. Also some Gaussian noise was added to the "observed" signals to test the estimation performance with different *Signal-To-Noise Ratio (SNR)* levels, which are computed for each channel i by

$$\text{SNR}_i = 10\log_{10}\frac{E_{x_i}}{E_{n_i}}, \tag{15}$$

where E_{x_i} and E_{n_i} are the energies of the observed signals $x_i(t)$ and the added noise for each channel, respectively. Energy for signal $x_i(t)$ of length N is calculated by

$$E_{x_i} = \frac{1}{N}\sum_{t=1}^{N} x_i(t)^2. \tag{16}$$

The probability of finding the real likelihood maximum and, unfortunately, the computational burden increases as the number of particles is increased. During the testing it was found out that 2000–3000 particles are enough for the filter to converge in a computational time of about 10 seconds (using Matlab). The number of iterations k was now set to 20, which was decided by following the general progression of the particle weights. The estimation method was tested by generating *four* separate trajectories with a common microphone array located at the origin. The geometrical parameters of the trajectories are shown in Table I. Simulated gunshots were shot 5-7 times along each trajectory, so that in total over 20 estimations were made for each different noise level.

TABLE I
THE USED GEOMETRICAL PARAMETERS IN THE SIMULATIONS.

parameter	x (m)	y (m)	z (m)
$\mathbf{g}_1$	20.000	90.000	1.000
$\mathbf{h}_1$	-60.000	-150.000	0.000
$\mathbf{g}_2$	-15.000	110.000	0.000
$\mathbf{h}_2$	55.000	-150.000	0.000
$\mathbf{g}_3$	15.000	80.000	1.000
$\mathbf{h}_3$	-20.000	-150.000	0.000
$\mathbf{g}_4$	-25.000	80.000	1.000
$\mathbf{h}_4$	20.000	-150.000	0.000
$\mathbf{m}_1$	-0.150	0.000	0.040
$\mathbf{m}_2$	0.149	0.000	0.040
$\mathbf{m}_3$	0.000	0.175	0.040
$\mathbf{m}_4$	0.000	0.094	0.196

The normalized Root-Mean-Square errors (RMSE) of the *averaged S-point estimates* are drawn as a function of SNR in Fig. 5. The RMS error is defined for an observation sequence of length s as

$$\text{RMSE} = \sqrt{\frac{1}{s}\sum_{ind=1}^{s}(\hat{\theta}_{ind} - \theta_{ind})^2}, \tag{17}$$

where $\hat{\theta}$ and θ are the estimated and observed parameter vectors of length s, respectively. Normalization is done here by

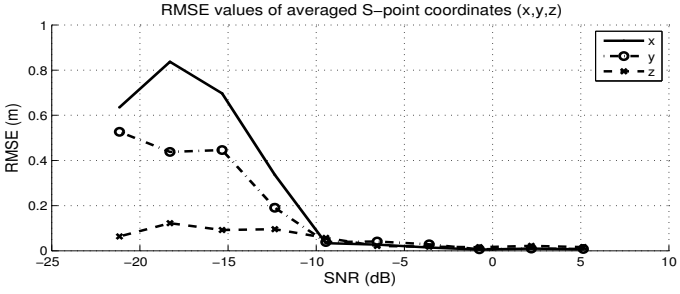

Fig. 5. Normalized RMSE values for the averaged S-point estimates as a function of SNR level.

mapping both the known and the estimated S-point vectors into *unit vectors* before calculating the RMSE values. Hence only the direction of the S-point is considered, which simplifies the comparison between estimation performances at different SNR levels. As the SNR value decreases, the RMS error starts to increase, as one would expect. However, the performance is still good at the SNR level of -8 dB, after which a sudden rise on the error values occur. The results are promising considering the usability of the method in a noisy environment.

V. RESULTS USING REAL-DATA

The simulation does not take into account any non-ideal aspects, such as echo, noise, or wind, which can render the observation circumstances challenging. Moreover, due to the aforementioned facts, the actual observed shock wave form does not perfectly obey the simplified signal model (6), as could be clearly seen in Fig. 2. Therefore, the state estimation procedure was also tested with over a hundred actual shock wave *recordings*, recorded in a outdoor shooting range with a 7.62 mm rifle and two separate microphone arrays with four *Sennheiser MKE 2P-C* condenser microphones attached to both ([16]). Rifle shots were recorded with both arrays at the same time, meaning that the caused shock waves were the same for the both arrays. The shooting sites, as well as the target point, were roughly spatially annotated during the recordings, so that the real bullet trajectory can be kept as more or less known. The microphone positions $\mathbf{m}_i$ are exactly known, and the shock wave observation moments are here manually annotated to avoid false detections and misses.

In Fig. 6, the CPA and S-point estimation results for both the microphone arrays with the recorded 7.62 mm rifle shots can be seen. Totally 114 shots from three different distances (75 m, 50 m, and 25 m) were used. The estimations were done using 2500 particles with 20 iteration rounds, yielding the results with averages μ and standard deviations σ shown in Table II. The average bullet trajectories are plotted with blue dashed lines in Fig. 6, and they are based on the μ results of Table II. The column D on the Table II stands for the estimated miss distance, and it is determined as

$$D = \frac{1}{n}\sum_{i=1}^{n}||\mathbf{m}_i - \mathbf{a}||. \tag{18}$$

The prior knowledge used to initialize the particles is also shown on the last row of Table II. The particle values are kept within these borders during the estimation, e.g. no caliber values less than 2.50 mm are allowed.

TABLE II
THE AVERAGE RESULTS FOR THE CALIBER, BULLET SPEED, AND MISS DISTANCE ESTIMATIONS FOR BOTH ARRAYS.

Array 1	**caliber (mm)**	**speed (m/s)**	**D (m)**
ground truth	7.62	$\approx 670 - 690$	$\approx 3.0 - 7.0$
$\mu \pm \sigma$	6.85 ± 1.31	673 ± 105	11.3 ± 6.0
Array 2	**caliber (mm)**	**speed (m/s)**	**D (m)**
ground truth	7.62	$\approx 670 - 690$	$\approx 16.0 - 20.0$
$\mu \pm \sigma$	7.98 ± 1.14	665 ± 108	16.2 ± 6.8
Priors	$\phi \in (2.50, 12.00)$	$v \in (450, 900)$	$D \in (0.0, 40.0)$

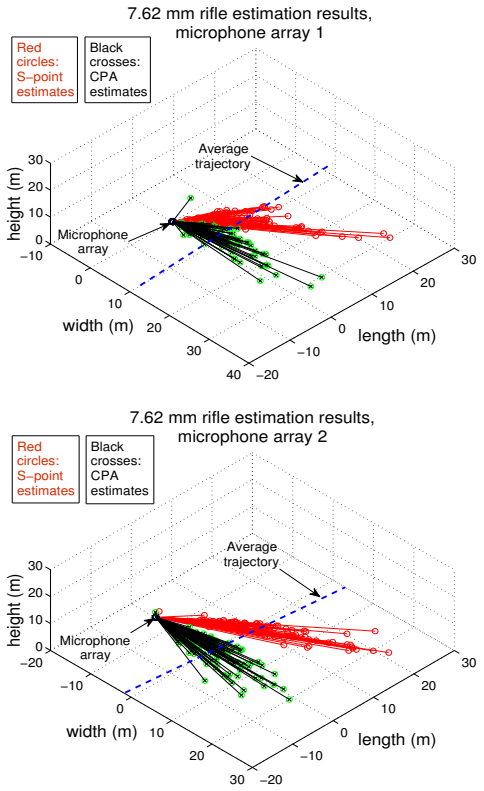

Fig. 6. Estimated CPAs (black crosses) and S-points (red circles) of both microphone arrays for 114 rifle shots. The average bullet trajectories are plotted with blue dashed lines.

In the case of array 1, the miss distance estimates of the nearest 25 m shooting site are too large, which affects also on the overall miss distance result. As the gun is close to the array, the delay between the shock wave and muzzle blast becomes much shorter than with longer shooting distances. The muzzle blast signature can thus easily mix up the estimation, especially since the annotated shock wave observations are not perfect. Due to a rather high amount of separate gunshots, also the standard deviation values are quite high. However, the mean values are approximately correct, as most of the state estimates are concentrated near the ground truth values. It should be yet noted, that the actual trajectory estimation results of Fig. 6 are *not* completely comparable between the two arrays, since the arrays are not sharing exactly the same Cartesian coordinate grid. Nevertheless, it can be stated that the estimation procedure is verified to work with real data.

VI. Conclusions

Using particle filters for trajectory estimation is shown to be working trustworthily with a moderate computational effort. The amount of particles needed in the estimation procedure is scalable, allowing adjustment to be made between the speed and accuracy of the method. Estimations of the bullet caliber and speed are also working relatively well, although the standard deviation can raise rather large, due to misestimated single shots in a longer series.

The proposed estimation approach is shown to be capable to solve the multidimensional inference problem of bullet state estimation. The cross correlation is not dependent on the signal amplitude, which, again, is the most reliable signature for determining the miss distance for the trajectory. Therefore, more research is focused on refining the proposed scheme, as some new recordings with calibrated microphones are planned to be made. Testing the method with correlating noise sources is another future task to be studied.

Acknowledgment

We would like to thank the Finnish Funding Agency for Technology and Innovation (TEKES) for funding (contract 1203/31/07). The data used in the implementation was gathered together with Hasse Sinivaara, Teemu Korhonen and Antti Löytynoja. Hasse Sinivaara also organized the shooting session together with Pasi Auranen, and the original idea of utilizing the shock waves for the estimation was presented by them.

References

[1] N. Shachtman, "Iraq sniper attacks quadruple," 2007, read 18-May-2009. [Online]. Available: http://blog.wired.com/defense/2007/10/iraq-sniper-att.html

[2] Brian M. Sadler *et al.*, "Optimal and wavelet-based shock wave detection and estimation," in *J. Acoust. Soc. Am. 104(2)*, August 1998, pp. 955–963.

[3] Brian G. Ferguson *et al.*, "Acoustic sensing of direct and indirect weapon fire," in *Intelligent Sensors Sensor Networks and Information Processing*, December 2007, pp. 167–172.

[4] P. Volgyesi *et al.*, "Shooter localization and weapon classification with soldier-wearable networked sensors," in *Proceedings of the 5th International conference on Mobile systems, Applications and Services*, June 2007, pp. 113–126.

[5] BBN Technologies, "Boomerang," 2009, read 15-April-2009. [Online]. Available: http://www.bbn.com/products_and_services/boomerang/

[6] Areva, "PIVOT system," 2007, read 26-May-2009. [Online]. Available: http://www.01db-metravib.com/defense.3/news.269/?no_cache=1&L=1

[7] G. Whitman, "The flow pattern of a supersonic projectile," in *Communications on Pure and Applied Mathematics*, 1952, pp. 301–348.

[8] M. Sanjeev Arulampalam *et al.*, "A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking," in *IEEE TRANSACTIONS ON SIGNAL PROCESSING, VOL. 50, NO. 2*, February 2002, pp. 174–187.

[9] Edward M. Carapezza *et al.*, "Darpa counter-sniper program: Phase 1 acoustic systems demonstration results," Edward M. Carapezza and D. Spector, Eds., vol. 2938, no. 1. SPIE, 1997, pp. 299–310.

[10] R. Maher, "Modeling and signal processing of acoustic gunshot recordings," in *IEEE Signal Processing Society 12th DSP Workshop*, September 2006, pp. 257–261.

[11] A. Gelman *et al.*, *Bayesian Data Analysis*, 2nd ed. CRC Press, 2004, ISBN 1-58488-388-X.

[12] C. Knapp and C. Carter, "The generalized correlation method for estimation of time delay," in *IEEE TRANSACTIONS ON ACOUSTICS, SPEECH, AND SIGNAL PROCESSING, VOL. ASSP-24, NO 4*, August 1976, pp. 320–327.

[13] T. Chen and H. Wu, "Adaptive impulse detection using center-weighted median filters," in *IEEE Signal Processing Letters, Vol. 8, No. 1*, January 2001, pp. 1–3.

[14] A. Dufaux *et al.*, "Automatic sound detection and recognition for noisy environment," in *EUSIPCO 2000: European signal processing conference No. 10*, September 2000, pp. 1033–1036.

[15] E. Weisstein, "Point-line distance – 3-Dimensional," 2009, read 11-May-2009. [Online]. Available: http://mathworld.wolfram.com/Point-LineDistance3-Dimensional.html

[16] T. Mäkinen, "Detection and direction estimation of impulsive sound sources using a microphone array." Tampere University of Technology, September 2008, Master of Science Thesis.

Publication P2

Toni Mäkinen and Pasi Pertilä, Shooter localization and bullet trajectory, caliber, and speed estimation based on detected firing sounds. *Applied Acoustics*, vol. 71, pages 902 – 913, 2010.

Publication P3

Toni Mäkinen, Serkan Kiranyaz, and Moncef Gabbouj, Content-based audio classification using collective network of binary classifiers. *Proceedings of the IEEE Workshop on Evolving and Adaptive Intelligent Systems (EAIS)*, pages 116 – 123, Paris, France, April 2011.

Content-based Audio Classification using Collective Network of Binary Classifiers

Toni Mäkinen Serkan Kiranyaz Moncef Gabbouj

Tampere University of Technology
Department of Signal Processing
P.O. Box 553, Tampere, Finland

***Abstract*—In this paper, a novel collective network of binary classifiers (CNBC) framework is presented for content-based audio classification. The topic has been studied in several publications before, but in many cases the number of different classification categories is quite limited and needed to be fixed *a priori*. We focus our efforts to increase both the classification accuracy and the number of classes, as well as to create a *scalable network* design, which allows introducing new audio classes incrementally. The approach is based on dividing a major classification problem into several *networks* of *binary classifiers (NBCs)*, where each NBC adapts its internal topology according to the classification problem at hand, by using *evolutionary Artificial Neural Networks (ANNs).* In the current work, feed-forward ANNs, or the so-called Multilayer Perceptrons (MLPs), are evolved within an architecture space, where a *stochastic optimization* is applied to seek for the optimal classifier configuration and parameters. The performance evaluations of the proposed framework over an 8-class benchmark audio database demonstrate its scalability and notable potential, as classification error rates of less than 9% are achieved.**

Keywords - audio content - based classification; evolutionary neural networks; particle swarm optimization; multilayer perceptron

I. INTRODUCTION

The rapid growth of the database sizes both in the Internet and home computers has created new and challenging tasks in maintaining the flexibility in handling such large amounts of data. Audio content-based retrieval offers several advantages and possibilities over traditional text-based queries, as manual annotation of audio information in large databases is not convenient or perhaps feasible at all. Moreover, in many practical situations it would be ideal to retrieve certain kind of audio content from a large database using a *reference* audio clip, for example when searching for a certain type of music or environmental sounds. For this, audio content-based classification is needed, which is studied in this work using a novel approach of collective (evolutionary) classifier networks.

The idea of content-based audio indexing and retrieval was first presented by Wold et al. in [1], where pitch, loudness, brightness and bandwidth features were used in classifying the used audio database (*Muscle Fish*). Since then, the research has been rather active, and many classification schemes have been proposed to improve the accuracy of the classification performance. In this paper, the focus is put purely on the audio *classification* problem, meaning that audio *segmentation* and *change-point detection* approaches are out of the scope of this study. The pure audio classification approaches found from the literature can be divided into two main categories, namely the *model-based* and the *rule-based* methods. The latter ones are convenient when no complete training data is available, as the classification is performed in an unsupervised manner. This is accomplished by using *thresholds* for different audio features, as performed in [2]-[4]. There are, however, several problems with the unsupervised learning techniques, such as the need for high amount of heuristics, and the limited number of classes that need to be fixed *a priori*. Such drawbacks limit their practical use for dynamic, ever-growing multimedia repositories, which are common in many environments and applications today.

In model-based methods, based on supervised learning, Chen et al. in [5] used a *Support Vector Machine (SVM)* to classify audio from two movies into 5 classes, namely music, speech, environmental sound, speech with music, and music with environmental sound. The results (~78% classification accuracy) showed improvements in classification error rates when compared to *k-Nearest Neighbour (kNN), Artificial Neural Networks (ANNs),* and *Naive Bayes (NB)* classifiers. Rather high training dataset (70% of the entire database) was used in achieving the results, whereas Zhu et al. in [6] used the same kind of SVM approach with a smaller training set and additional validation set, achieving more or less similar outcome with [5]. Chu and Champagne [7] used a slightly different approach for SVM-based classification by introducing their FFT-based noise-robust spectrum. Improved classification results were achieved in noisy test cases, but only speech, music, and noise were classified in their work. SVM was used also in [8], where it was applied to transform domain indexing by using a non-standard audio codec in a music genre-classification application. As a popular classifier, SVM was also used, along with the *Hidden Markov Models (HMMs),* in [9] to classify audio content into five non-silent classes. In [9], a unique HMM-model is trained for each non-silent class using MPEG-7 features. Training set encapsulated 50% of the entire dataset in achieving the reported accuracy rates, which are, however, highly dependent on the selection of the SVM parameters, which is a well-known fact in the field. Peeters in [10] used *Gaussian Mixture Models (GMMs)* together with the

This work was supported by the Academy of Finland, project No. 213462 (Finnish Centre of Excellence Program (2006 - 2011)

HMMs to model individual classes in the context of music genre recognition (with six categories). *Principal Component Analysis (PCA)* and *Linear Discriminant Analysis (LDA*) were needed to lower the feature space dimensionality, whereas the classification itself was done based on the generated statistical models. The classification results obtained in music genre recognition are close to the state-of-the-art, but selection of the best classifier configuration remains an unsolved problem. Another supervised classification approach was presented by Harb and Chen in [11], where modeling based on human perception was applied. An average classification accuracy of 63.5% was achieved for six music genres.

In general, the aforementioned audio classification efforts, and many alike, put lots of effort in adjusting the classifier parameters, so as to "fit" to the specific classification problem at hand as close as possible. However, it is obvious that this is not too applicable for a general classifier that should achieve robust and efficient performance levels over generic audio databases. Therefore, for any audio repository, the setting of the classifier parameters, as well as the choice of the classifier configuration, should be optimal, so as to maximize the classification accuracy. Also, the number of different classes is usually quite limited and specific to a certain audio domain, whereas in order to have a reliable retrieval performance on a versatile database, more classes should be supported. Furthermore, support for dynamic updates in the databases is rare at the moment, as in most cases the training dataset and the number of classes need to be fixed beforehand.

In order to address these problems, in this paper, we shall focus on a global and data-adaptive framework design that embodies a collective network of evolutionary binary classifiers (CNBC). The main idea of the framework was first introduced in a previous work [12], for another application area, being now specifically designed for audio classification purpose. Earlier, fundamentally similar type of approaches of constructing an ensemble of neural networks (a.k.a. *neuro-ensemble*) have been introduced (see e.g. [13]), but, to our knowledge, the framework structure presented in this paper has not been used in audio classification scheme before. The issues specifically targeted in our approach are:

- *Evolutionary Search*: Seeking for the optimum classifier network architecture among a certain collection of different configurations (the so-called *Architecture Space, AS*).
- *Evolutionary Update in the AS*: Keeping only "the best" individual configuration in the AS among indefinite number of evolution runs.
- *Class / feature Scalability*: Support for varying number of classes and audio features. A new class / feature can be dynamically inserted into the framework without requiring a full-scale re-configuration or re-training.
- *High efficiency* for the evolution (training) process: Using as compact and simple classifiers as possible.
- *Maximizing the classification accuracy:* Using several audio features to take advantage of the discrimination power of each one of them.

In this work, *Multilayer Perceptrons (MLPs)* are evolved in the proposed CNBC framework. The recently proposed *Multi-Dimensional Particle Swarm Optimization (MD-PSO)* [14] is used as the primary evolutionary search technique.

The rest of the paper is organized as follows. Section II briefly presents the applied evolutionary ANNs and the MD-PSO technique, whereas Section III describes the feature extraction process and introduces the audio features used. The proposed CNBC framework and the evolutionary update mechanisms are explained in detail in Section IV, and the classification results and performance evaluation over an 8-class database are given in Section V. Finally, Section VI concludes the paper and discusses future research directions.

II. EVOLUTIONARY NEURAL NETWORKS

In this section, we will first briefly discuss the applied evolutionary technique, MD-PSO, which is used in an architecture space to search for the optimal classifier configuration. Second, the concept of evolutionary feed-forward artificial neural networks is introduced. Finally, an overview of the well-known *Back Propagation (BP)* method will be given, which can be used also exhaustively to perform a sequential search for the optimal classifier in an AS.

A. Multi-Dimensional Particle Swarm Optimization

The Particle Swarm Optimization (PSO) was introduced by Kennedy and Eberhart [15] in 1995 as a population-based stochastic search and optimization process. In a PSO process, a swarm of particles, each of which represents a potential solution to the optimization problem at hand, navigates through a search space. The particles are randomly distributed over the search space, and the goal is to *converge* to the global optimum of a function or a system. Each particle keeps track of both its current position, and the best position achieved so far in the search space. The latter is called the *personal best* value (*pbest*), while the PSO process keeps also track of the *global best* solution achieved so far by the whole swarm (*gbest*). During their journey in the search space with discrete time iterations, the *velocity* of each particle in the next iteration is computed by the best position of the swarm (position of the particle *gbest*, the *social* component), the best personal position of the particle (*pbest*, the *cognitive* component), and the current velocity of the particle (the *memory* term). Both *social* and *cognitive* components contribute randomly to the position of the particle in the next iteration.

In this research we will use the multi-dimensional (MD) extension of the basic PSO (*bPSO*) method, the MD-PSO. Instead of operating with a fixed number of dimensions, D, the MD-PSO algorithm is designed to seek both *positional* and *dimensional* optima within a certain dimension range $\{D_{\min}, D_{\max}\}$. For this, each particle has *two sets* of components, each of which has been subjected to two independent and consecutive processes. The first set is a regular positional PSO, taking care of the traditional velocity updates and positional shifts in the D-dimensional search (solution) space, whereas the second set is the dimensional PSO, allowing the particles to navigate through dimensions. Accordingly, now each particle keeps track of its latest position, velocity and personal best position in a *particular dimension,* so that when the particle re-

visits the same dimension later, it can perform its regular "positional" update. The dimensional PSO process of each particle may then move the particle to another dimension, where it will remember its positional status and shall be updated with the positional PSO process at that dimension. The swarm, on the other hand, keeps now track of the *gbest* particle in *each dimension*, and the dimensional PSO process of each particle uses its personal best dimension (in which the personal best fitness score has been achieved so far). Finally, the swarm keeps track of the global best dimension, *dbest*, among all the personal best dimensions. Thus, the *gbest* particle in the *dbest* dimension represents the optimum solution found.

In a MD-PSO process at time (iteration) t, each particle a in the swarm with S particles, $\xi = \{x_1,..,x_a,..,x_S\}$, is represented by the following symbols:

$x_{a,j}^{d_a(t)}(t)$: j^{th} component of the *position* of particle a in dimension $d_a(t)$.

$\tilde{x}_{a,j}^{d_a(t)}(t)$: j^{th} component of the *personal best position* of particle a in dimension $d_a(t)$.

$v_{a,j}^{d_a(t)}(t)$: j^{th} component of the *velocity* of particle a in dimension $d_a(t)$.

$d_a(t)$: dimension of particle a.

$\tilde{d}_a(t)$: *personal best dimension* of particle a.

$vd_a(t)$: dimensional velocity of particle a.

$gx_j^d(t)$: j^{th} component of the *global best position* of swarm in dimension d.

Let f denote a *fitness function* that is to be optimized within a certain dimension range, $\{D_{\min}, D_{\max}\}$. Without loss of generality, assume that the objective is to find the *minimum* of f at the optimum dimension within a multi-dimensional search space. Assume also, that the particle a visits (back) the same dimension after T iterations (i.e. $d_a(t) = d_a(t+T)$). Then, the personal best position can be updated at iteration $t+T$ as,

$$\tilde{x}_{a,j}^{d_a(t+T)}(t+T) = \begin{cases} \tilde{x}_{a,j}^{d_a(t)}(t) & if\ f\left(x_{a,j}^{d_a(t+T)}(t+T)\right) > f\left(\tilde{x}_{a,j}^{d_a(t)}(t)\right) \\ x_{a,j}^{d_a(t+T)}(t+T) & else, \end{cases} \quad (1)$$

$$j = 1, 2, \ldots, d_a(t+T).$$

Furthermore, the personal best dimension of particle a can be updated in iteration $t+1$ as,

$$\tilde{d}_a(t+1) = \begin{cases} \tilde{d}_a(t) & if\ f\left(x_{a,j}^{d_a(t+1)}(t+1)\right) > f\left(\tilde{x}_{a,j}^{\tilde{d}_a(t)}(t)\right) \\ d_a(t+1) & else. \end{cases} \quad (2)$$

Fig. 1 shows an example MD-PSO and *bPSO* particles. Particle a in *bPSO* is at (fixed) dimension, $D = 5$, and contains only positional components, whereas in MD-PSO, particle a contains both the positional and dimensional components. The dimension range for MD-PSO is given by $\{D_{\min}, D_{\max}\} = \{2, 10\}$, so that 9 sets of positional components are included in a.

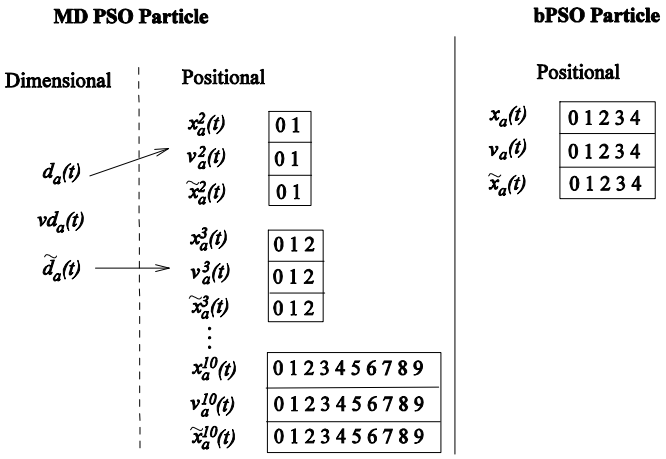

Figure 1. MD-PSO (left) vs. bPSO (right) particle structures for dimensions $\{D_{min}=2, D_{max}=10\}$. At time t, $\boldsymbol{d_a(t) = 2, \tilde{d}_a(t) = 3}$.

In this example the particle a currently resides at dimension 2 ($d_a(t) = 2$), while its personal best dimension is 3 ($\tilde{d}_a(t) = 3$). Hence, at time t a positional PSO update is first performed over the positional components of $x_a^2(t)$, after which the particle may move to another dimension with respect to the dimensional PSO. Recall that each positional component $x_a^2(t)$ represents a potential solution in the data space to the problem. The algorithmic flowchart and further details about MD-PSO can be obtained from [16].

B. MD – PSO for Evolving MLPs

The MD-PSO seeks (near-) optimal networks in an AS, which can be defined over any type of ANNs with any properties. All network configurations in the AS are enumerated into a *hash table* with a proper hash function, which ranks the networks with respect to their complexity, i.e. associates higher hash indices to networks with higher complexity. The MD-PSO can then treat each index as a unique dimension in the search space. The dimension thus corresponds to the optimal classifier architecture, while the position (solution) encapsulates the optimum network parameters (connections, weights and biases). Suppose, for the sake of simplicity, that a certain *range* is defined for the minimum and maximum number of MLP layers, $\{L_{\min}, L_{\max}\}$, as well as for the number of neurons in the hidden layer l, $\{N_{\min}^l, N_{\max}^l\}$. The sizes of both input and output layers, $\{N_i, N_o\}$, are determined by the problem, and hence fixed. The AS can then be defined by only two range arrays:

$$R_{\min} = \{N_i, N_{\min}^1, \ldots, N_{\min}^{L_{\max}-1}, N_o\},$$
$$R_{\max} = \{N_i, N_{\max}^1, \ldots, N_{\max}^{L_{\max}-1}, N_o\},$$

where the first one is for the minimum-, and the second one is for the maximum number of neurons allowed for each layer of a MLP. The size of the both arrays is naturally $L_{\max} + 1$, where the corresponding entries define the range of the l^{th} hidden layer for all those MLPs that can have the l^{th} hidden layer. The terms $L_{\min} \geq 1$ and $L_{\max}$ can be set to any value meaningful for the problem at stake. The hash function then enumerates all potential MLP configurations into hash indices, starting from the simplest MLP with $L_{\min} - 1$ hidden layers (each of which has the minimum number of neurons given by $R_{\min}$), to the

most complex one with $L_{\max} - 1$ hidden layers (each of which has the maximum number of neurons given by $R_{\max}$).

Let N_l be the number of hidden neurons in layer l of a MLP with the input and output layer of sizes N_i and N_o, respectively. The input neurons are merely fan-out units, as no processing is done in them. Let g be the *activation function* (e.g. sigmoid) applied over the weighted inputs and a bias. Thus we can write,

$$y_n^l(p) = g\left(Y_n^{p,l}\right), \qquad Y_n^{p,l} = \sum_m w_{mn}^{l-1} y_m^{l-1}(p) + \theta_n^l, \quad (3)$$

where $y_n^l(p)$ is the output of the n^{th} neuron of the l^{th} hidden / output layer when a pattern p is fed into it, w_{mn}^{l-1} is the weight from the m^{th} neuron in layer $l-1$ to the n^{th} neuron in layer l, and θ_n^l is the bias value of the n^{th} neuron in the l^{th} layer. The training mean square error, MSE, is formulated as:

$$MSE = \frac{1}{2PN_o} \sum_{p \in A} \sum_{n=1}^{N_o} \left(t_n(p) - y_n^o(p)\right)^2, \quad (4)$$

where $t_n(p)$ is the target (desired) output, and $y_n^o(p)$ is the actual output from the n^{th} neuron in the output layer, l=o, for pattern p in the training dataset A with size P, respectively. At time t, the particle a has the positional component formed as,

$$x_{a,j}^{d_a(t)}(t) = \\ = \Psi^{\mathrm{d_a(t)}}\left\{\{w_{mn}^0\}, \{w_{mn}^1\}, \{\theta_{\mathrm{n}}^1\}, \ldots, \{w_{mn}^{o-1}\}, \{\theta_{\mathrm{n}}^{o-1}\}, \{\theta_{\mathrm{n}}^o\}\right\}, \quad (5)$$

where $\{w_{mn}^l\}$ and $\{\theta_{\mathrm{n}}^l\}$ represent the sets of weights and biases of the layer l of the MLP-configuration $\Psi^{\mathrm{d_a(t)}}$. Note that the input layer (l=0) contains only weights, whereas the output layer (l=o) contains only biases. By the means of such a direct encoding scheme, the particle a thus represents all potential network parameters of the MLP architecture at the dimension (hash index) $d_a(t)$. As mentioned earlier, the dimension range $\{D_{\min}, D_{\max}\}$ where the MD-PSO particles can make inter-dimensional jumps, is determined by the AS defined. Apart from the regular limits, such as (positional) velocity range, $\{V_{\min}, V_{\max}\}$, and dimensional velocity range, $\{VD_{\min}, VD_{\max}\}$, the data space can be also limited by some practical range, i.e. $X_{\min} < x_{a,j}^{d_a(t)}(t) < X_{\max}$. Setting the MSE in (4) as the fitness function, to be used in the MD-PSO, enables then performing evolutions of both the network parameters and the network architectures. Further details and an extensive set of network evolution experiments can be found in [14].

C. The Back-Propagation Algorithm

Back Propagation (BP) is the most commonly used training technique for feed-forward ANNs. It is a supervised training technique that has been used in pattern recognition and classification problems in many application areas. Essentially, BP is just a *gradient descent* algorithm in the *error* space, which may be complex and contain many deceiving local minima (multi-modal). Therefore, BP gets easily trapped into a local minimum, making it entirely dependent on the initial (weight) settings. However, due to its simplicity and relatively lower computational cost, BP can be applied exhaustively over the network architectures with *random initializations*, to find out the optimal architecture. Since the AS is composed of only compact networks, with such an exhaustive search the probability of finding (converging) to a (near-) optimum solution in the error space is significantly increased.

The used BP algorithm can be summarized as follows:

1. Initialize the weights w_{mn}^l and biases θ_n^l randomly.
2. Feed a pattern p to the network and compute the output $y_n^l(p)$ of each neuron n in each hidden layer l.
3. Calculate the error between the final output $y_n^o(p)$ of each output neuron and the desired output $t_n(p)$ as $e_n^o(p) = t_n(p) - y_n^o(p)$.
4. For each neuron n, calculate the partial derivatives $\frac{\partial E(p)}{\partial h_n^l}$, where $E(p)$ is the total *error energy* defined as $E(p) = \frac{1}{2}\sum_{n \in o}\left(e_n^o(p)\right)^2$, and h_n^l is a uniform symbol for the parameters w_{mn}^l and θ_n^l.
5. Update the parameters as follows:

$$h_n^l(t+1) = h_n^l(t) - \eta \frac{\partial E(p)}{\partial h_n^l}, \quad (6)$$

where η is a *learning rate* parameter.

6. Repeat steps 2-5 until some stopping criterion is reached.

One complete run over the training dataset is called an *epoch*. Usually many *epochs* are required to obtain the best training results, but, on the other hand, too many training *epochs* can lead to over-fitting. In the above realization of the BP algorithm, the network parameters are updated after every training sample (pattern p). This is called an *online* or *sequential* training mode. Another possibility is the *batch* mode, where all the training samples are first presented to the network, and then the parameters are adjusted so to minimize the total training error. The *sequential* mode is often favored over the *batch* mode, as less storage space is required. Moreover, the *sequential* mode is less likely to get trapped into a local minimum, as updates at every training sample make the search stochastic in nature. Hence, *sequential* BP mode is used for MLP training in this study.

III. Audio Feature Extraction

As a common approach in audio signal processing, the audio signal to be analyzed is first divided into short time windows / frames (20-40 ms), from which the audio features are extracted. This is to prevent averaging the signal over long segments, in which case the discrimination of audio features may decrease significantly. In this study, *three sets* of features are extracted from each audio clip to be classified, divided as:

- *General Audio Features:* These consist of sub-band power (4 bands), band energy ratio (BER), sub-band centroid (SC), zero-crossing rate (ZCR), short average energy (SAE), brightness, bandwidth, spectral roll-off, spectral flux, and fundamental frequency (FF).
- *MFCCs:* The first 24 coefficients of the extracted Mel-frequency cepstral coefficients.

- *Linear Prediction Coefficients (LPC):* The 8^{th} order LP coefficients (the order is based on the *16 kHz sampling frequency* used in the classified audio samples).

The *feature vector (FV)* dimensions for each feature set are thus 13, 24, and 8, respectively.

Extracting the features from short time frames leads into a rather big number of FVs even from a short audio clip. Therefore, in our work, a certain amount of *key frames* (*KFs,* see [4]), are first selected among the frames, being sort of "prototypes", which are chosen so to represent the others as accurately as possible. The ultimate goal is to find concise and representative feature sets from each audio clip, to speed up the classification process without losing any vital information of the original signals. The reasoning behind the idea of exploiting only a small fraction of the audio frames is based on an assumption, that the elementary sounds within an audio clip are immensely repetitive, and often entirely alike. For an efficient KF selection, audio frames with similar acoustical features within an audio clip are *clustered*, and only one or few frames from each cluster are considered as KFs to represent the others in that cluster. Here the number of KFs is empirically set between ~1-2% of the total amount of frames, being relatively lower for longer clips. Thus, the actual number of key frames selected from each clip varies approximately between 40 and 200 frames, depending on the length and variation of the signal. Once the number of KFs is determined, a *Minimum Spanning Tree (MST)* clustering technique is applied. Every node in the tree represents the extracted features of a unique audio frame. The clustering scheme is illustrated in Fig. 2, where the word "Speech Lab" is divided into seven separate clusters, according to the similarity of the extracted frames. More technical details about the audio clustering scheme can be found in [4].

For each of the three extracted features sets, the proposed classification framework evolves a separate binary classifier (BC) per each pre-determined class, so that a unique *network* of BCs (NBC) is created for each class. Note that, as such, the proposed framework performs classification over the KFs, and not the actual audio clips. Hence, a *majority rule* is applied to the classified KFs to decide the final class of the corresponding clip. For this, a specific table is created, where the KF indices corresponding to each audio clip in the database are stored.

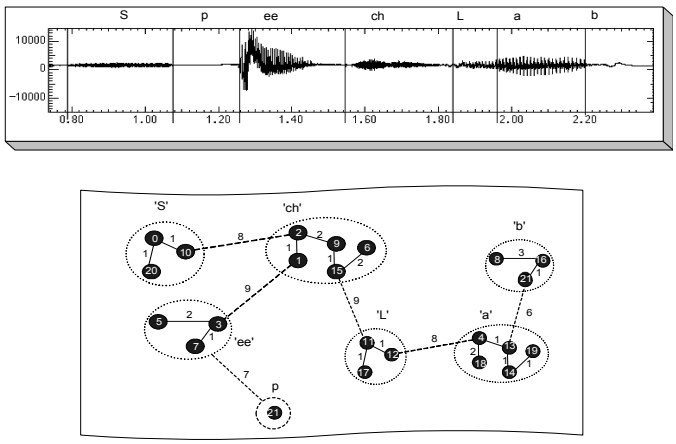

Figure 2. An illustrative MST clustering scheme.

IV. The Classification Framework

This section describes in detail the proposed classification framework: the Collective Network of (Evolutionary) Binary Classifiers (CNBC). The framework takes as an input the extracted feature vectors from the training dataset KFs, after which the internal network topology is configured, and all the corresponding binary classifiers are evolved individually. Before going into full details of the CNBC, the used AS evolutionary update mechanism will be introduced.

A. Evolutionary Update in the Architecture Space

Since the primary evolution technique used, MD-PSO, is a stochastic optimization method, it is not guaranteed that it will always find the optimal solution. Thus, in order to improve the probability of convergence to the global optimum, several evolutionary *runs* can be performed. Let Q_R be the number of runs and Q_C be the number of configurations in the AS. For each run, the objective is to find the optimal classifier within the AS, with respect to some pre-defined criterion. Note that, along with the *best* classifier, all the other configurations in the AS are also evolved simultaneously, so that the configurations are continuously (re-)trained within each run. Thus, during the process, any network configuration may *replace* the current best one in the AS, if it is surpassed in terms of the classification performance criterion. This is also true in the *exhaustive search*, where each network configuration in the AS is evolved using Q_R Back-Propagation (BP) runs.

Fig. 3 demonstrates the evolutionary update operation over a sample AS containing 5 MLP configurations. The bigger table in Fig. 3 shows the training *Mean Square Error (MSE)*, which is the criterion used to select the optimal configuration at each run. The best runs for each configuration are highlighted, and the best configuration in each run is tagged with '*'. In this case, at the end of three runs, the overall best network with MSE = 0.1 has a configuration 15x3x2, and thus it is used as the classifier for all the forthcoming classification tasks, until a new configuration may surpass it in a future run. As can be seen from this example, each BC configuration in the AS can only evolve into a *better* state from the previous one (in terms of the training MSE), which is the main motivation for the proposed evolutionary update mechanism.

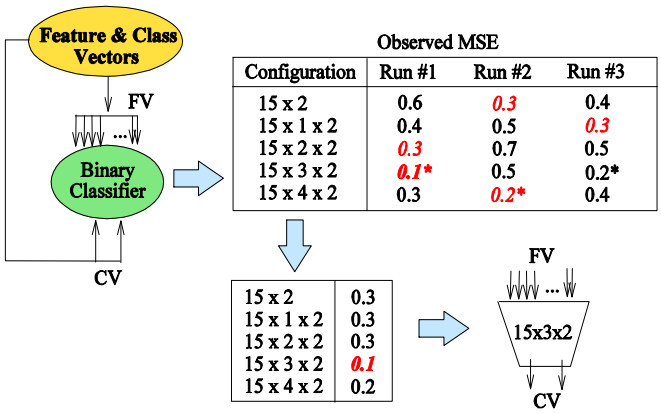

Figure 3. Evolutionary update in a sample AS for MLP configuration arrays $\boldsymbol{R_{min}} = \{\mathbf{15}, \mathbf{0}, \mathbf{2}\}$ and $\boldsymbol{R_{max}} = \{\mathbf{15}, \mathbf{4}, \mathbf{2}\}$, where $Q_R = 3$ and $Q_C = 5$. The best runs for each configurations are highlighted, and the best configuration in each run is tagged with '*'.

B. Collective Network of Binary Classifiers

1) The Topology:

In the CNBC framework, the individual networks of BCs (NBCs) *evolve* continuously with the ongoing evolution sessions by using the *ground truth training data* (*GTD*) given by the user. Each BC in a particular NBC performs binary classification using one of the three extracted FVs. Each NBC has also a "fuser" BC in its output layer, which collects and fuses the binary outputs of all the BCs in the input layer. A single binary output is then generated from each NBC, indicating the *relevancy* of the current input KF to the NBC's corresponding class. Due to its structure, the CNBC can be dynamically *scaled* into any number of classes, as whenever a new class is defined, a new corresponding NBC will be created and evolved on top of the existing structure. The procedure does not require changing or updating the other NBCs, as long as they pass the so-called *verification test*, which is performed by selecting a specific accuracy threshold, and by seeing whether the existing NBCs classify the training samples of the new class(es) accurately enough (and not confuse with them). In this work, an accuracy threshold of 95% was applied.

As is shown in Fig. 4, in CNBC, a learning problem with many classes and features can be *divided* into as many NBCs (and BCs within) as necessary, so as to negate the need for complex classifiers. This is a notable advantage, since the performance of the training and evolution processes degrades significantly as the classifier complexity increases (due to the well-known *curse of dimensionality* – phenomenon). Another major benefit of the approach, with respect to efficiency, is that the configurations in the AS can be kept very compact, so that unfeasibly large storages and heavy computations can be avoided. This is especially important for the BP method, since the amount of deceiving local minima is significantly lower in the error space for simple and compact ANNs.

In order to maximize the final classification accuracy, a dedicated *class selection technique* is applied. In all the BCs, a *1-of-M* encoding scheme, with *M=2,* is used. Let us denote $CV_{f,1}$ and $CV_{f,2}$ as the first and second output of the f^{th} BC's *class vector (CV)*, respectively. The class selection in the *1-of-2* encoding scheme is then performed by comparing the two individual outputs, and the encoded output is determined as *positive* if $CV_{f,2} > CV_{f,1}$, and *negative* otherwise. The same encoding scheme applies for the fuser BC output, which determines the output of the whole NBC. A *class selection* block, illustrated at the bottom of Fig. 4, then collects the CVs of each NBC, and selects the "most positive" output among all the NBCs as the final classification outcome. Here a so-called *winner-takes-all* strategy is utilized, where the positive class index, c^*, ("the winner") is defined as,

$$c^* = \arg \max_{c \in [0, C-1]} (CV_{c,2} - CV_{c,1}), \qquad (7)$$

where C is the number of classes (NBCs). This way the erroneous cases (false positives), where there exist more than one NBC with positive outcome, can be handled properly.

2) Evolution of the CNBC:

The evolution of the CNBC, or a subset of NBCs, is performed for each NBC individually with a *two-phase* operation, as is illustrated in the upper part of Fig. 4. In Phase 1, the BCs

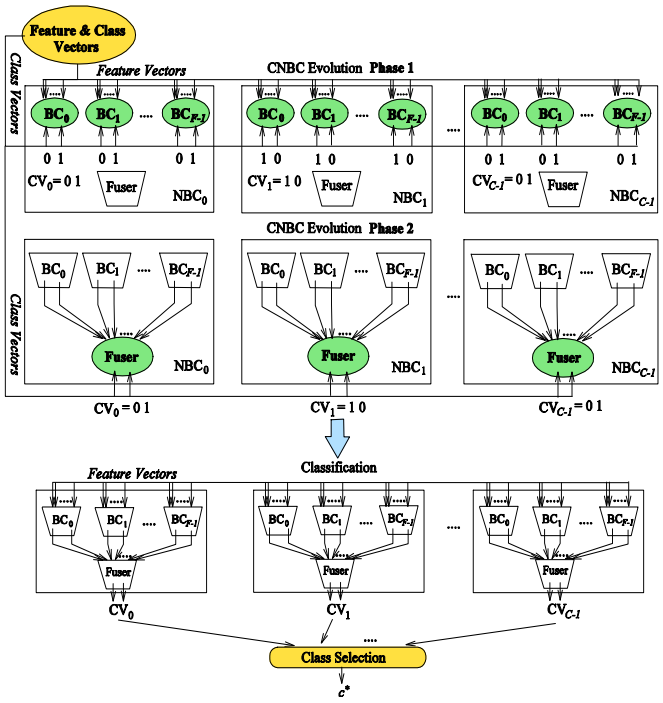

Figure 4. Illustration of the two-phase evolution session over BC architecture spaces in each NBC, and the topology of the CNBC framework with C classes and F feature sets.

of each NBC are first evolved by giving an input set of FVs and the target CVs (the GTD). Recall, that each CV is associated with a unique NBC, and that the fuser BCs are not yet used at this phase. Once the evolution session is over, the AS of each BC is *saved*, so as to be used for potential (incremental) evolution sessions in the future (Section IV.B.3). The best BC configuration in the AS is used to forward-propagate the respective FVs of the training dataset, in order to compose the BC outputs, which, again, are used as input FV for the corresponding fuser BC. The fuser BCs are then evolved in Phase 2 of the CNBC evolution process, where each fuser BC *learns* the *significance* of its individual BCs (and their feature sets). This can be viewed as a way of applying an efficient *feature selection* scheme, so that the fuser, if properly evolved and trained, can "weight" each BC accordingly. This way the potential of each feature set (and its BC) will be optimally fused according to their discrimination power over each class. Similarly, each BC in the first layer shall in time learn the significance of the individual feature components of the corresponding feature set. That is, the CNBC, if properly evolved, will learn the significance (or the discrimination power) of each feature set, as well as their individual components (the single features).

3) Incremental Evolution of the CNBC:

The proposed CNBC framework is designed for continuous "incremental" evolution sessions, where each session may further improve the classification performance of each BC using the advantage of the "evolutionary updates". The main difference between the initial and the subsequent evolution sessions is in the *initialization* phase of the evolution process: the former uses *random* initialization, whereas the latter starts from the previously saved AS parameters of each classifier in each NBC. Note that the training dataset used for the

incremental evolution session may differ from the ones used in the previous sessions, and that each session may contain several runs. The evolutionary update rule hence compares the performance between the previously received, and the *current* (after the update) network over the *current* training dataset. Consequently, for the proposed MD-PSO evolutionary technique, the swarm particles are randomly initialized (as in the initial evolutionary step), with the exception that the *first particle* has its personal best value, *pbest*, set to the optimal solution found in the previous evolutionary session. That is,

$$\tilde{x}_0^d(0) = \\ = \Psi^d\{\{w_{mn}^0\},\{w_{mn}^1\},\{\theta_n^1\},\dots,\{w_{mn}^{o-1}\},\{\theta_n^{o-1}\},\{\theta_n^o\}\}, \quad (8) \\ \forall d \in [2, L_{max}+1],$$

where Ψ^d is the d-dimensional MLP-configuration retrieved from the previous AS search.

It is expected that, especially at the early stages of the MD-PSO run, the first particle is likely to be the *gbest* particle in every dimension, guiding the swarm towards the previous solution. However, if the training dataset is considerably different in the incremental evolution sessions, it is quite probable that MD-PSO will converge to a new solution, while taking the past solution (experience) into account. In the case of the BP training technique, the weights w_{mn}^l and biases θ_n^l will be initialized with the parameters retrieved from the last AS search. Starting from this as the initial point, and using the current training dataset with the target CVs, the BP algorithm can then perform its gradient descent in the error space.

V. Experimental Results

The audio database used in the classification experiments consists of 367 clips, divided into 8 classes. The database is gathered mostly from the "FreeSound Project" web page [17], but also RWC Music Database [18] was used to collect the music classes. The abbreviations of the used audio classes are as follows: MS (male speech), FS (female speech), FV (female vocals/singing), MV (male vocals/singing), B (bird chirping), W (water sounds), CM (classical music) and GM (general (pop) music). We left the majority of the database clips (75%) for testing, while a training set containing only 25% of the samples from each class was used to evolve the CNBC. For the AS, we used simple ANN configurations with the following range arrays: $R_{\min} = \{N_i, 8, 2\}$ and $R_{\max} = \{N_i, 16, 2\}$, which indicate that, besides a single-layer perceptron (SLP), all the MLPs contain only one hidden layer, i.e. $L_{\max} = 2$, with no more than 16 hidden neurons. The software implementations were made using Visual C++ 6.0 with FFTW library for FFT processing. Parallel processing was utilized in evolving the CNBC classifiers, yielding an approximate CPU time of 1-1.5 h to obtain the classification results with the exhaustive BP method. However, the needed CPU time is highly dependent on the parameters used, i.e. the number of runs, Q_R, and epochs, Q_E, so that the reported CPU time should be considered as suggestive only (for example, with Q_R=1 and Q_E=100, the CPU time decreases to only 10-15 minutes). The processor used was Intel® Core™2 Quad Q9400, 2.66 GHz with 8 Gb of RAM.

For the both evolution methods, exhaustive BP and MD-PSO, the number of runs and training epochs (or iterations in the case of MD-PSO) were varied to see their effect on the classification performance, as is shown in Table I. In order to compare the results with a method representing the current state of the art ([5], [6]), also *SVM* classifiers were tested by applying the *libSVM* library. Results with four different kernels (linear, polynomial, radial basis function (RBF), and sigmoid) were evaluated, for which the best classification accuracy of *89.38%* was obtained. Here a "*one against one"* approach (one SVM for each pair of classes) was applied in evaluating the results for the stated multi-class problem.

The performed CNBC evolutions of Table I are much alike to the (batch) training of traditional classifiers (such as ANNs, K-nearest neighbour, Bayesian), where the training data (the features) and the number of classes are all fixed, and the entire GTD is used during the training (evolution). However, as detailed earlier, the CNBC can be also evolved incrementally, i.e. the evolutions can be performed whenever new features / classes are introduced. For evaluating the incremental evolution performance, the training dataset was divided into *three* distinct partitions, containing 4 (MS, FS, CM, and W), 2 (B and MV) and 2 (*FV* and *GM*) classes, respectively. Three *stages* of incremental evolutions were then performed, where at each stage the CNBC was further evolved using only the dataset belonging to the new classes in the corresponding partition. At the end, the resulting CNBC with $4+2+2=8$ NBCs, encapsulating $8\times(3+1)=32$ BCs within, was created. Verification test between each stage was performed (with the 95% accuracy threshold) to determine whether the existing NBCs needed to be re-trained with the new training samples. A final classification accuracy of 87.38% was achieved, indicating only a moderate loss of performance when compared to the classification accuracies listed in Table I. It is thus evident that the proposed CNBC design can cope up with the incremental evolutions also.

The *confusion matrix (CM)* given in Table II is composed from the classification results of the exhaustive BP evolutions with Q_R = 5, and Q_E = 200. The rows of the CM correspond to the ground truth labels of the classes, whereas the columns indicate the actual classifications results. The average *precision (P)* and *recall (R)* calculated from the CM are:

$P = 0.9151 \qquad R = 0.9005,$

respectively. The overall classification error rate of ~8.8% with 8 classes indicates a substantial level of classification accuracy, considering the quite limited training dataset used (25%), and the somewhat overlapping audio classes with inter-class similarities (e.g. male speech / male vocals). It can be noticed that the music classes are classified perfectly, whereas some

TABLE I. Classification Accuracies of the Both Evolutionary Techniques with Different Number of Epochs and Runs.

		Number of Epochs / Iterations		
	Evol. Method	Q_E =100	Q_E =200	Q_E =300
Q_R =1	**BP**	90.11%	91.07%	91.07%
	MD PSO	87.91%	88.64%	88.64%
Q_R =5	**BP**	90.48%	**91.21%**	**91.21%**
	MD PSO	89.01%	**91.21%**	87.91%

TABLE II. CONFUSION MATRIX OF THE EXHAUSTIVE BP EVOLUTIONS WITH $Q_R = 5$, AND $Q_E = 200$.

Actual Result		**MS**	**FS**	**FV**	**MV**	**B**	**W**	**CM**	**GM**
Ground Truth	**MS**	**27**	4	0	0	0	0	0	3
	FS	0	**22**	2	0	1	0	0	0
	FV	0	0	**39**	0	0	0	1	0
	MV	2	0	1	**32**	0	0	1	0
	B	0	0	0	0	**37**	1	0	0
	W	0	1	0	0	0	**18**	0	7
	CM	0	0	0	0	0	0	**40**	0
	GM	0	0	0	0	0	0	0	**34**

confusion occurs between the speech classes and, rather surprisingly, between the water sound and general pop music classes. Nevertheless, for the tested database, the results are more accurate than those obtained using the "one against one" SVM approach.

VI. CONCLUSIONS

In this paper, a novel CNBC framework was introduced to address the problem of accurate and efficient content-based audio classification within large and dynamic audio databases. The achieved classification results show improvements in accuracy when compared to the tested Support vector machine (SVM) classifiers. Furthermore, two notable advantages can be mentioned on behalf of the proposed approach: First, the dynamic and evolving structure of the framework supports for dynamic variations of audio classes in the considered database, meaning that there is no need to re-train the entire classifier network again whenever new data is added to the database. Second, the optimum classifier for the classification problem at hand can be searched by the underlying evolution technique, which allows creating a dedicated classifier to discriminate a certain class type from the others by using only some specific feature set. This negates the necessity of configuring the classifier parameters and configurations strictly for some specific audio dataset, and thus, hopefully, broadens the usage of the framework for varying type of audio databases.

Future research will be concentrating on supporting larger number of audio classes and developing more descriptive audio features. Due to the structure of the CNBC, it would be ideal to have features with high discrimination power over one (or some particular) class(es). We aim to develop a *perceptual* audio key frame extraction scheme that would take into account the human auditory system. Also, additional classifiers, such as weak classifiers, RBFs and random forests, are planned to be applied within the CNBC network, because of their strong discriminating power reported for certain type of classification tasks in the literature. Finally, content-based indexing and retrieval of audio (and video) clips is a natural way to add more value to the framework, and will be certainly put under study in a near future. Based on the achieved classification accuracy, reliable retrieval results can be expected in the future research.

REFERENCES

[1] E. Wold, T. Blum, D. Keislar, and J. Wheaton, "Content-based classification, search, and retrieval of audio", in IEEE Multimedia Journal, Vol. 3, No. 3, 1996, pp. 27-36.

[2] C. Wu and C. Hsieh, "Multiple change-point audio segmentation and classification using an MDL-based Gaussian model", in IEEE Transactions on Audio, Speech, and Language Processing, Vol. 14, No. 2, March 2006, pp. 647-657.

[3] W. Pan, Y. Yao and Z. Liu, "An unsupervised audio degmentation and classification approach," in Fourth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD), 2007.

[4] S. Kiranyaz, A. F. Qureshi, and M. Gabbouj, "A generic audio classification and segmentation approach for multimedia indexing and retrieval", in IEEE Transactions on Audio, Speech, and Language Processing, Vol. 14, No. 3, May 2006, pp. 1062-1081.

[5] L. Chen, S. Gündüz and M. T. Özsu, "Mixed type audio classification with support vector machine," in IEEE International Conference on Multimedia and Expo (ICME), pp. 781–784, April 2006.

[6] Y. Zhu, Z. Ming and Q. Huang, "SVM-based audio classification for content-based multimedia retrieval", in Proc. of the 2007 International Conference on Multimedia Content Analysis and Mining (MCAM), Weihai, China, 2007, pp.474-482.

[7] W. Chu and B. Champagne, "A noise-robust FFT-based spectrum for audio classification", in Proc. of Acoustics, Speech and Signal Processing (ICASSP), May 2006, pp. V-213-V-216.

[8] E. Ravelli, G. Richard, and L. Daudet, "Audio signal representations for indexing in the transform domain", in IEEE Transactions on Audio, Speech, and Language Processing, Vol. 18, No. 3, March 2010, pp. 434-446.

[9] E. Dogan, M. Sert, A. Yazici, "Content-based classification and segmentation of mized-type audio by using MPEG-7 features", in Proc. of the First International Conference on Advances in Multimedia (MMEDIA), July 2009, pp. 152-157.

[10] G. Peeters, "A generic system for audio indexing: Application to speech/music segmentation and music genre recognition", in Proc. of the 10th Int. Conference on Digital Audio Effects (DAFx-07), Bordeaux, France, September 2007.

[11] H. Harb and L. Chen, "A general classifier based on human perception motivated model", in Multimedia Tools and Applications Journal, Vol. 34, No. 3, 2007, pp. 375-395.

[12] S. Kiranyaz, M. Gabbouj, J. Pulkkinen, T. Ince and K. Meissner, "Network of evolutionary binary classifiers for classification and retrieval in macroinvertebrate databases", in IEEE International Conference on Image Processing, September 2010, pp. 2257 – 2260.

[13] H. Abbass, "Pareto neuro-evolution: Constructing ensemble of neural networks using multi-objective optimization", in the IEEE Congress on Evolutionary Computation 2003, pp. 2074 – 2080 Vol. 3.

[14] S. Kiranyaz, T. Ince, A. Yildirim and M. Gabbouj, "Evolutionary artificial neural networks by multi-dimensional particle swarm optimization", Neural Networks, vol. 22, pp. 1448 – 1462, Dec. 2009.

[15] J. Kennedy, R Eberhart., "Particle swarm optimization", in Proc. of IEEE Int. Conference On Neural Networks, vol. 4, pp. 1942–1948, Perth, Australia, 1995.

[16] S. Kiranyaz, T. Ince, A. Yildirim and M. Gabbouj, "Fractional Particle Swarm Optimization in Multi-Dimensional Search Space", *IEEE Trans. on Systems, Man, and Cybernetics – Part B*, pp. 298 – 319, vol. 40, No. 2, 2010.

[17] The Freesound Project web page, http://www.freesound.org/.

[18] M. Goto, H. Hashiguchi, T. Nishimura, and R. Oka, "RWC music database: Popular, classical, and jazz music databases", in Proc. of 3rd International Conference on Music Information Retrieval, Oct. 2002

Publication P4

Serkan Kiranyaz, Toni Mäkinen, and Moncef Gabbouj, Dynamic and scalable audio classification by collective network of binary classifiers framework: An evolutionary approach. *Neural Networks*, vol. 34, pages 80 – 95, 2012.

Publication P5

Toni Mäkinen, Serkan Kiranyaz, Jenni Pulkkinen, and Moncef Gabbouj, Evolutionary feature generation for content-based audio classification and retrieval. *Proceedings of the 20th European Signal Processing Conference (EUSIPCO)*, pages 1474 – 1478, Bucharest, Romania, August 2012.

EVOLUTIONARY FEATURE GENERATION FOR CONTENT-BASED AUDIO CLASSIFICATION AND RETRIEVAL

Toni Mäkinen, Serkan Kiranyaz, Jenni Pulkkinen, and Moncef Gabbouj, *Fellow IEEE*

Department of Signal Processing
Tampere University of Technology
P.O. Box 553, Tampere, Finland
email: *firstname.lastname@tut.fi*

ABSTRACT

Many commonly applied audio features suffer from certain limitations in describing the data content for classification and retrieval purposes. To remedy this drawback, in this paper we propose an evolutionary *feature synthesis* (EFS) technique, which is applied over traditional audio features to improve their data discrimination power. The underlying evolutionary optimization algorithm performs both feature *selection* and feature *generation* in an interleaved manner, optimizing also the *dimensionality* of the synthesized feature vector. The process is based on multi-dimensional *particle swarm optimization* (MD PSO) with two additional techniques: the fractional global best formation (FGBF) and simulated annealing (SA). The experimented classification and retrieval performances over a 16-class audio database show improvements of up to 11% when compared to the corresponding performances of the original features.

Index Terms— Feature generation, particle swarm optimization, neural networks, content-based classification

1. INTRODUCTION

Content-based audio classification and retrieval is a widely studied topic in the field of signal processing and, especially, machine learning. Since the pioneer work of [1], the development of supervised machine learning techniques has led to more advanced and expanded classification methods, such as the one proposed in [2], where *support vector machines* (SVM) were successfully applied. Statistical models, specifically *Gaussian mixture models* (GMM) and *hidden Markov models* (HMM) have also provided satisfactory classification and retrieval results (see e.g. [3] and [4], respectively). The idea in these is to estimate the probability density function (pdf) for the feature vectors of each predefined audio class. Recently, studies related to environmental sounds and context recognition have also emerged. In [5], environmental sounds were indexed and retrieved successfully in both indoor and outdoor conditions using HMMs and a modified spectral clustering algorithm, whereas in [6] event histogram-based context recognition was proposed with a versatile collection of environmental sounds, providing a recognition rate of 92.4%. In addition to context recognition, several other applications can be mentioned for audio indexing and retrieval, such as advanced database browsing, query-by-example, and highlight spotting.

In this work, a distinct *feature generation* phase is to precede the actual audio classification and retrieval. An early work of feature generation (proposed in [7] for digit recognition) suggested taking the originally extracted features and combining those in a proper manner to produce more descriptive *new* (or transformed) features. A similar fundamental idea was applied by Krawiec and Bhanu in [8], where the term *evolutionary feature synthesis* (EFS) was first adopted to describe the use of evolutionary algorithms, such as *genetic programming* (GP), for feature generation purposes. The idea was to encode *potential* object recognition procedures, while the training process consisted of co-evolving feature extraction procedures, each being a *sequence* of elementary image processing and feature extraction operations. The method avoided recurring to the means commonly used in recognition systems, whereas the obtained recognition ratios themselves were not superior to those achieved by standard methods. In [9] and [10], the idea of feature generation was brought into audio domain, as GP was used to produce new (artificial) audio features. Encouraging results were reported e.g. over music genre classification, although only 4 classes were involved.

In this paper, we propose an evolutionary feature synthesis (EFS) technique to enhance common audio descriptors. The technique uses multi-dimensional particle swarm optimization (MD PSO) [11] to search for the optimal feature synthesis parameters among a predefined search space. An initial work of the method was reported in [12] for image retrieval, whereas here the focus is on audio classification (by support vector machines) and retrieval. An overview of an ideal feature synthesis process is illustrated in Figure 1, in which considerable improvements in feature discrimination can be observed after the synthesis operation. Contrary to the figure, in our approach also the output feature vector *dimension* is optimized, which is a property being omitted in the previous feature generation approaches.

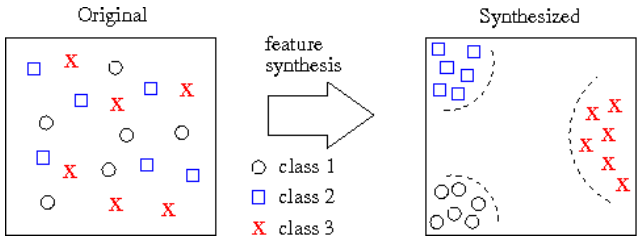

Fig. 1. An illustrative example of an ideal feature synthesis operation over 2-D feature vectors of a 3-class dataset.

The rest of the paper organizes as follows: Section 2 introduces the underlying stochastic optimization algorithm applied in the process, the MD PSO, whereas the feature synthesis process itself is presented in Section 3. The obtained classification and retrieval results are shown in Section 4, and Section 5 concludes the paper.

2. STOCHASTIC OPTIMIZATION ALGORITHM

2.1. Multi-dimensional particle swarm optimization

Particle swam optimization (PSO) was first introduced by Kennedy and Eberhart in [13]. It is a *population-based* optimization technique, in which a swarm of particles propagates iteratively in a predefined search space. After the initialization phase of the algorithm, where the particles are randomly (uniformly) distributed, each particle is evaluated using a proper *fitness function*, $\mathcal{F}[p]$, and moved accordingly within the search space. The ultimate goal is to converge to the global optimum of the search space, for which each particle p has the so-called *social* and *cognitive* terms. The former corresponds to the best position found by the entire swarm (the global best, GB), whereas the latter stands for the best position found by the particle p itself.

In the case of MD PSO, the native PSO operation is extended by allowing the particles to perform *inter-dimensional* jumps within a set dimension range, $d \in [D_{\min}, D_{\max}]$. Thus, the MD PSO searches for the global best solution among *several* search spaces with different dimensions. The dimensional navigation is controlled by a *dimensional* PSO process interleaved with the traditional PSO operations, in which each particle keeps also track of the global and personal best *dimension* (from which the best fitness value so far has been achieved). The pseudo-code and more details about MD PSO can be found in [11].

2.2. Global convergence methods

2.2.1. Fractional global best formation

In order to better avoid local minima during the MD PSO search process, a *fractional global best formation* (FGBF) method [14] is performed within the MD PSO process. The method exploits the potential of individual particle *elements*, evaluating a separate fitness score for each. It then produces a new *artificial global best* (aGB) particle by combining the best elements found from the entire swarm. Whenever the new aGB particle surpasses in fitness the native global best, the aGB is considered as the new global best. In the case of MD PSO, a separate aGB is assigned for *every* dimension. Thus, as illustrated in Figure 2, the aGB particle can be formed by combining elements collected from several dimensions, which further increases the probability of finding aGB particles with improved fitness scores.

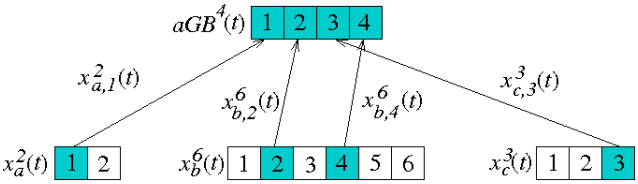

Fig. 2. An illustration of the formation of an aGB particle in dimension 4. Elements of three different particles, a, b, and c (of dimensions of 2, 6, and 3) are combined in the process.

2.2.2. Simulated annealing

As suggested in [15], a *simulated annealing* (SA) algorithm can be utilized within the PSO process to search around the current global best position found by the swarm. In short, after each PSO iteration, a new "neighbor" solution is suggested, which may then replace the current global best. The process is controlled by a specific *temperature* term, T_k, an *update* constant u, and a *cooling* constant, $C < 1$. The number of iterations, $k_{\max}$, needs to be assigned for the SA algorithm, for which the pseudo-code is given in Table 1.

3. EVOLUTIONARY FEATURE SYNTHESIS

3.1. Overview of the system

To meet the objectives assigned for an ideal feature synthesis process, i.e. to perform an optimal feature *selection* and *modification* in an optimal output feature vector dimension, four processing steps are performed. For

Table 1. The SA algorithm in the MD PSO process

1. Randomly distribute the particles into the search space.
2. Evaluate the fitness of each particle, $\mathcal{F}[p]$, $p \in [1, P]$.
3. For $(d = D_{\min};\ d < D_{\max})$ {
   3.1 Set $T_k = T_0$; $k = 0$;
   3.2 While $(k < k_{\max})$ {
      3.2.1 Generate a neighbor solution, nGB^d:
         $nGB^d = GB^d + randn(d) * u$;
         // $(randn(d)$ is a d-dimensional random vector)
      3.2.2 Evaluate the fitness of nGB^d
      3.2.3 Compute $\Delta = \mathcal{F}[nGB^d] - \mathcal{F}[GB^d]$;
      3.2.4 If $(\min[1, \exp(-\Delta/T_k)] > \text{rand}[0,1])$
            Set $GB^d = nGB^d$;
      3.2.5 Set $T_{k+1} = CT_k$; // (C is the *cooling* constant)
   }
}
4. Update the particle positions, see [11] for details.
5. If (PSO iterations left) return to 2.

each new synthesized feature, the system, with a specified *synthesis depth* value K,

1. selects $K+1$ original features $f_0, \ldots, f_K$,
2. scales the selected features with weights $w_0, \ldots, w_K$,
3. selects K operators, $\theta_1, \ldots, \theta_K$, to be applied over the selected and scaled features, and
4. bounds the output with a non-linear operator (here *tangent hyperbolic* is applied).

Suppose $\theta_n(f_a, f_b)$, where $n \in [1, K]$, stands for applying a specific operator θ_n over the features f_a and f_b. Then, a formula for synthesizing a new feature s_j can be defined as

$$s_j = \tanh[\theta_K(\theta_{K-1}(\ldots\theta_2(\theta_1(w_0 f_0, w_1 f_1), w_2 f_2), \ldots), w_K f_K)], \quad (1)$$

that is, first the operator θ_1 is applied to the *weighted* features f_0 and f_1, after which the operator θ_2 is applied to the *result* of the first operation and the weighted feature f_2, and so on. Finally, the operator θ_K is applied to the result of *all* the previous operations and the weighted feature f_K.

The term "evolutionary" applied in this work refers to both the underlying computing technique, the MD-PSO, as well as the nature of the feature synthesis process itself, which can be performed in either one or several *runs*. Here the idea is that each additional run can further synthesize the features from the previous run and further increase the discrimination power. A block diagram of the overall synthesis process is illustrated in Figure 3, where R synthesis runs are performed.

3.2. Particle encoding

In a MD PSO process, the search space dimension, $d \in [D_{\min}, D_{\max}]$, corresponds to the number of features to be synthesized into the output feature vector (FV), that is, the output FV dimension. Each particle position represents a *potential solution* on how to perform the synthesis for the original features. For this, each particle position encapsulates a complete set of synthesis *parameters*: the indices of the selected features, the feature weights, and the selected operators. Accordingly, the j^{th} *element* of the position of a particle p corresponds to a *way* of synthesizing the j^{th} feature of the output feature vector. Thus, each positional element must include the following: $K+1$ feature indices, $K+1$ feature weights, and K operators. For this, the positional elements of each particle are encoded as a vector of length $2K+1$, including $K+1$ "A-type" and K "B-type" components. These define the corresponding synthesis parameters as follows:

$$\begin{aligned} f_n &= \lfloor A_n \rfloor + 1, \quad && n \in \{0, K\}, \\ w_n &= A_n - \lfloor A_n \rfloor, && n \in \{0, K\}, \\ \theta_n &= \lceil B_n \rceil, && n \in \{1, K\}, \end{aligned} \quad (2)$$

where the $\lfloor \cdot \rfloor$ and $\lceil \cdot \rceil$ operators correspond to the *floor* and *ceiling* mathematical integer functions, respectively. The value ranges for the components can be defined based on the *input* feature vector dimension, F, and the total number of operators available, Θ, as $A_n \in [0, F[$ and $B_n \in]0, \Theta]$. The weight values are limited to $0 \leq w_n < 1$.

To give an example of the encoding, Figure 4 presents a particle p position in dimension 6 with the corresponding synthesis process. The synthesis process of the first element of the output feature vector at run r, $FV(r)$, is shown in detail, while a similar process is performed separately for all the output vector elements. For simplicity, the synthesis depth value, K, is set to 3, meaning that $K+1=4$ features, $f_0, \ldots, f_3$, are selected from input feature vector $FV(r\text{-}1)$. Thus, as is demonstrated in the figure, each of the particle elements include $2K+1=7$ encoded synthesis parameters, $A_0, \ldots, A_3$ and $B_1, \ldots, B_3$. The dimension of the input feature vector is $F=8$ and the total number of available operators is $\Theta=5$, meaning that the value ranges for the two component types can be defined as $A_n \in [0,8[$ and $B_n \in]0,5]$. According to Figure 4, the selected features obtained by the underlying MD PSO process are the 7th, 3rd, 1st, and again the 3rd element of the input feature vector, while the corresponding operators are selected as '+', 'min', and '∗'. Thus, performing the synthesis process as given in (1), the 1st element of the output FV is obtained by

$$s_1 = \tanh\left[\min\left(\left(w_0 f_{[7]} + w_1 f_{[3]}\right), w_2 f_{[1]}\right) * w_3 f_{[3]}\right], \quad (3)$$

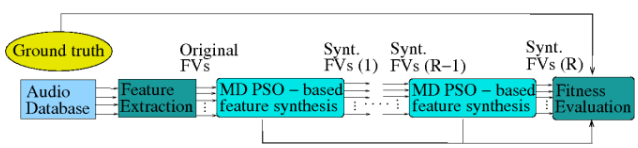

Fig. 3. A block diagram of the proposed EFS with R runs. The arrows correspond to distinct feature vectors (FVs).

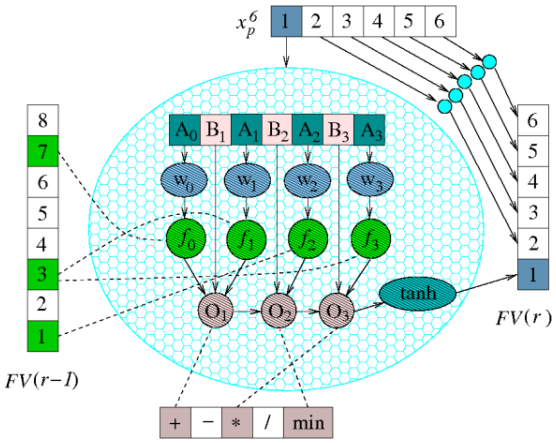

Fig. 4. An example of a particle encoding in a 6-dimensional search space with a synthesis depth set to $K=3$, input feature vector dimension of $F=8$, and the number of operators set to $\Theta=5$.

where $f_{[n]}$ stands for the n^{th} element of the input FV.

Note that by setting $K+1=F$, discarding the feature selection as $f_n = f_{[n]}$, and setting each operator θ_n to '+', the approach becomes identical to a s*ingle-layer perceptron* (SLP) classifier. Also, performing several runs with such synthesis parameters corresponds to a *multi-layer perceptron* (MLP) with a one-to-one analogy between the number of hidden layers and the number of runs performed in the synthesis process. In this sense, it can be stated that a regular feed-forward artificial neural network (ANN) is a *special case* of the proposed synthesis approach.

3.3. The fitness function

The fitness measure for evaluating the discrimination ability provided by the synthesized features plays an important role in the whole synthesis process. Here we propose a measure based on clustering criteria. Suppose the different labels of a L-class database are denoted as $l_0, \ldots, l_{L-1}$, and the corresponding class centroids as $\mu_0, \ldots, \mu_{L-1}$. Then, a discrimination measure (DM) can be defined for a set of synthesized feature vectors, $S = \{s\}$, as

$$DM[S] = FP(S) + \delta_{\text{mean}}(l_n)/\delta_{\min}(l_n, l_m),$$

where

$$\delta_{\text{mean}}(l_n) = \frac{1}{L}\sum_{n=0}^{L-1}\frac{\sum_{\forall s \in l_n}\|\mu_n - s\|}{|l_n|}, \qquad (4)$$

$$\delta_{\min}(l_n, l_m) = \min_{n \neq m}(\|\mu_n - \mu_m\|).$$

The terms of (4) are defined as follows: $FP(S)$ stands for the number of *false positive* feature vectors occurring among the synthesized feature vectors S (meaning that those feature vectors are actually located in a closer proximity to some other class centroid than their own), $\delta_{\text{mean}}(l_n)$ is the average intra-class distance, and $\delta_{\min}(l_n, l_m)$ corresponds to the minimum centroid distance among all the classes. The $DM[S]$ measure strives for minimizing the intra-class distance, while maximizing the shortest inter-class distance. Ideally, minimizing this fitness measure leads to a situation where each synthesized feature vector is in the closest proximity of its own class centroid, thus leading to a high discrimination among classes as illustrated in Figure 1.

4. EXPERIMENTAL RESULTS

The features obtained by the proposed evolutionary feature synthesis (EFS) technique were tested with classification and retrieval experiments. For this, *three* feature vectors, consisting of *commonly applied* low-level audio features (see e.g. [3], [5]),

- STATISTICS (39-D): audio signal statistics (mean, variance, standard deviation, average deviation, skewness, kurtosis) + band energy ratio, spectral centroid, transition rate, fundamental frequency, irregularity, flatness, and tonality,
- MFCC (39-D): 13$^{\text{th}}$ order coefficients + deltas,
- ACOUSTIC (38-D): tri-stimulus, smoothness, spectral spread, spectral roll-off, RMS, amplitude, inharmonicity, spectral crest, loudness, noisiness, power, odd-to-even ratio, and 6 sub-band powers,

were extracted from a general audio database of 1421 clips. Frame features of 40 ms were first extracted, which were then merged and averaged over longer segments to decrease the total number of feature vectors per clip. The database included 16 pre-defined general audio classes of wide range: *male/female speech, male/female singing, whistling, bird sounds, dog barking, fire sounds, breaking glass, classical/rock/techno music, motorcycle sounds, footsteps, applause,* and *crowd cheering*. The samples were gathered from *TIMIT* corpus (speech), *RWC music database* (music), and *StockMusic.com* net store (environmental sounds). Such a database was selected to demonstrate the EFS scalability and performance with several types (and rather high number) of classes. The EFS parameters were *empirically* set to no. of particles $P = 500$, no. of MD PSO iterations $I = 800$, and synthesis depth $K = 5$. The output dimension range was set to $[D_{\min}, D_{\max}] = [24,45]$, while the total number of operators, listed in Table 2 for features f_a and f_b, was set to $\Theta = 18$. Finally, the number of iterations for the SA algorithm was, also empirically, set to $k_{\max} = 25$.

In all the shown experiments, a randomly selected EFS *train* set (45% of the original features) was first used in searching the optimal synthesis parameters (f, w, θ). These were then used to synthesize the actual output features for the *whole* dataset. In the first experiment, the discrimination measure (DM) was evaluated for all the feature vectors before and after the synthesis. The obtained results are presented in Table 3, demonstrating a considerable improvement with all the features (recall that the smaller the DM value, the better the separation between individual classes). This strongly supports the suitability of the EFS to

Table 2. List of operators applied for features f_a and f_b

θ_n	formula	θ_n	formula
0	$-f_a$	**9**	$f_a * f_b$
1	$-f_b$	**10**	$10(f_a * f_b)$
2	$\max(f_a, f_b)$	**11**	f_a / f_b
3	$\min(f_a, f_b)$	**12**	$\sin\big(100\pi(f_a + f_b)\big)$
4	$f_a * f_a$	**13**	$\cos\big(100\pi(f_a + f_b)\big)$
5	$f_b * f_b$	**14**	$\tan(100\pi f_a * f_b)$
6	$f_a + f_b$	**15**	$\tan\big(100\pi(f_a + f_b)\big)$
7	$10(f_a + f_b)$	**16**	$0.5 * \exp\big(-(f_a - f_b) * (f_a - f_b)\big)$
8	$f_a - f_b$	**17**	$0.5 * \exp\big(-(f_a + f_b) * (f_a + f_b)\big)$

Table 3. The obtained discrimination measures (DM) for the original and synthesized features

	STATISTICS	MFCC	ACOUSTIC
Orig. DM	4092	10560	3895
Synth. DM	1086	4948	1290

Table 4. The classification error (CE) statistics obtained by SVM over the original and synthesized features

	STATISTICS	MFCC	ACOUSTIC
Orig. CE (%)	25.4 ± 2.8	11.1 ± 1.5	19.1 ± 2.5
Synth. CE (%)	14.4 ± 1.5	10.7 ± 2.0	17.0 ± 1.9

clustering tasks. Next, classification evaluations were performed in a *five-fold* cross-validation manner, so that every sample in the database was tested during the process. The classification was done using SVMs (libSVM, [16]) with *sigmoid* kernel and different parameter combinations, ranging from $\gamma = \{0.5^1, \dots, 0.5^4\}$ and $C = \{2^0, \dots, 2^3\}$. Such a kernel selection makes the SVM model equivalent to a 2-layer-perceptron ANN. As can be seen in Table 4, the smallest average classification errors demonstrate clear improvements with the synthesized features, especially with the STATISTICS features. The MFCCs are designed to be treated more as "a whole", so that the rather sparse feature selection may diminish the performance obtained with them.

As a final experiment, Table 5 shows the *retrieval* average precisions (AP) obtained for each feature vector. *Precision* stands for the percentage of true positives in the retrieved results, whereas the average was obtained after querying all the database items one by one (by taking the Euclidean distance between the FVs of the query item and each database item). Interestingly, now the MFCCs show an increase of 11% in AP score after 2 EFS runs, and the AP values of other features are also notably improved. Moreover, performing consecutive EFS runs can further enhance the results in most cases. The minor contradiction obtained between the MFCC classification and retrieval performance may suggest that the used fitness function is actually more applicable for clustering than classification.

5. CONCLUSIONS

An evolutionary feature synthesis (EFS) approach for providing discriminative (artificial) features for audio classification and retrieval purposes was proposed. The applied multi-dimensional PSO algorithm proved being able to find the (near-) optimal parameters for the synthesis process. The EFS combines feature selection and feature generation, being also capable of searching the optimal solution among several output vector dimensions. Depending on the original feature vector, the EFS method is capable of improving the classification and retrieval precisions by over 10% with consecutive EFS runs. Testing the method with other databases, classifiers, and fitness functions are potential topics for future research.

Table 5. The average precision (AP) retrieval performances over original and synthesized features

Feature set	Original AP (%)	EFS Run 1 AP (%)	EFS Run 2 AP (%)	EFS Run 3 AP (%)
STATISTICS	44.7	51.4	53.2	54.9
MFCC	35.0	39.4	46.0	44.6
ACOUSTIC	48.5	50.6	52.1	52.6

REFERENCES

[1] E. Wold, T. Blum, D. Keislar, and J. Wheaton, "Content-based classification, search, and retrieval of audio", *in IEEE Multimedia Journal*, vol. 3, no. 3, pp. 27-36, 1996.

[2] G. Guo and S. Z. Li, "Content-based audio classification and retrieval by support vector machines", in *IEEE Trans. Neural Networks*, vol. 14, no. 1, pp. 209-215, 2003.

[3] G. Tzanetakis and P. Cook, "Musical genre classification of audio signals", in *IEEE Trans. Speech and Audio Processing*, vol. 10, no. 5, Jul. 2002.

[4] M. Helén and T. Virtanen, "Audio query by example using similarity measures between probability density functions of features", in *EURASIP Journal on Audio, Speech, and Music Processing*, vol. 2010, ID 179303, 12 p, Jan. 2010.

[5] G. Wichern, J. Xue, H. Thornburg, B. Mechtley, and A. Spanias, "Segmentation, indexing, and retrieval for environmental and natural sounds," in *IEEE Trans. Audio, Speech and Language Processing*, vol. 18, no. 3, pp. 688–707, 2010.

[6] T. Heittola, A. Mesaros, A. Eronen, and T. Virtanen, "Audio context recognition using audio event histograms", in Proc. *18th European Signal Processing Conference (EUSIPCO)*, Aalborg, Denmark, pp. 1272-1276, Aug. 2010.

[7] P.D. Gader and M.A. Khabou, "Automatic feature generation for handwritten digit recognition", in *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 18, no. 12, pp. 1256-1261, 1996.

[8] K. Krawiec and B. Bhanu, "Visual learning by evolutionary feature synthesis," in Int. Conf. on *Machine Learning*, pp. 376-383, Washington, DC, Aug. 2003.

[9] I. Mierswa and K. Morik, "Automatic feature extraction for classifying audio data", in *J. Machine Learning*, vol. 58, no. 2-3, pp. 127-149, 2005.

[10] F. Pachet and P. Roy, "Analytical features: a knowledge-based approach to audio feature generation," *EURASIP Journal on Audio, Speech, and Music Processing*, vol. 2009, Article ID 153017, 23 pages, 2009.

[11] S. Kiranyaz, T. Ince, A. Yildirim, and M. Gabbouj, "Evolutionary artificial neural networks by multi-dimensional particle swarm optimization", in *Neural Networks*, vol. 22, pp. 1448-1462, Dec. 2009.

[12] S. Kiranyaz, J. Pulkkinen, T. Ince, and M. Gabbouj, "Multi-dimensional evolutionary feature synthesis for content-based image retrieval", in Proc. *IEEE Int. Conf. on Image Processing*, Sep. 2011.

[13] J. Kennedy, R Eberhart., "Particle swarm optimization", in Proc. *IEEE Int. Conf. On Neural Networks*, vol. 4, Perth, Australia, pp. 1942–1948, 1995.

[14] S. Kiranyaz, T. Ince, A. Yildirim, and M. Gabbouj, "Fractional particle swarm optimization in multi-dimensional search space", in *IEEE Trans. Systems, Man, and Cybernetics – Part B: Cybernetics*, vol. 40, no. 2, pp. 298-319, April 2010.

[15] F. Zhao, Q. Zhang, D. Yu, X. Chen, and Y. Yang, "A hybrid algorithm based on PSO and simulated annealing and its applications for partner selection in virtual enterprise", in *Advances in Intelligent Computing*, vol. 3644, pp. 380-389, 2005.

[16] C.-C. Chang, C.-J. Lin, "LIBSVM: a library for support vector machines", in *ACM Transactions on Intelligent Systems and Technology*, vol. 2, no. 3, pp. 1-27, Apr. 2011.

Publication P6

Toni Mäkinen, Serkan Kiranyaz, Jenni Raitoharju, and Moncef Gabbouj, An evolutionary feature synthesis approach for content-based audio retrieval. *EURASIP Journal on Audio, Speech, and Music Processing*, vol. 2012, no. 23, pages 1 – 23, 2012.

Mäkinen et al. EURASIP Journal on Audio, Speech, and Music Processing 2012, 2012:23
http://asmp.eurasipjournals.com/content/2012/1/23

EURASIP Journal on
Audio, Speech, and Music Processing
a SpringerOpen Journal

RESEARCH Open Access

An evolutionary feature synthesis approach for content-based audio retrieval

Toni Mäkinen*, Serkan Kiranyaz, Jenni Raitoharju and Moncef Gabbouj

Abstract

A vast amount of audio features have been proposed in the literature to characterize the *content* of audio signals. In order to overcome specific problems related to the existing features (such as lack of discriminative power), as well as to reduce the need for manual feature selection, in this article, we propose an evolutionary feature *synthesis* technique with a *built-in* feature selection scheme. The proposed synthesis process searches for optimal linear/nonlinear *operators* and feature *weights* from a pre-defined multi-dimensional search space to generate a highly discriminative set of new (artificial) features. The evolutionary search process is based on a stochastic optimization approach in which a multi-dimensional *particle swarm optimization* algorithm, along with *fractional global best formation* and heterogeneous particle behavior techniques, is applied. Unlike many existing feature generation approaches, the dimensionality of the synthesized feature vector is also searched and optimized within a set range in order to better meet the varying requirements set by many practical applications and classifiers. The new features generated by the proposed synthesis approach are compared with typical low-level audio features in several classification and retrieval tasks. The results demonstrate a clear improvement of up to 15–20% in average retrieval performance. Moreover, the proposed synthesis technique surpasses the synthesis performance of evolutionary *artificial neural networks*, exhibiting a considerable capability to accurately distinguish among different audio classes.

Keywords: Content-based retrieval, Evolutionary computation, Particle swarm optimization, Feature selection, Feature generation

Introduction

Due to the drastically increased amount of multimedia data available in the Internet and in various public and personal databases, the development of efficient indexing and retrieval methods for large multimedia databases has become a widely studied research topic. Scientific fields, such as digital signal processing (DSP) and computer science (particularly *machine learning*), provide efficient and mathematically well-defined methods for data mining and knowledge discovery from specific observations or databases [1]. One of the major research foci in the field is concentrated around *content-based classification* using supervised learning methods. In these, a dataset together with its perceived class labels, known as "*ground truth*," is used to train a classifier, so that it can learn to discriminate among the individual classes in the training dataset. This enables the classifier to classify new, previously unseen data items with a certain degree of accuracy. Once successful such methods can be applied to several application areas, such as advanced database browsing, query-by-example retrieval, highlight-spotting from movies and/or sport events, speaker recognition, and so on.

In general, whenever machine learning techniques are to be applied to data classification or clustering tasks, certain *features* need to be extracted from the data. The features can be numerical or nominal scalars or vectors describing specific characteristics of the data such as, in the case of audio signals, *tonality* or *fundamental frequency* (FF). Because the data classification and mining methods are strongly dependent on the extracted features, their quality and discriminative capability have an obvious influence on overall classification performance. Unfortunately, despite the enormous number of different audio feature extraction methods available in the

* Correspondence: toni.makinen@tut.fi
Department of Signal Processing, Tampere University of Technology, P.O. Box 553, Tampere, Finland

literature, the features have limitations and drawbacks in describing the data content, so that the current audio classifiers cannot really compete with the human auditory perception system. As will be shortly reviewed, such a lack of semantic representation, the "*semantic gap*," has led to developing several promising techniques to obtain more power of discrimination from the extracted low-level features. The related work in this field is presented in the following section, which focuses on the two most important feature enhancement methods in the literature, *feature selection* and *feature synthesis* (also known as feature *generation/construction/transformation*).

Related work

Generally in machine learning, it is desirable to work with low-dimensional feature vectors (FVs) to reduce computational complexity, and also to avoid the so-called *curse of dimensionality* phenomenon [2], which basically states that in high-dimensional representations the available data become too sparse for any decent statistical or structural analysis. In a feature selection scheme, the FV dimensionality is lowered by selectively choosing an *expressive* and *compact* set of features among a possibly much larger original set. Evolutionary algorithms, such as *genetic algorithms* (GAs) [3] and *genetic programming* (GP) [4], are encountered in several feature selection approaches in the literature (see, e.g., [5,6]). Recently, another population-based stochastic optimization algorithm, *particle swarm optimization* (PSO) [7], was used by Ramadan and Abdel-Kader [8]. They applied PSO to features extracted by discrete cosine transform and discrete wavelet transform. The face-recognition results were comparable to GA-based feature selection but with the benefit of fewer features. Another PSO-based feature selection approach was presented by Chuang et al. [9] in which an *improved binary* particle swarm optimization was applied to a set of gene expression data classification problems. The highest classification accuracy was obtained in 9 out of the 11 tested gene expression problems. It was also reported that the average classification accuracy obtained by a *K*-nearest neighbor classifier was increased by 2.85% when compared to the previously published methods. Other types of classifiers, such as *support vector machines* (SVM) [10,11] and *back-propagation networks* [12], have also been tested with PSO-based feature selection in varying types of classification problems. Finally, in [13], a survey of several other feature selection methods was presented, leading the authors to conclude that "*applying first a method of automatic feature construction yields improved performance and a more compact set of features.*" Hence, research for generating completely new (or modified) features has gained more attention during the past few years.

To date, several feature generation approaches have been proposed [14], which have shown improvements over many types of classification problems. In one of the pioneer works, Markovitch and Rosenstein [15] proposed a framework for feature generation based on a *grammar* consisting of feature construction functions (such as arithmetic and logic operators). In their research, new features were *iteratively* constructed using decision trees, while the evaluation of the framework was done using the Irvine repository of (symbolic) classification problems. Improved classification results were obtained with several tested classifiers when applying them with the original and constructed feature sets (FS). However, such grammar-based methods lack the ability to generalize across more concrete and realistic cases, where, instead of symbols, the input data consists of raw signals. The challenge with signals is that there are no "universally good" features available; rather one has to *manually* choose and extract a specific set of features among the huge amount of existing possibilities. Thus, it can never be guaranteed that the selected features truly represent the optimal set of features for the problem at stake. To address the issue, a fascinating and rather recent idea of *automatic feature generation* has proven to be a promising approach, as it allows going beyond the limitations of human imagination in producing new transformations and (artificial) features.

Automated feature generation approaches are generally based on a "trial and attempt" type of methodology, meaning that *stochastic* searching and optimization algorithms are commonly applied. In [16], a combination of both feature selection and generation was proposed based on a modified GA. The algorithm was applied for the feature transformation process, and an inductive learner was used to evaluate the constructed features on an interpretation of chromatography time series. It was confirmed that one can significantly improve the learning performance when using the constructed features instead of the original time series data. The term "feature synthesis" was first used by Krawiec and Bhanu [17] in the context of object recognition. They applied linear GP to encode potential recognition *procedure synthesis* solutions, expressed in terms of elementary operations. The training consisted of co-evolving feature extraction procedures, each being a *sequence* of elementary image processing and feature extraction operations. The recognition accuracies obtained were comparable to those achieved by standard methods. Bhanu et al. [18,19] continued the work in the context of face expression recognition, where a Gabor-wavelet representation was used for primitive features and linear/nonlinear *operators* were selected among 37 different options to synthesize new features. Each individual in the applied GP algorithm was represented by a binary tree, each of which

corresponded to a single *composite* operator (consisting of several primitive operators). The operator selection was tuned using a Bayesian classifier, and the operator yielding the best classification accuracy was used in synthesizing new FVs for each image in the database. Improved classification accuracy with fewer features was obtained compared to results obtained using the original set of primitive features in the expression recognition task. The work was expanded in [20], where *co-evolutionary* processing was added to the approach to enable using *several sub-populations* in the GP algorithm. In this case, the final FVs were formed by combining the composite features synthesized by each individual subpopulation. The obtained classification results for synthetic aperture radar images showed occasional improvements compared to the primitive features; as before, fewer features were required to obtain comparable recognition rates. However, the authors also conclude that "*... it is still very important to design effective primitive features. We cannot entirely rely on CGP (co-evolutionary GP) to generate good features.*"

Probably the first *audio* feature generation system was one proposed by Pachet and Zils [21]. Their approach uses GP as the core feature generation algorithm in an *extractor discovery system* (EDS) framework, to explore large operator function space and to automatically discover new high-level audio features. The search is guided by specific heuristics, which enable applying knowledge representation schemes about signal processing functions as part of the feature generation process. More recently, Pachet and Roy [22] applied the same EDS framework, where *analytical features* (AF) were also introduced. These represent a large subset of all possible audio DSP functions, and are expressed as a functional term consisting of basic operators. The main idea in [22] is to apply genetic transformations in order to improve the current population of the (first random) AFs, while the fitness of each AF is evaluated using an SVM classifier. The idea of EDS bears some similarities to the framework proposed in [15] and some other feature generation approaches, but differs in providing operator knowledge (such as function *patterns* and *heuristics*) within the process. As a result, improved classification results compared to common audio features were obtained with the AFs in several challenging classification tasks. The authors also participated to the research made in [23] with AFs proposing a method to improve search performance involved in feature generation tasks. The applied algorithm is a variant of *simulated annealing*, guided by the so-called *spin patterns*, which are statistical properties of the feature space. Three audio classification problems were evaluated using the generated features, and significant improvements in execution time were reported when the results were compared to those obtained with features searched using GA, as described in [22].

Generally in audio signal processing, *ad hoc* domain-specific features have also gained considerable attention during the past few years, mainly applied to specific audio classification problems. For example, Mörchen et al. [24] constructed a large set of features by applying *cross products* between several existing short- and long-term (obtained by several aggregation methods) feature functions, resulting in approximately 40,000 audio features in total. It was shown that some of the constructed features could indeed improve music classification performance relative to conventional features. Another such example was presented by Mierswa and Morik [25], where *method trees* consisting of *ad hoc* features for a given audio signal were introduced. The trees were automatically generated with GP by combining elementary feature extraction *methods*. To do this, an additional complexity constraint was applied to keep the computational processing feasible. Improvements were reported in music genre classification accuracy over approaches with traditional audio features. Furthermore, in [26] the same approach was applied to speech emotion recognition with comparable results.

Considering potential drawbacks and uncertainties in the previously proposed feature generation approaches, an important issue relates to the computational time and *complexity* required for the synthesis process. More specifically, generating new FVs with *high dimensionality* may be laborious and time-consuming; for example, in [20], a separate subpopulation needed to be generated for each new generated feature. However, despite the increasing amount of computation required, also high-dimensional representations should be considered when striving to generate efficient new FVs. The *individual feature search* method, introduced in [23], provides a significant contribution to the field in decreasing computation time with respect to GP (by an order of magnitude). Due to the somewhat constrained set of experiments, however, eventually the method shows particularly significant differences to traditional GP mainly at the initial stage of the search, whereas the fitness difference becomes less significant as the search goes on. The allowed search space size in general plays an important role, as it may become too large to be explored efficiently (and throughout). For example, in the case of the AF proposed in [22], it was said that "*the space of 'reasonable size' AFs is huge*" (as it should be to allow capturing a sufficient collection of DSP functions), and, later, that "*eventually the EDS framework reaches the fringe of the space, although it certainly does not explore all of it.*" Now, depending on the case, such partial exploration might cause some lack of performance to the generated features, which is addressed in this article by applying two

dedicated techniques for converging towards the global optimum of the parameter search space. The EDS framework applied and discussed in [22] uses several *heuristics* as a vital component to guide the search. These may complicate the system implementation and, in some cases, cause additional uncertainty or fuzziness (due to stochastic behavior) to the process. Considering the combined feature selection and feature generation methods proposed so far, a *separate* feature selection scheme is generally required as a part of the main system topology, whereas the approach proposed in this article provides feature selection as a built-in property within the underlying PSO algorithm. This makes the overall design and implementation of the method easier, and possibly decreases the number of adjustable parameters. Finally, in some cases (e.g., in [20] or in [24] with most of the cases), it was demonstrated that the generated features cannot always improve the final classification results, which is an issue worth taking into a deeper discussion in order to discover valid reasoning for such synthesis behavior. It could be that the search for the optimal synthesis parameters in a high-dimensional solution space gets trapped into a local optimum, ultimately yielding deficient synthesis results. Another reason could be that the evaluation of the feature quality does not correlate with the actual performance obtained using the features. The problem might also relate to the *manually* selected dimension for the synthesized FV, which may serve as an apparent source of sub-optimality. Nonetheless, in the previously proposed feature generation approaches, the dimensionality issue is only rarely considered and discussed. In [22,27], separate classification experiments with different FS dimensions are performed and compared. Automatic, simultaneous, and *on-going* search and optimization regarding to output FV dimensionality, however, is a novel property provided by the synthesis technique proposed in this article.

The proposed feature synthesis technique

In this article, we aim to overcome the mentioned problems by proposing an evolutionary feature synthesis (EFS) technique based on PSO. The technique is applied for audio feature selection and synthesis. The main motivation for the work is to provide improved content-based audio classification and retrieval performance. To motivate the selection of PSO instead of the generally applied GA (or its derivatives), a comparison of the two algorithms, based on [9], is provided in Table 1. In short, the main advantage of PSO over GAs is that the algorithm provides more profound *intelligent background* [28], and it can be performed more easily than GAs [28]. Also, the computation time of PSO is usually less than for GAs, because all the particles in PSO tend to converge to the best solution rather quickly [29]. The synthesis approach presented in this article provides the ability to apply any fitness measure found appropriate for the final feature task at hand (such as classification). Furthermore, we apply a *multi-dimensional extension* of the basic PSO algorithm (MD PSO, [30]) to allow *dynamic* output FV dimensions. This avoids the need of fixing the dimension of the solution space (corresponding to the dimensionality of the synthesized vector) in advance, which is a property not considered in the audio feature generation methods published before.

In order to better avoid the problem of *premature convergence* related to the traditional PSO, a recent technique, the *fractional global best formation* (FGBF) suggested in [31], is also adopted within the proposed synthesis approach. A preliminary work was presented in [32], in which the performance improvement provided by the approach was tentatively verified in the context of images. Furthermore, in this article, a *heterogeneous particle behavior* approach, recently proposed by Engelbrecht [33], is considered. The approach provides further assistance for the particle swarm to converge to the global optimum of the search space by altering the particle velocity update rules. Hence, finally, as a combination of all the PSO extensions, in this study we propose applying an MD PSO algorithm with FGBF and heterogeneous particle behaviors for audio feature synthesis. To the best of the authors' knowledge, we are not aware that such an approach (or PSO in general) should have been proposed earlier in the audio feature generation field.

Table 1 Comparison of GA and PSO as search algorithms

Property	GA	PSO
Genetic operators	Included	Excluded
Key functions	Crossover	Social particle interaction
	Mutation	Particle velocity updates
Information source	All the chromosomes (the whole population moves in one group)	The *global best* particle (evolution only looks for the best solution)
Update occurrence	Probabilistic (cross-over and mutation *rates*)	All particles are updated after each iteration
Local optimum	Can become easily trapped	Can avoid well the local optima

The rest of the article is organized as follows. The low-level audio features considered in this research are described in "Low-level audio features" section, whereas in "Evolutionary optimization techniques" section, the underlying evolutionary optimization technique, the MD PSO, is introduced in detail. "EFS and selection" section presents an overview and technical details of the proposed feature synthesis approach, and the experimental results with comparative evaluations are shown and discussed in "Experimental results" section. Finally, "Conclusion" section concludes the article and discusses topics for future research.

Low-level audio features

In order to provide some important background information for the ultimate goal of this study, i.e., improving the audio retrieval performance, we start by introducing the applied original (low-level) audio features and the extraction procedures preceding the actual feature synthesis process. As mentioned in "Introduction" section, audio features play an essential role in content-based classification and retrieval tasks, so that selecting and extracting the "correct" features for the problem at hand is of utmost importance. Thus, the major focus is on synthesizing new features from the low-level audio features most typically used in the literature. This provides us a solid "baseline" FSs and allows us to demonstrate the effectiveness of the proposed synthesis technique.

Two separate audio feature extraction approaches are used to evaluate the performance of the feature synthesis with different types of features. In order to detect specific temporal signal characteristics, the considered audio clips are processed in short time *frames* of 40-ms duration. The first approach extracts *segment-based* features, while the second one is based on the so-called "*bag-of-frames*" approach [34], where the features are extracted directly from the short-time frames. The details of the extraction methods are provided in the following sections.

Segment features

The segment features are extracted based on the method introduced in [35]. The *energy levels* of the audio frames are computed, and then compared to the average energy level of the whole audio signal to detect and discard silent audio frames. In the second phase, seven audio FSs (specified on the left column of Table 2) are extracted from the non-silent frames, and consecutive *non-silent* frames are *merged* to form distinct audio segments. Hence, theoretically, an audio signal with no silent sections would be considered as a single segment. However, due to some background noise or environmental acoustics, occasional non-silent frames may occur in the middle of a silent section. To filter out such noisy frames in the process, an empirically determined threshold of *five* consecutive frames was set as a minimum duration for a signal segment. Finally, the actual audio segment features are formed by computing the *mean* (μ) and *standard deviation* (σ) statistics of *each* FS (including also the STAT features, i.e., means of means, etc.) over the formed segments. Thus, as an example, an audio signal consisting of four separate segments is represented by four corresponding segment FVs of each FS. For a more detailed description of the segment feature extraction, the reader is referred to [35].

Table 2 The extracted low-level audio features

Segment features	Key-frame features
STAT[a] (39-D)	MFCC + Δ-MFCC + ΔΔ-MFCC (39-D)
13 Mel-frequency cepstral coefficients (MFCC) (26-D)	12th-order LPC + 14th-order LPCC (26-D)
13 Δ-MFCC (26-D)	K_AUDIO[c] (31-D)
13 ΔΔ-MFCC (26-D)	
10th-order linear prediction coefficients (LPC) (20-D)	
14th-order linear prediction cepstral coefficients (LPCC) (28-D)	
S_AUDIO[b] (38-D)	

[a]STAT includes the mean (μ) and standard deviation (σ) values of signal *statistical features*, both in time and frequency domain: mean, variance, standard deviation, average deviation, skewness, kurtosis, and *also* the following *segment* features (μ,σ): band-energy ratio (BER), spectral centroid, transition rate, FF, irregularity (2 versions), flatness (both in linear and decibel scale), and tonality.
[b]S_AUDIO includes the following *segment* features (μ,σ): tristimulus, smoothness, spectral spread, spectral roll-off, RMS amplitude, inharmonicity, spectral crest, loudness, noisiness, power, odd-to-even ratio, and sub-band powers of six frequency bands.
[c]K_AUDIO includes the following *key-frame* features: irregularity (two versions), tristimulus, smoothness, spectral spread, zero-crossing rate, spectral roll-off, loudness, flatness (linear and decibel scale), tonality, noisiness, RMS amplitude, inharmonicity, spectral crest, odd-to-even ratio, spectral slope, FF, skewness, kurtosis, spectral skewness, spectral kurtosis, and 7-band sub-band powers.

Key-frame features

In the second feature extraction approach, the features are extracted directly from the short time frames. Due to this, the frames are first *Hamming*-windowed to avoid sharp discontinuities at the frame edges. Because there are many frames already in a single audio clip, a specific *key-frame* extraction method, proposed in [36], is applied to reduce the most *redundant* frames. Such redundancy occurs because many audio classes, such as music or speech, contain similar and almost identical sounds (such as common vowels or same notes of an instrument). In short, the main idea in the frame reduction approach is to partition the extracted frame features into distinct *clusters* (based on their similarity/distance between each other) and to select only one or few *key-frames* from each cluster to represent its corresponding sound. For this, a *minimum spanning tree* clustering algorithm is applied, which is detailed in [36]. As an outcome of the procedure, the overall amount of frames is

significantly decreased, whereas most of the feature description power is still maintained. The three frame-level FSs used in this study are listed on the right column of Table 2.

The extracted features of Table 2 represent an inclusive set of common audio features found from the literature. The feature implementations are based on a publicly available "*LibXtract*" library [37], although some of the segment features, such as dominant BER and segment FF, are extracted as described in [35]. The dimensions of the extracted FSs/vectors are also given in parenthesis for all sets in Table 2. For example, taking the mean and standard deviation of the 13th-order segment MFCC features results to a 26-dimensional FV. The feature values of each segment/key-frame—as well as their corresponding ground truth class labels and clip indices (describing from which audio file a particular FV is extracted from)—are stored in a single plain text file, so that the actual audio files are not needed anymore after the feature extraction phase. For specific definitions and formulas for the extracted features, the reader is referred to the audio signal processing literature (see, e.g., [38-40]).

Evolutionary optimization techniques

In this section, the PSO algorithm, its multi-dimensional extension, and the FGBF technique are introduced. The adaptation of *heterogeneous* particle behaviors within the MD PSO process is discussed in detail at the end of the section.

MD PSO

PSO was first introduced by Kennedy and Eberhart [7]. The PSO algorithm is a *population-based* optimization technique, in which a swarm of particles propagates in a pre-defined search space. Each individual particle, p, of the swarm represents a *potential solution* to an underlying optimization problem, in which the particles are evaluated using a proper *fitness function*, $\mathcal{F}[p]$. The PSO algorithm is specifically designed for solving nonlinear optimization problems. Due to the *diversity* associated with randomly distributed particles, the algorithm is capable of searching the best solution among several local minima. After the initialization phase of the algorithm, where the particle randomization is performed, the particles are evaluated and moved iteratively in the search space. In order to eventually converge to the global optimum of the search space, each particle p holds in its memory both *social* and *cognitive* terms, where the former corresponds to the best position found so far by the entire swarm (the global best) and the latter stands for the best position found by the particle p itself (personal best). As will be shortly seen, both the social and cognitive terms contribute in a *stochastic manner* to the particle position at the next iteration round.

An MD PSO algorithm was proposed in [30]. The algorithm allows the particles to make inter-dimensional jumps and visit any dimension, d, within a given range, $d \in [D_{\min}, D_{\max}]$. Thus, in order to provide improved fitness scores, the MD PSO searches for the global best solution among *several* search spaces with different dimensions. The particle navigation among the dimensions is controlled by a separate *dimensional* PSO process, which is interleaved with the regular positional update process. For this, each particle keeps also track of its personal (and the global) best dimension (from which the best fitness value so far has been achieved).

In a MD PSO process, the components of each particle p at iteration round t in a swarm of P particles are presented as,

$d_p(t)$: *dimension* of particle p,
$x_{p,j}^{d_p(t)}(t)$: j^{th} element of the *position* of particle p in dimension $d_p(t)$,
$v_{p,j}^{d_p(t)}(t)$: j^{th} element of the *velocity* of particle p in dimension $d_p(t)$,
$y_{p,j}^{d_p(t)}(t)$: j^{th} element of the *personal best position* of particle p in dimension $d_p(t)$,
$vd_p(t)$: *dimensional velocity* of particle p,
$yd_p(t)$: *personal best dimension* of particle p,
$\hat{y}_j^d(t)$: j^{th} element of the *global best position* of the whole swarm in dimension d, j, $\in [1,d]$
$\hat{y}d(t)$: *global best dimension* of the whole swarm,

where (if not stated otherwise) $j \in \{1,\ldots,d_p\ (t)\}$. The particle fitness values are only evaluated within its current dimension, meaning that the *positional* PSO components in all other dimensions remain the same for the next iteration round $t+1$, that is, $x_p^d(t+1) = x_p^d(t)$, $v_p^d(t+1) = v_p^d(t), y_p^d(t+1) = y_p^d(t), \forall d \in [D_{\min}, D_{\max}] \wedge d \neq d_p(t)$. After computing the fitness score of each particle position with the applied fitness function $\mathcal{F}$, the following update equations are used for the personal best position and

$$y_p^{d_p(t+1)}(t+1) = \begin{cases} y_p^{d_p(t+1)}(t), & \text{if } \mathcal{F}\left[x_p^{d_p(t+1)}(t+1)\right] > \mathcal{F}\left[y_p^{d_p(t+1)}(t)\right] \\ x_p^{d_p(t+1)}(t+1), & \text{else,} \end{cases} \quad (1)$$

dimension of particle p for iteration $t+1$:
and

$$yd_p(t+1) = \begin{cases} yd_p(t), & \text{if } \mathcal{F}\left[x_p^{d_p(t+1)}(t+1)\right] > \mathcal{F}\left[y_p^{yd_p(t)}(t)\right] \\ d_p(t+1), & \text{else.} \end{cases} \tag{2}$$

Furthermore, for each dimension $d \in [D_{\min}, D_{\max}]$, the global best particle position is updated as

$$\hat{y}^d(t+1) = \begin{cases} \hat{y}^d(t), \text{if } \min\limits_p \left(\mathcal{F}\left[y_p^d(t+1)\right]\right) \geq \mathcal{F}\left[\hat{y}^d(t)\right] \\ \underset{y_p^d(t+1)}{\text{argmin}} \left(\mathcal{F}\left[y_p^d(t+1)\right]\right), \text{else,} \end{cases} \tag{3}$$

and, finally, the global best dimension is updated as

$$\hat{y}d(t+1) = \begin{cases} \hat{y}\,d(t), \text{if } \min\limits_p \left(\mathcal{F}\left[y_p^{yd_p(t+1)}(t+1)\right]\right) \geq \mathcal{F}\left[\hat{y}^{\hat{y}d(t)}(t)\right] \\ \underset{yd_p(t+1)}{\text{argmin}} \left(\mathcal{F}\left[y_p^{yd_p(t+1)}(t+1)\right]\right), \text{else,} \end{cases} \tag{4}$$

where, in both (3) and (4), $p \in [1, P]$.

The particle positions within the current dimension $d_p(t)$ are updated after each iteration as shown in (5), where $w(t)$ is a so-called *inertia weight*, c_1 and c_2 are pre-determined constants, $\bar{r}_1$ and $\bar{r}2$ are vectors of uniformly distributed random variables, that is $r_{1,j}, r_{2,j} \sim U(0,1), \forall j \in \left[1, d_p(t)\right]$, and $\mathbb{C}(\bar{\boldsymbol{z}}, [Z_{\min}, Z_{\max}])$ works as a *clamping* operator that limits the elements of vector $\bar{\boldsymbol{z}}$ between the specified values $Z_{\min}$ and $Z_{\max}$. Typical PSO parameters [41] were used in this study, that is, the inertia weight was linearly decreased from 0.9 to 0.4, and c_1 and c_2 were both set to 2. The limiting values for the particle position, velocity, and dimensional velocity, *X*, *V*, and *VD*, respectively, were empirically set into proper values, as will be discussed later in "Experimental results" section. Note that the new particle position, $x_p^{d_p(t)}(t+1)$, remains in the current dimension $d_p(t)$ after the positional update, whereas the dimension may change afterwards in the dimension update process defined in (6), which is performed at the end of each iteration round. The $\lfloor . \rfloor$ operator in (6) stands for a floor function, and r_1 and r_2 are now *scalar* uniformly distributed random variables; otherwise the update is performed similarly to the positional updates. For an interested reader, the pseudo-code and further details of the MD PSO are provided in [30].

$$\begin{aligned} v_{p,j}^{d_p(t)}(t+1) &= w(t) v_{p,j}^{d_p(t)}(t) + c_1 r_{1,j}(t)\left(y_{p,j}^{d_p(t)}(t) - x_{p,j}^{d_p(t)}(t)\right) + c_2 r_{2,j}(t)\left(\hat{y}_j^{d_p(t)}(t) - x_{p,j}^{d_p(t)}(t)\right) \\ \underset{x}{p,j d_p(t)}(t+1) &= x_{p,j}^{d_p(t)}(t) + \mathbb{C}\left(v_{p,j}^{d_p(t)}(t+1), [V_{\min}, V_{\max}]\right) \\ x_{p,j}^{d_p(t)}(t+1) &\leftarrow \mathbb{C}\left(x_{p,j}^{d_p(t)}(t+1), [X_{\min}, X_{\max}]\right) \end{aligned} \tag{5}$$

$$\begin{aligned} vd_p(t+1) &= \lfloor vd_p(t) + c_1 r_1(t)\left(yd_p(t) - d_p(t)\right) + c_2 r_2(t)\left(\hat{y}d(t) - d_p(t)\right)\rfloor \\ d_p(t+1) &= d_p(t) + \mathbb{C}\left(vd_p(t+1), [VD_{\min}, VD_{\max}]\right) \\ d_p(t+1) &\leftarrow \mathbb{C}\left(d_p(t+1), [D_{\min}, D_{\max}]\right) \end{aligned} \tag{6}$$

FGBF algorithm

In some cases, the (MD) PSO algorithm suffers from the so-called *premature convergence* problem, meaning that the global best particle traps into a local minimum in the search space. This is especially true in high-dimensional and multi-modal search spaces, which are often encountered in real-world applications. The problem is mainly caused by the loss of diversity, meaning that all the particles are clamped too close to each other in the search space. A recent method called FGBF [31] has been proposed to tackle this problem by exploiting the potential of individual particle *elements* of each particle position. The idea in the technique is to evaluate a separate fitness score for each particle element, in order to form an *artificial global best* (*aGB*) particle by combining the best elements found from the entire swarm. The formed *aGB* particle is then applied whenever it surpasses in fitness the regular global best particle, $\hat{y}^d(t)$, of the swarm. For MD PSO, this means that a separate *aGB* particle needs to be assigned for each search space dimension, $d \in [D_{\min}, D_{\max}]$. However, in this case the *aGB* particle of a particular dimension can be also formed by combining particle elements from dimensions *other* than the current one. This is demonstrated in Figure 1, where elements from three different particles from separate dimensions,

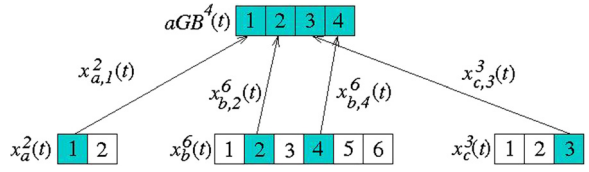

Figure 1 An illustration of the formation of an *aGB* particle dimension of 4. Elements of three different particles, *a*, *b*, and *c*, having the dimensions of 2, 6, and 3, respectively, are brought together and combined in the formation process. Note that several elements can be taken from a single particle.

$x_{p,j}^{d_p(t)}(t)$, are combined to form the *aGB* particle. Such an approach increases the probability of finding an *aGB* particle with a higher fitness score than the existing $\hat{y}^d(t)$ solution for the dimension d at stake. For the sake of clarity, pseudo-code for the FGBF approach within the MD PSO algorithm is shown in Algorithm 1.

Algorithm 1 Pseudo-code of the FGBF algorithm in MD PSO

Let $[j] = \arg\min_{p\in[1,P]}\mathcal{F}\left[x_{p,j}^{d_p(t)}(t)\right]$, then

1. Select the best particle indices for each element, $b[j]$, where $j \in [1,D_{\max}]$, among all the particles, $p\in[1,P]$.
2. For ($d\in[D_{\min}, D_{\max}]$) do:
   a. Assign the best elements into the *aGB* solution: $aGB_j^d(t) = x_{b[j],j}^{d_{b[j]}(t)}, j \in [1,d]$.
   b. If $\left(\mathcal{F}\left[aGB^d(t)\right] < \mathcal{F}\left[\hat{y}^d(t)\right]\right)$, then $\hat{y}^d(t) = aGB^d(t)$.
3. Re-evaluate: $\hat{y}d(t) = \arg\min_d\mathcal{F}\left[\hat{y}^d(t)\right]$.

Heterogeneous particle behaviors

As proposed recently by Engelbrecht [33], variation of the particle *behaviors*, i.e., the velocity update rules, is another efficient way to enhance the convergence ability of the swarm in highly multi-modal problems. In addition to the particle velocity update equations (5) and (6), *four* other update models are introduced in this section. The models are extended to be used with the MD PSO approach so that the corresponding dimensional update rules are changed accordingly. Whenever a particle seems to get stuck into a local optimum, a new behavior model is assigned to it in a random manner. As well, the initial behaviors are chosen randomly for each particle.

Cognitive-only MD PSO model

In the cognitive-only MD PSO model [42], as the name suggests, the *social* terms $\hat{y}^{jd_p(t)}(t)$ and $\hat{y}d(t)$ of the particles (i.e., the latter terms of (5) and (6) with the c_2 coefficients) are *removed* from the velocity update equations. This leads to broader particle exploration as interaction among particles ceases. This, instead, causes every particle to become an *independent* hill-climber in the search space. Thus, whenever the particle position is updated using this model, the (artificial) global best particle position is not considered in the update process.

Social-only MD PSO model

Like the previous model, in the social-only velocity rules [42], the *cognitive* terms $y_{p,j}^{d_p(t)}(t)$ and $yd_p(t)$ of the particles (i.e., the former terms of (5) and (6) with the c_1 coefficients) are removed from the velocity update equations. Such a behavior provides faster particle exploitation, as now the entire swarm becomes a single *stochastic* hill-climber.

Barebones MD PSO

The Barebones PSO was suggested in [43], and in this model the velocity update rule is replaced by

$$v_{p,j}^{d_p(t)}(t+1) \sim \mathcal{N}\left(\frac{y_{p,j}^{d_p(t)}(t) + \hat{y}_j^{d_p(t)}(t)}{2}, \sigma\right), \tag{7}$$

where $\sigma = \left|y_{p,j}^{d_p(t)}(t) - \hat{y}_j^{d_p(t)}(t)\right|$. The position update changes to $x_{p,j}^{d_p(t)}(t+1) = v_{p,j}^{d_p(t)}(t+1)$, so that the velocity ends up being the new position of the particle, sampled from the described Gaussian distribution $\mathcal{N}$. Similarly, for the particle dimensional velocity, the following equation is applied

$$vd_p(t+1) \sim \mathcal{N}\left(\frac{yd_p(t) + \hat{y}d(t)}{2}, \sigma\right), \tag{8}$$

where $\sigma = \left|yd_p(t) - \hat{y}d(t)\right|$. Again, the dimension velocity is considered as the actual new dimension, i.e., $d_p(t+1) = vd_p(t+1)$. Note that the clamping operation is still applied to the obtained positions, so that the set limiting values are not exceeded.

In the positional point of view, the Barebones MD PSO facilitates an *initial exploration*, because at first the personal best positions are far away from the global best solution, causing large deviations to the Gaussian distribution. However, as more iterations are performed, the deviation approaches to zero, causing the behavior to focus on exploitation of the average of the personal best and global best positions.

Modified barebones MD PSO

A modified version of the Barebones, also suggested in [43], is defined as

$$v_{p,j}^{d_p(t)}(t+1) = \begin{cases} y_{p,j}^{d_p(t)}(t), & \text{if } U(0,1) < 0.5 \\ \mathcal{N}\left(\frac{y_{p,j}^{d_p(t)}(t) + \hat{y}_j^{d_p(t)}(t)}{2}, \sigma\right), & \text{else} \end{cases} \tag{9}$$

and for dimensional update, similarly, as,

$$vd_p(t+1) = \begin{cases} yd_p(t), & \text{if } U(0,1) < 0.5 \\ \mathcal{N}\left(\frac{yd_p(t) + \hat{y}d(t)}{2}, \sigma\right), & \text{else.} \end{cases} \tag{10}$$

Such a modification increases the exploration during the initial stages of the search process (compared to

regular Barebones), as now 50% of the time the focus is on the personal best solutions. As the process converges, the behavior will turn to exploitation, when all of the personal best solutions converge towards the global best solution.

As mentioned, each particle obtains its starting behavior randomly among the introduced models (including the regular one), after which the behavior is changed whenever a particle cannot improve its fitness score for the last *ten* consecutive iteration rounds. The whole idea here is that a new behavior model may help the particle to step out from a possible local optimum, and hence eventually provide improvements to the particle fitness score.

EFS and selection

As earlier discussed, the motivation behind proposing the EFS technique is to obtain enhanced audio features, so that audio classification and retrieval performance can be improved. In this section, we will describe the proposed feature selection and synthesis technique in detail. It will be shown that, with a proper encoding scheme (encapsulating several linear/nonlinear operators and the selected original features with their weights), the MD PSO particles can perform an evolutionary search towards finding the optimal synthesis parameters and output vector dimension. For this, a proper fitness measure is to be designed, which maximizes the overall classification (or retrieval) performance. The fitness functions applied in this study for evaluating the particle swarm performance during the synthesis procedure are introduced at the end of the section.

Definition of the main objectives

In an ideal case, a feature synthesis system, also called here as a *feature synthesizer*, receives as its input a specific set of (low-level) audio features, selects the most representative and appropriate subset among them, combines and modifies the features by applying a proper set of transformation operators and feature weights, and finally produces a set of new and descriptive features in an optimal dimension with respect to the fitness function assigned for the problem. Such an ideal feature synthesis operation (for the purpose of clustering) is demonstrated in Figure 2, where two-dimensional features of a 3-class dataset are successfully synthesized into clearly distinct clusters, enabling a much easier classification and/or retrieval task compared to the original feature distribution. Note that, unlike in the figure, the proposed feature synthesis approach allows the output dimension to differ from the original dimension.

Changing the output dimensionality makes the approach somewhat similar to SVM, which attempt to transform the original features into a higher dimension

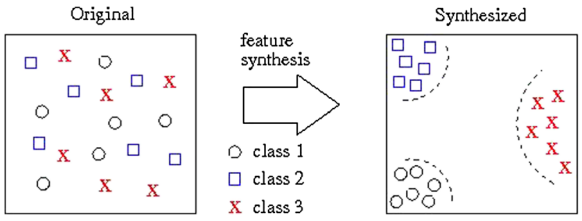

Figure 2 An example of an ideal feature synthesis performed to 2D FVs representing three classes. The synthesized features are much more distinct from each other than the original ones.

to enable linear separation. However, a drawback with SVMs is their high dependency on the applied kernel function and the corresponding internal parameters, which may not fit to the problem at hand properly. This phenomenon is demonstrated in Figure 3, where two successful sample feature synthesizers are presented for a two-class classification problem. The upper case corresponds to an SVM with a *polynomial* kernel in a quadratic form. It is indeed capable of performing a proper transformation from 2D to 3D, enabling thus a linear separation between the two classes. However, in the lower case, a sinusoid with a proper angular frequency, ω, needs to be applied instead for satisfactory class discrimination. Hence, it can be seen that searching for the correct transformation (instead of applying a fixed kernel) is of paramount importance, and this is actually considered as one of the main motivations for designing the feature synthesis scheme proposed in this article.

In light of the above discussion, the main objectives of the proposed EFS technique are to

- perform a proper *feature selection* among the original features,

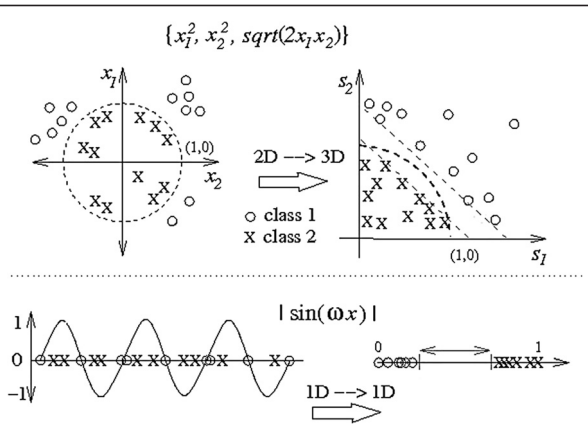

Figure 3 Two examples of feature synthesis (or transform) performed on 2D (upper case) and 1D (lower case) feature spaces using SVM. Depending on the original data distributions, different transformation functions are needed for successful feature separation.

- search for the optimal *operators* and *feature weights* for the synthesis process, and
- search for the optimal output FV dimension among a defined dimension range.

Overview of the proposed feature synthesis system

To meet the aforementioned objectives, an evolutionary search procedure is performed. For each new synthesized feature (meaning a specific *single* feature in the generated output FV), the system, with a specified *synthesis depth* value K,

1. selects $K+1$ original features $f_0,\ldots f_K$, ,
2 scales the selected features with proper weights $w_0,\ldots,w_K$,
3. selects K operators, $\theta_1,\ldots,\theta_K$, to be applied over the selected and scaled features, and
4. bounds the output with a nonlinear operator (here *tangent hyperbolic* is applied).

Now, suppose that $\theta_n(f_a,f_b)$, where $n\in[1,K]$, stands for performing a specific operator θ_n over the features f_a and f_b. Then, a formula for synthesizing a new feature s_j can be defined as

$$s_j=\tanh[\theta_K(\theta_{K-1}(\ldots\theta_2(\theta_1(w_0f_0,w_1f_1),w_2f_2),\ldots),w_Kf_K)], \tag{11}$$

that is, first the operator θ_1 is applied to the *weighted* features f_0 and f_1, after which the operator θ_2 is applied to the *result* of the first operation and the weighted feature f_2, and so on. Finally, the operator θ_K is applied to the result of *all* the previous operations and the weighted feature f_K. With the details provided in "Encoding of the particles" section, the dimension of the synthesized FV, $\bar{s}$, along with the rest of the parameters in (11) are simultaneously optimized within the applied MD PSO search process.

In this article, the term "evolutionary" refers both to the underlying computing technique, the MD PSO, as well as to the nature of the feature synthesis process itself, which can be performed in one or several *runs*. The idea here is that each new run can further synthesize the features generated at the previous run and, hopefully, further increase their discrimination power. A block diagram of the overall synthesis process is illustrated in Figure 4, where R synthesis runs are performed. The total number of runs, R, can either be determined in advance or *adaptively*, in which case the fitness evaluation results are monitored after each run, and the process is stopped after no significant improvement is obtained anymore. Whether there should be more than a single set of features to be synthesized, an *individual* feature synthesizer will be evolved for *each* FS. This is done in order to decrease the computational time needed for the overall processing, as this enables synthesizing the FSs *in parallel* by separate processes.

In a sense, the proposed EFS technique can be seen as a *generalized form* of *artificial neural networks* (ANN). Considering the four system steps listed above, a *single-layer perceptron* (SLP) classifier performs only steps 2 and 4, as neither feature nor operator selection is involved in the process. Instead, SLP does add a bias value to its weighted features, which can be mimicked also in our approach by inserting an additional constant value of *1* at the end of each original input FV (which the synthesizer can then select and scale among the other selected features). However, as no notable performance gain was witnessed by performing such an action, in the end the bias encoding is not considered in the proposed EFS technique. For further comparison, note that in the SLP topology the output layer dimension is fixed, whereas in the EFS the output dimension is (as mentioned) optimized within the set range. Also notice that performing several consecutive EFS runs corresponds to a *multi-layer perceptron* (MLP), or, in fact, any feed-forward ANN type. Similarly to SLP, the MLP does not include feature selection, and it also performs with fixed operator and output dimension. An important difference between the MLP and EFS approaches is that in the EFS technique the fitness of the synthesized FV is evaluated after *each run*, whereas in MLPs only the *final* fitness score in the output layer is considered, as the intermediate network layers are "hidden."

Encoding of the particles

Recalling the PSO definitions introduced in "MD PSO" section, the position of a $d_p(t)$-dimensional particle p at

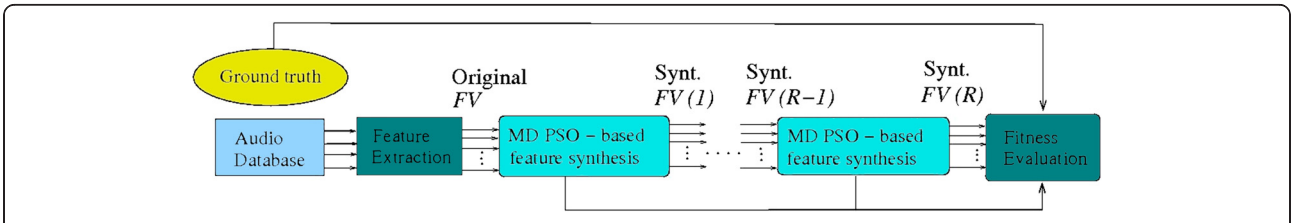

Figure 4 A block diagram of the proposed EFS approach with *R* runs. Different arrows correspond to parallel synthesis processes with different FSs. The fitness evaluation is performed with a specified fitness function.

time t, $x_p^{d_p(t)}(t)$, represents a *potential solution* on how to perform the synthesis operation over the original features, that is, a *potential synthesizer*. The search space dimension, $d_p(t)$, corresponds to the number of features synthesized into the output FV, that is, the output FV dimension. Each particle position encapsulates a complete set of synthesis *parameters*: the indices of the selected features, the feature weights, and the selected feature operators. Accordingly, each *positional element* of a particle p, $x_{p,j}^{d_p(t)}(t)$, where $j \in [1, d_p(t)]$, corresponds to a way of synthesizing the jth feature of the output FV. Thus, referring to the previously introduced four system steps, each such element must contain the following: $K+1$ indices for selecting the original features, $K+1$ feature weights, and K operators in an encoded form to synthesize the corresponding output feature. For this, the positional elements of each particle in the particle swarm are encoded in a *vector form* of length $2K+1$, including $K+1$ "A-type" and K "B-type" components. These define the corresponding synthesis parameters as follows:

$$\begin{aligned} f_n &= \lfloor A_n \rfloor + 1, n \in \{0, K\}, \\ w_n &= A_n - \lfloor A_n \rfloor, n \in \{0, K\}, \\ \theta_n &= \lceil B_n \rceil, n \in \{1, K\}, \end{aligned} \tag{12}$$

where the $\lfloor \cdot \rfloor$ and $\lceil \cdot \rceil$ operators correspond to the *floor* and *ceiling* mathematical integer functions, respectively. The value ranges for the components can be defined based on the *input* FV dimension, F, and the total number of operators available, Θ, as $A_n \in [0, F[$ and $B_n \in]0, \Theta]$. The weight values are limited to $0 \le w_n < 1$.

To give an example of the particle encoding scheme, Figure 5 presents a 6D particle p with the corresponding synthesis process. Note that only the synthesis process of the first element of the output FV at run r, FV(r), is shown in detail, although a similar process is performed for all the output vector elements. For simplicity, the synthesis depth value, K, is set to 3, meaning that only $K+1=4$ features, $f_0,\ldots,f_3$, are selected from input FV($r-1$). Thus, as demonstrated in the figure, each of the particle elements includes $2K+1=7$ encoded synthesis components, $A_0,\ldots,A_3$ and $B_1,\ldots,B_3$. The dimension of the input FV (which may either refer to the original FV consisting of a specific set of low-level features, or to an output FV from a previous EFS run, $r-1$) is $F=8$. As the total number of operators is set to $\Theta=5$, the value ranges for the two component types can be defined as $A_n \in [0, 8[$ and $B_n \in]0, 5]$. In Figure 5, the selected features obtained by the underlying MD PSO process are the 7th, 3rd, 1st, and again the 3rd element of the input FV, while the corresponding operators are selected as '+', 'min', and '$*$' Thus, performing the synthesis process as given in (11), the first element of the output FV is obtained by

$$s_1 = \tanh\left[\min\left(\left(w_0 f_{[7]} + w_1 f_{[3]}\right), w_2 f_{[1]}\right) * w_3 f_{[3]}\right], \tag{13}$$

where $f_{[n]}$ stands for the nth element of the input FV.

Considering again the similarities of the EFS technique and an SLP classifier, it can be noticed that by setting

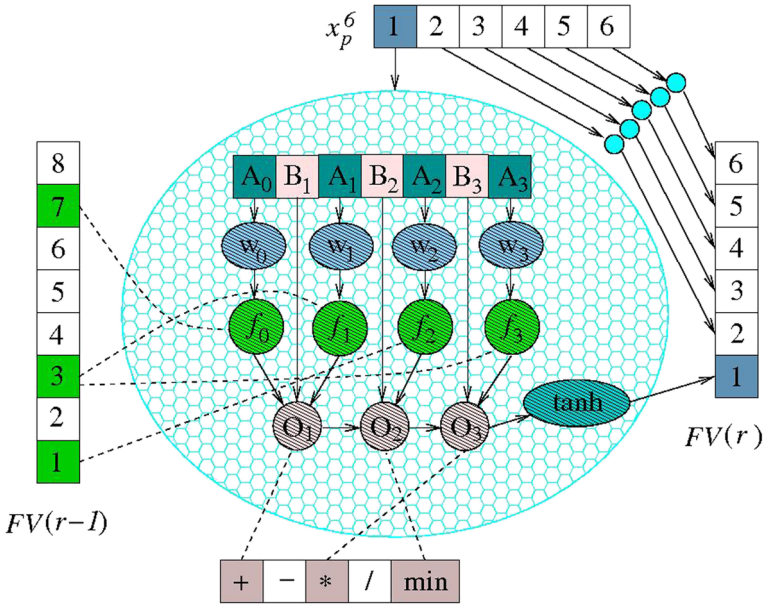

Figure 5 An example of a particle encoding in a 6D feature space with a synthesis depth set to $K=3$, an input FV dimension of $F=8$, and the number of operators defined as $\Theta=5$.

$K+1=F$, discarding the feature selection as $f_n = f_{[n]}$, and setting each operator θ_n to '+', the two approaches become identical. Similarly, by performing several runs with such synthesis parameters corresponds to an MLP approach with a one-to-one analogy between the number of hidden layers and the number of runs performed. In this sense, it can be stated that a regular feed-forward ANN is a *special case* of the proposed EFS approach.

The fitness measures

Proper designing of the applied fitness measure plays an essential role in the feature synthesis process. The design is dependent on the intended use of the features, as the measured fitness value should highly relate to the objective of the synthesis process. Traditionally in content-based classification and retrieval scheme, specific *similarity measures*, such as Euclidean distance, are applied to measure the distances between the FVs of a classified (or queried) item and each item belonging to the database. In a retrieval case, the performance can be evaluated using the *average precision* (*AP*) metric, or the so-called *average normalized modified retrieval rank* (*ANMMR*), which is defined in the MPEG-7 standard [44].

As the main goal of the research is to improve audio retrieval performance by the means of feature synthesis, an intuitive approach for constructing a proper fitness function $\mathcal{F}[\cdot]$ for the task would involve computing either the average retrieval precision (*AP*) or the *ANMMR* values. However, neither option would be computationally feasible for large databases, as they both require conducting a separate *batch query* (i.e., selecting each item in the database as a query item one by one, performing a separate query for each of them, and finally taking the mean of the obtained retrieval results) for every fitness evaluation during the synthesis process. Therefore, in this article, we concentrate on obtaining a *maximal discrimination* between the features of different database classes, which should in turn result in improved retrieval performance. To achieve this we propose two alternative ways to form the fitness function, described in the following sections.

Discrimination measure

First, a measure for evaluating the discrimination capability provided by the synthesized features is proposed. The measure is based on two widely used criteria in clustering:

- *Compactness*: the database items of one cluster should be similar and close to each other in the feature space, and
- *Separation*: different clusters and their centroids should be distinct and well separated from each other.

Suppose the different labels of an L-class database are denoted as $l_0, \ldots, l_{L-1}$, and the corresponding class centroids as $\mu_0, \ldots, \mu_{L-1}$. Then, the following *discrimination measure* (*DM*) can be defined over a set of synthesized FVs, $S = \{s\}$,

$$\begin{aligned} &\mathrm{DM}[S] = \mathrm{FP}(S) + \delta_{\mathrm{mean}}(l_n)/\delta_{\mathrm{min}}(l_n, l_m), \\ &\text{where} \\ &\delta_{\mathrm{mean}}(l_n) = \frac{1}{L}\sum_{n=0}^{L-1}\frac{\sum_{\forall s \in l_n}\|\mu_n - s\|}{|l_n|}, \\ &\delta_{\mathrm{min}}(l_n, l_m) = \min_{n \neq m}(\|\mu_n - \mu_m\|). \end{aligned} \tag{14}$$

The terms of (14) are defined as follows: FP(S) stands for the number of *false positive* FVs occurring among the synthesized FVs S, meaning that those FVs are actually located in closer proximity to some other class centroid than their own, $\delta_{\mathrm{mean}}(l_n)$ is the average intra-class distance, and $\delta_{\mathrm{min}}(l_n, l_m)$ corresponds to the minimum centroid distance among all the classes. Thus, the discrimination measure, DM[S], strives for minimizing the intra-cluster distance, while maximizing the shortest inter-cluster distance. Ideally, each synthesized feature is in the closest proximity of its own class centroid, thus leading to a high discrimination among classes as illustrated in Figure 2. However, minimizing the DM does not always lead to improved retrieval results. This is due to the fact that query items located at the outskirt of their own classes may be actually situated in closer proximity to some other FV located on the outskirt of its corresponding (wrong) class. Thus, in order to improve not only the feature discrimination in the feature space, but also the audio retrieval performance, next we propose applying a similar methodology that is utilized in feed-forward ANNs.

Target vector assignment

In the second approach, the idea is to assign a binary synthesizer *target vector* for each class, and let the underlying optimization algorithm to search for the proper synthesis parameters producing the desired output. The actual fitness value is then obtained by comparing the obtained and desired output vectors in a *mean square error* (MSE) sense. However, as the output dimension of the EFS is not fixed in advance, a separate target vector is generated for each dimension $d \in [D_{\mathrm{min}}, D_{\mathrm{max}}]$, resulting to a complete target *matrix*. For producing the matrix, a so-called *error correcting output code* [45] analogy is applied, which suggests two criteria for generating proper binary target matrices:

- *Row separation*: The target vectors should be well separated from each other in the terms of Hamming distance.

- *Column separation*: Also the columns of the target matrix should be well separated from each other in the terms of Hamming distance.

Large row separation allows the synthesized vectors to differ somewhat from the actual target vectors without losing the discrimination between different classes. The reasoning for large column separation follows from the fact that each column in the target matrix can be seen as an individual *binary* classification task (between the classes having a value 1 and the classes having a value −1 in a specific column). Because of the varying similarity between two arbitrary audio classes, some of such binary classifications are likely to become much easier than others. Hence, as the same target vectors are nonetheless applied to any given input classes, it is also beneficial to keep the binary classification tasks as different as possible by maximizing the column separation.

To generate a binary matrix with maximal row and column separation, the following matrix generation procedure is used:

1. Compute the minimum number of bits, $b_{\min}$, needed to represent the total number of classes, L, in the database.
2. Form an empty matrix with L rows.
3. For each row, assign a binary representation of the row number $n \in [0, L-1]$ as the first/next $b_{\min}$ target vector values.
4. Move the first row of the matrix to the bottom and shift the other rows up by one.
5. Repeat the steps 3 and 4 until $D_{\max}$ target vector values have been assigned.
6. Replace the first L values of each target vector with a 1-of-L coded [46] section.

The procedure generates new columns until the matrix is rotated back to its original order, resulting into a high column separation. Simultaneously, the row separation is greatly increased compared to the regular 1-of-L coding section. However, in practice it was observed that for distinct classes it is often easiest to find a synthesizer that discriminates a certain single class from the others, and, therefore, sparing the 1-of-L coding section at the beginning of the matrix generally improves the synthesis results. For the vector dimensions $d < D_{\max}$, only the first d elements of the target vectors are considered. This yields identical target vector elements between the vectors of different length, allowing the FGBF algorithm to combine particle position elements from different dimensions (refer to FGBF algorithm).

For clarification of the procedure, a target matrix for a 4-class database is demonstrated in Figure 6, where the maximum dimension is set to $D_{\max} = 10$, and the minimum number of bits needed to represent the $L = 4$ classes is $b_{\min} = 2$. The empty entries of the matrix correspond to values −1, and $T[l_n]$ stands for a target vector for class l_n. As illustrated in this example, the generated matrix follows the provided algorithm with its 1-of-L coding section for the first L elements, followed by the elements created by the shifted rows of 2-bit row number representations.

By applying the definitions given above, the fitness value for the jth elements of all the synthesized FVs, S_j, can be computed as

$$\mathcal{F}_j[S_j] = \sum_{n=0}^{L-1} \sum_{\forall s \in l_n} \left(T[l_n]_j - s_j \right)^2, \tag{15}$$

where $T[l_n]_j$ denotes the jth element of the target vector of class l_n, and s_j is the jth element of a single synthesized output vector belonging to class l_n. The overall fitness score $\mathcal{F}[S]$ can then be formed by adding the *fractional* fitness scores $\mathcal{F}_j[S_j]$ together and applying normalization with respect to the number of dimensions. However, due to an observation that the first L vector elements, having the 1-of-L encoding, are usually the easiest ones to synthesize (and thus mainly favored in the dimension search by the MD PSO algorithm), the first L vector elements are handled separately in the summation process. Thus, for the dimensions $d > L$, the normalizing divisor is *strengthened* by an additional power parameter $\alpha > 1$, which is to moderately increase the probability of converging into higher output vector dimensions whenever found beneficial. As a result, the overall fitness function $\mathcal{F}[S]$ can be formulated as

$$\mathcal{F}[S] = \frac{1}{L} \sum_{j=1}^{L} \sum_{n=0}^{L-1} \sum_{\forall s \in l_n} \left(T[l_n]_j - s_j \right)^2 + \frac{1}{(d-L)^{\alpha}} \sum_{j=L+1}^{d} \sum_{n=0}^{L-1} \sum_{\forall s \in l_n} \left(T[l_n]_j - s_j \right)^2, \tag{16}$$

in which it is assumed that $D_{\min} > L$.

$T[l_0]$	1				1		1	1		
$T[l_1]$		1			1	1				1
$T[l_2]$			1					1	1	
$T[l_3]$				1		1	1		1	1

Figure 6 An illustration of a target matrix assigned for $L = 4$ classes.

Experimental results

The EFS technique proposed in this article was tested with several audio classification and retrieval experiments using two separate audio databases. The first database consists of six distinct classes, while the second one was created by adding another six classes to the first database to form a more challenging 12-class database. The contents of the two databases are shown in Table 3, where the class numbers ranging from 1 to 6 belong to the *basic* database of 6 classes, and the *extended* database with 12 classes includes also the class numbers ranging from 7 to 12. The audio class samples are collected from a few different data sources; the speech classes (class nos. 1 and 7) are derived from the TIMIT[a] database, the music classes (class nos. 5, 6, and 12) are from the RWC Music Database[b] and another music collection at Tampere University of Technology (TUT), the "general" audio sounds (class nos. 4, 9, 10, and 11) were purchased from the *StockMusic.com* webpage,[c] and, finally, the singing and whistling samples (class nos. 2, 3, and 8) are self-recorded and produced at TUT. The reasoning for two separate databases is to evaluate the *scalability* of the proposed feature synthesis system to more complex and difficult classification and retrieval tasks (i.e., to those cases where good features are truly needed).

In the experiments shown in this section, unless stated otherwise, the following parameters and settings were used for the EFS: the depth of the synthesis was set to $K = 7$, meaning that 7 operators and $K + 1 = 8$ features were chosen for the synthesis process of each output vector element, and the total number of operators, listed in Table 4 for features f_a and f_b, was set to $\Theta = 18$. Similarly, the parameters for the MD PSO algorithm were set as follows: the swarm size was set to $P = 600$ particles, 1,500 iterations were used, the dimension ranges for the basic and extended databases were set as $[D_{\min}, D_{\max}] = [7, 45]$ and $[D_{\min}, D_{\max}] = [13, 45]$, respectively, and the dimensional velocity range was experimentally determined as $[VD_{\min}, VD_{\max}] = [-6, 6]$. Finally, the parameter $\alpha > 1$ used in (16) was determined separately for the key-frame and segment features as $\alpha = 1.15$ and $\alpha = 1.3$, respectively, as these values were found out to yield the best trade-off between the computational cost caused by higher dimensionality and the possible lack of performance caused by reduction of the synthesized FV dimension.

In this study, separate "training sets" were considered for both databases. The training sets were formed by random selection, such that 45% of the audio clips of each class from both databases were included to them. Analogous to supervised machine learning, extracted features of these training sets were then used in searching the most suitable synthesis parameters for the corresponding databases, after which the found parameters were applied in synthesizing the features of the whole data. Thus, after the parameter search process, the resulting EFS system may synthesize new features for the rest of the (unseen) data without further optimization processes. The final obtained synthesized features were tested with multiple evaluations, demonstrating their enhanced discrimination capabilities over the original FSs.

Performance evaluations on feature discrimination

We will first concentrate on evaluating the feature DM of the original and synthesized FSs. The DM was given in (14), and the obtained results for the original and synthesized segment- and key-frame-level audio features of low-level audio features are shown in Table 5. Recall that the higher the DM value, the more the features are mixed up with each other in the feature space (indicating less discrimination). As

Table 3 The contents of the two audio databases

		Class number	Audio class name	Number of samples
12-class audio database (extended)	6-class audio database (basic)	1	Female speech	111
		2	Male singing	63
		3	Whistling	94
		4	Breaking glass	83
		5	Classical music	116
		6	Electrical/techno music	107
	The 6 added classes	7	Male speech	103
		8	Female singing	119
		9	Bird singing	91
		10	Dog barking	49
		11	Fire sounds	96
		12	Rock/metal music	72
			Total	1104

Table 4 The list of operators used in the feature synthesis process for features f_a and f_b

θ_n	Formula	θ_n	Formula
0	$-f_a$	9	$f_a * f_b$
1	$-f_b$	10	$10\ (f_a * f_b)$
2	$\max\ (f_a, f_b)$	11	f_a/f_b
3	$\min\ (f_a, f_b)$	12	$\sin\ (100\pi(f_a + f_b))$
4	$f_a * f_a$	13	$\cos(100\pi(f_a + f_b))$
5	$f_b * f_b$	14	$\tan(100\pi(f_a * f_b))$
6	$f_a + f_b$	15	$\tan(100\pi(f_a + f_b))$
7	$10\ (f_a + f_b)$	16	$0.5 * \exp(-(f_a - f_b) * (f_a - f_b))$
8	$f_a - f_b$	17	$0.5 * \exp(+(f_a + f_b) * (f_a + f_b))$

expected, it can be seen that discriminating the features of the extended 12-class database is significantly more challenging than those with the basic database. However, after performing the feature synthesis procedure with the found synthesis parameters, substantial DM improvements were obtained for all the FSs. Note that, due to the stochastic nature of the underlying optimization approach, *ten* separate EFS evaluations were performed for all the FSs, and the obtained mean (μ) and standard deviation (σ) values are also reported. The rest of the synthesis results presented in this section conform to the same protocol.

As the DM as such is not an intuitive measure of fitness for the FSs, we will concentrate on demonstrating the audio retrieval performances obtained with different FSs. Recall, however, that in general the usage of new synthesized features is not by any means restricted to merely retrieval purposes. Instead, as long as a proper fitness function can be designed, the EFS technique can be applied to basically any task requiring feature improvement. To show an example, a rather obvious application is demonstrated, in which a *K-means* classifier is applied. The algorithm was first run with the formed training sets to compute the class centroids, after which the unseen (55%) *test* data samples were classified according to their closest class centroid. The *classification error* (CE) results obtained with the synthesized features for the both databases were compared to the errors associated with the original FSs. The results, shown in Table 6, indicate clearly improved classification performance obtained by the new synthesized features.

Performance evaluations on audio retrievals

For evaluating the audio retrieval performance, the metrics, ANMRR and AP, introduced in "The fitness measures" section were applied. In our approach, we first compute (using Euclidean distance) the distances between the first FV of a query item and all the FVs of a particular database item from which we take the *minimum* distance (normalized by the vector length) and store it. Second, we take the next FV of the query item and continue similarly with all the query item FVs. Finally, before moving to the next database item, we take the *sum* of all the obtained minimum distances to obtain a single distance value between the queried item and the database item (which is used to rank that query item). In order to provide some baseline results, the ANMRR and AP values obtained by using a batch query and the originally extracted audio features are shown in Table 7. As expected, for both the segment- and the key-frame-based FSs, the retrieval performance deteriorates considerably as the database size increases from the basic 6-class database to the extended 12-class database. Like in the results shown in Tables 5 and 6, the segment MFCC features seem to provide the best performance among the original FSs.

Table 5 The DM statistics of ten separate EFS evaluations for the original and synthesized features

FS	Basic database (6 classes)		Extended database (12 classes)	
	Original DM	Synthesized DM (μ ± σ)	Original DM	Synthesized DM (μ ± σ)
Segment				
STAT	500.5	43.6 ± 6.7	1793	406.3 ± 47.6
MFCC	333.4	61.5 ± 8.5	1141	361.0 ± 22.4
Δ-MFCC	515.7	130.5 ± 7.9	1865	620.1 ± 27.1
ΔΔ-MFCC	622.6	140.9 ± 11.5	2072	532.7 ± 23.6
LPC	1114	270.3 ± 8.9	4520	1204 ± 29.1
LPCC	1638	310.3 ± 12.8	10830	1395 ± 26.5
S_AUDIO	342.6	56.0 ± 4.1	1365	382.8 ± 15.2
Key-frame				
MFCC + deltas	2627	820.6 ± 53.6	7113	3093 ± 138
LPC + LPCC	6951	2143 ± 39.4	23 900	6378 ± 157
K_AUDIO	3150	782.7 ± 50.5	10 140	2774 ± 154

Table 6 CE of a K-means classifier over the original and synthesized features

FS	Basic database (6 classes)		Extended database (12 classes)	
	Original CE (%)	Synthesized CE (%)	Original CE (%)	Synthesized CE (%)
Segment				
STAT	33.3	15.2 ± 1.6	54.8	37.4 ± 2.4
MFCC	18.2	13.5 ± 1.7	36.0	34.4 ± 2.9
Δ-MFCC	30.7	24.2 ± 1.4	48.3	39.9 ± 1.5
ΔΔ-MFCC	37.8	29.1 ± 2.1	56.7	39.4 ± 2.3
LPC	47.5	41.8 ± 1.3	70.8	68.4 ± 2.0
LPCC	57.6	45.6 ± 1.4	78.9	74.5 ± 1.6
S_AUDIO	22.1	18.0 ± 0.9	42.8	34.3 ± 2.0
Key-frame				
MFCC + deltas	37.2	29.2 ± 2.2	50.3	50.0 ± 2.0
LPC + LPCC	75.6	64.4 ± 1.2	86.8	84.5 ± 1.2
K_AUDIO	53.8	36.0 ± 4.6	69.1	46.2 ± 2.7

Next, a standard SLP classifier was trained with the MD PSO approach (by the methods described in [30]), in order to compare the proposed EFS technique with neural network-based approaches. Similar to EFS, an SLP can be treated as a feature synthesizer, in which case the original audio features are propagated through the network and the output layer dimensionality is set equal to the total number of classes, L. We call the resulting output vector a *class vector*, as it indicates the class designated for a particular input FV. The corresponding ANMRR and AP values obtained with these vectors are shown in Table 8. Compared to the values obtained with the original features, a clear improvement can be observed with nearly all the FSs, excluding some of the segment features of the extended database. As mentioned in "Low-level audio features" section, this is most probably due to the fact that, because of the nature of the segment-based features, the total number of FVs per an audio clip is considerably less for segment features than for key-frame features. Hence, because the FV dimensionality is highly decreased during the SLP synthesis process (to L), the extended database is extremely complicated for an SLP classifier to learn with such a limited amount of data. In contrast, the basic database is well learned by the SLP, as an AP of nearly 90% is achieved with the segment-based MFCC features.

We will now demonstrate the significance of the additional properties associated with the EFS technique

Table 7 The retrieval performances obtained with the original features for both audio databases

FS	Basic database (6 classes)		Extended database (12 classes)	
	ANMRR	AP (%)	ANMRR	AP (%)
Segment				
STAT	0.358	61.0	0.514	45.9
MFCC	0.228	73.8	0.367	60.2
Δ-MFCC	0.351	62.2	0.507	46.6
ΔΔ-MFCC	0.379	59.5	0.531	44.4
LPC	0.549	43.1	0.686	29.8
LPCC	0.569	41.1	0.717	26.9
S_AUDIO	0.263	71.0	0.450	52.3
Key-frame				
MFCC + deltas	0.460	51.4	0.568	41.0
LPC + LPCC	0.637	34.8	0.787	20.4
K_AUDIO	0.375	59.5	0.524	44.9

Table 8 The retrieval performances obtained with the class vectors of an MD PSO-trained SLP classifier for both audio databases

FS	Basic database (6 classes)		Extended database (12 classes)	
	ANMRR	AP (%)	ANMRR	AP (%)
Segment				
STAT	0.238	75.0	0.626	36.4
MFCC	0.101	89.2	0.518	47.3
Δ-MFCC	0.339	64.7	0.581	40.0
ΔΔ-MFCC	0.290	70.1	0.585	40.1
LPC	0.491	49.7	0.673	31.3
LPCC	0.564	42.3	0.714	27.4
S_AUDIO	0.229	75.7	0.578	41.2
Key-frame				
MFCC + deltas	0.279	70.2	0.414	56.5
LPC + LPCC	0.669	32.1	0.794	19.7
K_AUDIO	0.308	67.7	0.504	47.1

germane to SLPs. In the first demonstration, the feature selection property is enabled and applied to the input FVs, so that only $K+1=8$ original features are included into the actual synthesis process. The output FV dimensionality is fixed to L, and only the '+' and '−' operators are included in the synthesis process in order to mimic the behavior of typical neural networks (the subtraction operation is also needed to compensate with SLP weights, which are limited to [−1,1]). Such arrangements make the EFS similar to an SLP classifier (with the exception of additional feature selectivity). The retrieval performance obtained with these settings is shown in Table 9, from where it can be seen that, excluding the MFCC features, the results are fairly comparable to those obtained with the SLP classifier. This is a result worth noting as only *eight* features were selected among the original input FVs. Such a reduction in the number of original features (without significant performance loss) suggests that at least some of them may not be essential to attain optimal discrimination capability.

In the next phase, the fixed output dimensionality of the synthesis method was changed so that the optimal dimensionality found using the underlying MD PSO algorithm was used in the synthesis process. All the operators shown in Table 4 were included to the process to provide the synthesizer more possibilities for modifying the features. After the changes, a significant improvement in the retrieval performance was obtained, as shown in Table 10. Hence, optimizing the output FV dimension (and applying several operators in the synthesis process) significantly enhances the retrieval performance. However, it should be noted that retrieval performance obtained with the original segment-based MFCC features of the extended database could not be improved either by the SLP classifiers or by the EFS experiments made so far. We believe this indicates that the MFCC features need to be treated as *a whole* and that selecting only few of them may decrease the overall retrieval performance. However, in this instance the performance drop is limited and clear performance improvements relative to key-frame-based MFCC features are still achieved. Also key-frame LPC + LPCC features show only minor improvements in AP values, which implies that certain audio features are not as *synthesizable* as others or that different types of arithmetic or logic operators may be required to achieve significant improvements in discrimination performance.

Finally, several *consecutive* EFS runs were performed to see whether the obtained synthesis results can be further improved (see Figure 4 in "Overview of the proposed feature synthesis system" section). Retrieval performance associated with certain synthesized FSs improved over several runs (<25), whereas some other features could not be enhanced notably. More generally, the segment-based features were more suitable for consecutive EFS runs, as retrieval performance of the key-frame features remained rather stable during the runs. A graphical presentation of the experiments is shown in Figure 7, where the evolution of both the AP results and the synthesized FV dimensions over 25 synthesis runs are shown for several FSs and for both databases ($L=6$/$L=12$). For simplicity, those FSs unable to demonstrate

Table 9 The retrieval performance statistics obtained using the EFS technique with fixed output dimension and only two operators

FS	Basic database (6 classes)		Extended database (12 classes)	
	ANMRR ($\mu \pm \sigma$)	AP (%) ($\mu \pm \sigma$)	ANMRR ($\mu \pm \sigma$)	AP (%) ($\mu \pm \sigma$)
Segment				
STAT	0.278 ± 0.016	70.4 ± 1.7	0.534 ± 0.024	44.1 ± 2.4
MFCC	0.280 ± 0.022	69.6 ± 2.1	0.528 ± 0.023	44.9 ± 2.2
Δ-MFCC	0.375 ± 0.016	60.6 ± 1.6	0.574 ± 0.011	40.6 ± 1.0
ΔΔ-MFCC	0.393 ± 0.013	59.0 ± 1.3	0.598 ± 0.020	38.3 ± 2.0
LPC	0.549 ± 0.006	43.8 ± 0.6	0.729 ± 0.013	25.7 ± 1.2
LPCC	0.589 ± 0.015	39.6 ± 1.5	0.740 ± 0.006	24.8 ± 0.5
S_AUDIO	0.236 ± 0.015	74.7 ± 1.6	0.480 ± 0.013	49.9 ± 1.2
Key-frame				
MFCC + deltas	0.463 ± 0.024	51.4 ± 2.3	0.629 ± 0.030	35.2 ± 2.8
LPC + LPCC	0.709 ± 0.007	28.4 ± 0.7	0.813 ± 0.003	18.0 ± 0.3
K_AUDIO	0.346 ± 0.016	63.0 ± 1.5	0.532 ± 0.012	44.4 ± 1.2

Table 10 The retrieval performance statistics obtained using the EFS technique with dynamic output dimension and 18 operators

FS	Basic database (6 classes)		Extended database (12 classes)	
	ANMRR ($\mu \pm \sigma$)	AP (%) ($\mu \pm \sigma$)	ANMRR ($\mu \pm \sigma$)	AP (%) ($\mu \pm \sigma$)
Segment				
STAT	0.167 ± 0.013	81.7 ± 1.5	0.425 ± 0.012	55.0 ± 1.2
MFCC	0.221 ± 0.021	75.5 ± 2.2	0.398 ± 0.012	57.5 ± 1.2
Δ-MFCC	0.332 ± 0.015	64.7 ± 1.5	0.511 ± 0.096	43.9 ± 3.1
ΔΔ-MFCC	0.313 ± 0.016	66.7 ± 1.6	0.526 ± 0.033	45.0 ± 3.2
LPC	0.524 ± 0.016	46.1 ± 1.5	0.675 ± 0.008	31.1 ± 0.8
LPCC	0.556 ± 0.010	42.8 ± 1.0	0.720 ± 0.017	26.7 ± 1.6
S_AUDIO	0.171 ± 0.011	81.5 ± 1.2	0.442 ± 0.029	53.4 ± 2.9
Key-frame				
MFCC + deltas	0.371 ± 0.012	60.3 ± 1.2	0.470 ± 0.012	50.5 ± 1.1
LPC + LPCC	0.634 ± 0.013	35.3 ± 1.2	0.775 ± 0.011	21.6 ± 1.0
K_AUDIO	0.289 ± 0.029	68.7 ± 2.8	0.441 ± 0.009	53.2 ± 0.9

much improvement during the process are not shown in the graphs, whereas the best obtained AP values of all the FSs are shown in Table 11. The most significant observation concerns the behavior of the extended database segment features (upper right plots in Figure 7), where major improvements can be seen with all the shown FSs. Moreover, the dimensionality of the "S_AUDIO" FV could be reduced to only 13. Also note that now the AP performance of the synthesized segment-based MFCC features surpasses the original features after the second EFS run. However, in the case of key-frame features (the lower plots), the improvement is more moderate, stating that these features have not as much additional potential to be found by repeating the synthesis process, or at least such potential could not be found in the experiments using the applied MD PSO and EFS parameters.

Comparative evaluations and discussion

In order to provide some additional comparative results, an MLP classifier was trained using the MD PSO algorithm as described in [30]. In this approach, a specific *architecture space* (AS) with lower and upper limits for the number of neurons for each network layer is specified, from which the applied optimization algorithm searches for the optimal network structure for synthesizing new features. Two different ASs were experimented, specified by the limiting vectors

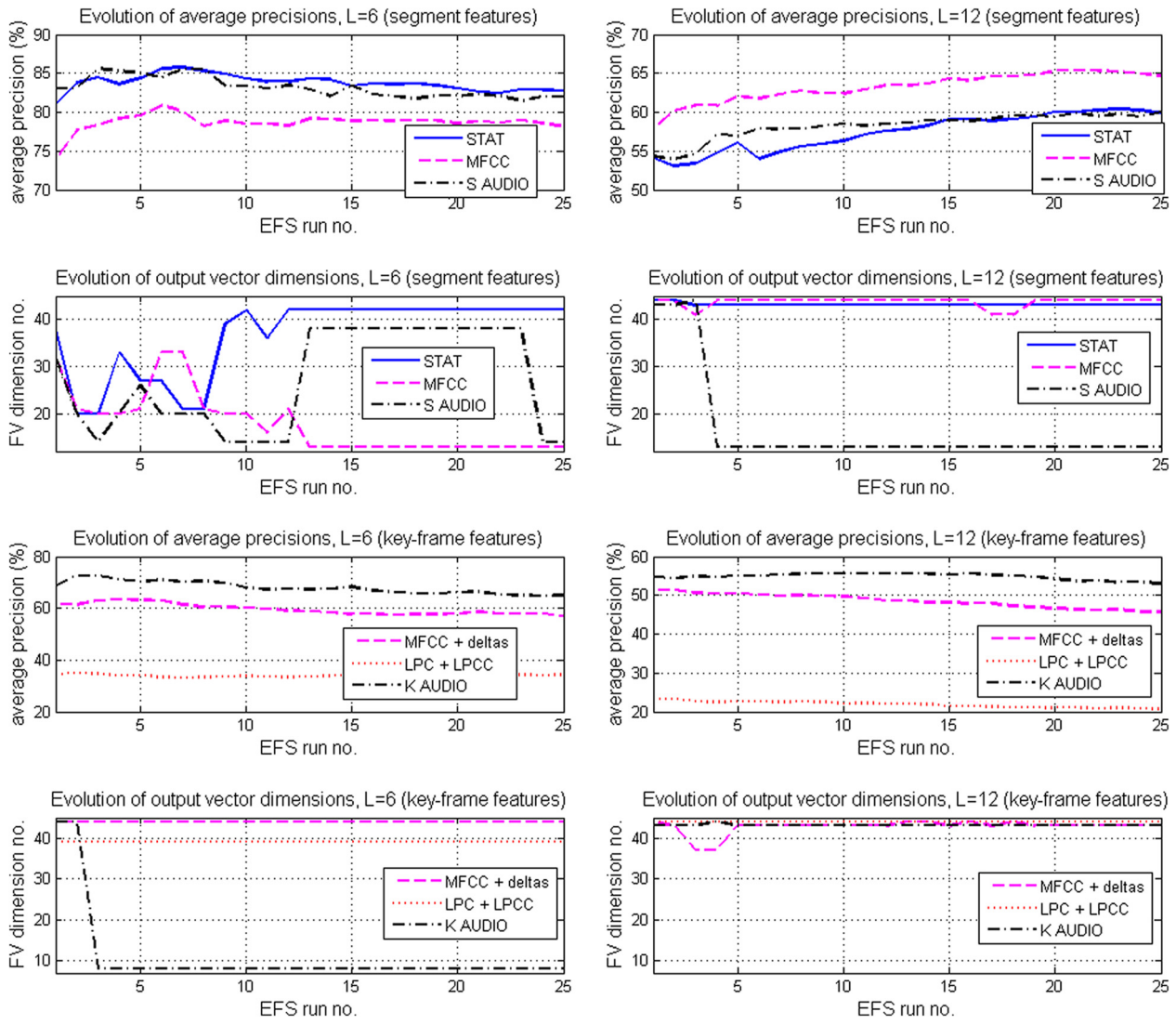

Figure 7 The consecutive EFS runs performed for basic (left) and extended (right) databases. It can be seen that the AP results generally improve during the couple of first runs. The segment features seem to improve over several runs, whereas the performance level obtained by key-frame features saturates more quickly. Note that satisfactory solutions can also be found with low FV dimensions.

Table 11 The best retrieval performances obtained by 25 consecutive EFS runs

FS	Basic database (6 classes)		Extended database (12 classes)	
	ANMRR (run no.)	AP (%) (run no.)	ANMRR (run no.)	AP (%) (run no.)
Segment				
STAT	0.134 (run 7)	85.8 (run 7)	0.380 (run 23)	60.4 (run 23)
MFCC	0.181 (run 6)	80.9 (run 6)	0.333 (run 22)	65.3 (run 22)
Δ-MFCC	0.235 (run 16)	76.0 (run 16)	0.514 (run 7)	47.0 (run 7)
ΔΔ-MFCC	0.258 (run 10)	73.2 (run 10)	0.467 (run 10)	51.4 (run 10)
LPC	0.446 (run 22)	54.3 (run 22)	0.673 (run 2)	31.4 (run 2)
LPCC	0.521 (run 24)	47.3 (run 24)	0.710 (run 1)	27.7 (run 1)
S_AUDIO	0.135 (run 3)	85.6 (run 3)	0.386 (run 25)	60.0 (run 25)
Key-frame				
MFCC + deltas	0.346 (run 4)	63.3 (run 4)	0.462 (run 2)	51.4 (run 2)
LPC + LPCC	0.638 (run 2)	35.0 (run 2)	0.757 (run 2)	23.3 (run 2)
K_AUDIO	0.251 (run 2)	72.5 (run 2)	0.426 (run 10)	55.6 (run 16)

Table 12 The retrieval performances obtained with the class vectors of an MLP classifier

FS	Basic database (6 classes)		Extended database (12 classes)	
	ANMRR	AP (%)	ANMRR	AP (%)
Segment				
STAT	0.334 (1)	66.0 (1)	0.620 (2)	37.1 (2)
MFCC	0.089 (1)	90.4 (1)	0.532 (1)	45.7 (1)
Δ-MFCC	0.340 (2)	65.3 (2)	0.680 (1)	30.9 (1)
ΔΔ-MFCC	0.320 (1)	66.8 (1)	0.634 (2)	35.3 (2)
LPC	0.489 (1)	50.2 (1)	0.704 (2)	28.7 (2)
LPCC	0.562 (1)	42.6 (1)	0.746 (1)	24.4 (1)
S_AUDIO	0.255 (1)	73.6 (1)	0.561 (2)	43.2 (2)
Key-frame				
MFCC + deltas	0.334 (1)	64.8 (1)	0.555 (1)	42.7 (1)
LPC + LPCC	0.669 (1)	32.1 (1)	0.805 (1)	18.8 (1)
K_AUDIO	0.391 (2)	59.4 (2)	0.604 (1)	38.0 (1)

$AS1_{\min} = \{F, 13, L\}, AS1_{\max}\{F, 45, L\}$ and $AS2_{\min} = \{F, 8, 4, L\}, AS2_{\max} = \{F, 16, 8, L\}$, where the fixed input and output layers F and L correspond to the number of input features and database classes, respectively. These settings correspond to performing two (AS1) and three (AS2) consecutive EFS runs. For this reason, the number of MD PSO iterations were set to $2*1500 = 3000$ and $3*1500 = 4500$, respectively, in order to have a fair comparison. Both ASs were separately searched with the MD PSO algorithm and the best network configuration was used to generate the corresponding MLP class vectors from the original features. The obtained retrieval performance measures are shown in Table 12, where also the AS number (1 or 2) yielding the better result is shown in parenthesis. The MLP classifier performs notably well with the segment-based MFCC features for the basic database, whereas its performance falls below the EFS results with the extended database. This result suggests that regular neural networks can succeed as synthesizers with rather small and simple databases. Also, as no feature selection is performed, the MLP can utilize the original MFCCs as a whole, which may ease the synthesis process. On the other hand, when it comes to the key-frame-based MFCC feature performance, the difference is not as significant. This suggests that the content and nature of the original FV indeed has an effect on the synthesis process output both with ANNs and EFS.

Strictly speaking, performing several EFS runs in a row is analogous to concatenating several SLP classifiers such that the output class vector of a previous SLP is considered as an input vector for the next one. To see whether such an arrangement makes a difference in retrieval performance, Figure 8 shows the evolution of AP values of ten concatenated SLP classifiers trained with the MD PSO. The obtained results are shown also numerically for all FSs in Table 13. Compared to single SLP evaluations, no significant improvements were achieved by concatenating the SLP classifiers. Rather, retrieval performance begins to decrease either immediately or after two or three SLP classifiers. This leads to a conclusion that, at least with the databases and parameters used in our study, SLP (as a feature synthesizer) achieves its best performance almost right away, whereas the proposed EFS framework can improve its performance over several additional synthesis runs. Moreover, in most cases the EFS results are clearly better than those obtained with MLPs, which supports the EFS approach of performing the fitness evaluation after every run.

Finally, the evaluation shown in Figure 7 was repeated for segment-based features so that every EFS run was repeated *three times,* with only the best synthesis solution (i.e., the one maximizing the retrieval performance in the training dataset) being used. This was done in order to demonstrate the full potential of the EFS technique; in this way the occasional sub-optimal solutions (caused by the stochastic nature of the search process) can be reduced efficiently. Note that unless grid computing with parallel processes can be used, the processing time would be three times longer. Hence, such a demonstration is more about providing an idea of EFS's potential than a practical application. Nevertheless, Figure 9 shows additional improvements in AP scores, especially in the case of the extended 12-class database where performance increases notably. With the basic database, the optimal performance level (in the sense of AP values) is reached after 5–7 runs. Note that improved retrieval

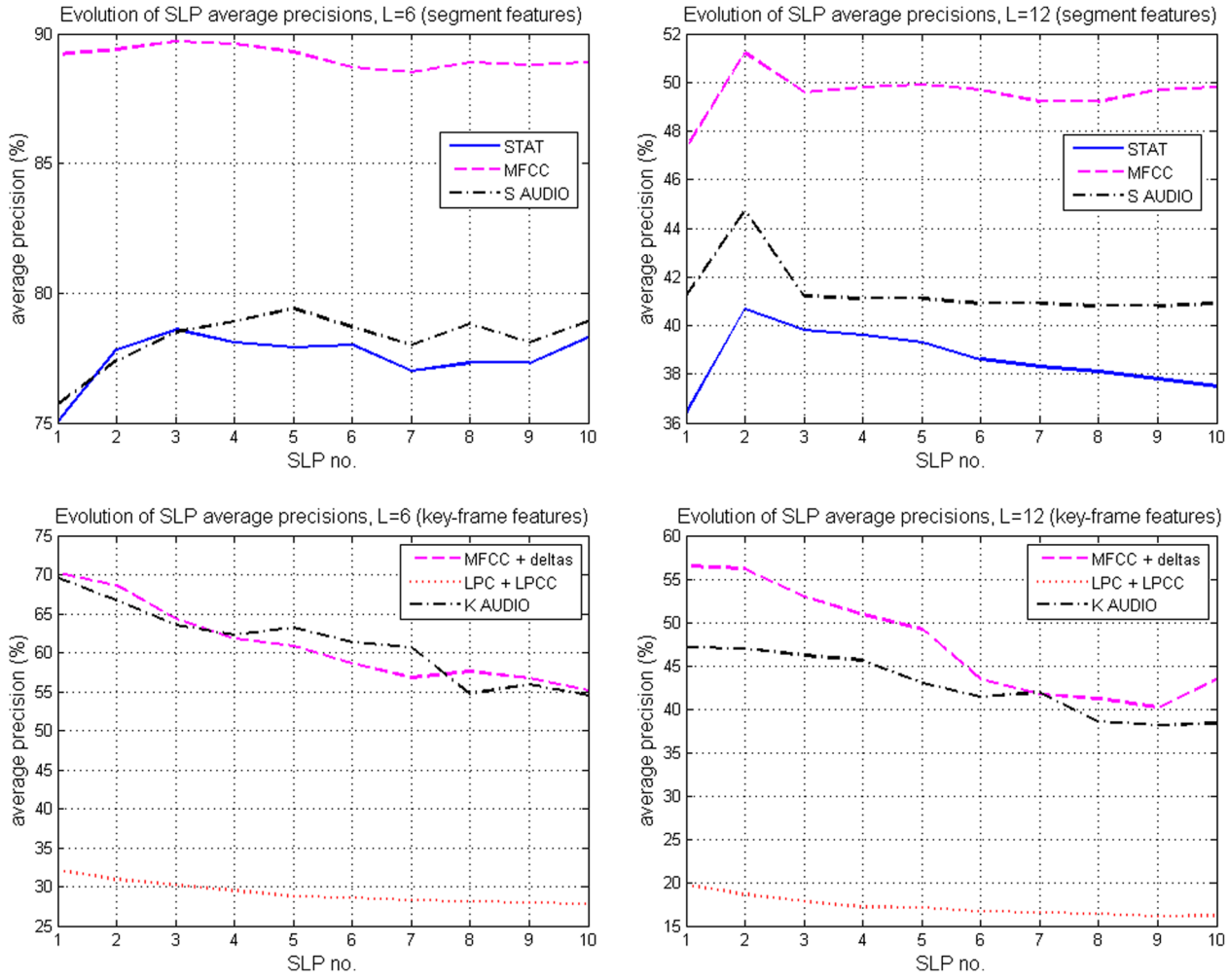

Figure 8 The AP performances obtained by consecutive SLP classifiers. In general, the results cannot be further improved by concatenating the SLPs, except with the segment features for the first few classifiers.

performance was also obtained using lower FV dimensions than the original ones, which is an attractive property considering the computational requirements for processing the vectors. For example, the dimensionality of the synthesized vectors can significantly be decreased for STAT and S_AUDIO features. However, the best FV dimensions found may vary depending on the FS. Specifically, it was observed that synthesizing the 26-dimension MFCC features of the extended database into a lower dimensionality (and yet obtain improved retrieval performance) is a challenging task. Nonetheless, allowing a dynamic output FV dimensionality is an advantage, as specific applications (or classifiers) requiring synthesized features may have strict computational (or other) constraints on the FS dimensions. Such requirements can be met with the EFS technique by setting the output dimension range equal to the requested one, or by setting it to a single specific value. Furthermore, by allowing the output vector dimension vary during the synthesis process, the most suitable FV dimension can be found *concurrently* in a single experiment, obviating the need to perform separate experiments with different (fixed) FV dimensions.

Computational complexity

Considering the computational complexity of the SLP, MLP, and EFS approaches, the most important factor is the number of particles and iterations applied to the underlying MD PSO algorithm. Their selection is influenced by the number and identity of the FV dimensions synthesized (i.e., database size, feature extraction approaches, and the original FSs). In the case of EFS, the synthesis depth, K, is another important factor, as in every particle element the synthesis process consists of $K - 1$ input features. There is usually a trade-off between computation time and performance attained, so that the parameters affecting processing time should be set depending on the specific task.

Table 13 The best retrieval performances obtained by ten consecutive SLP classifiers

FS	Basic database (6 classes)		Extended database (12 classes)	
	ANMRR (run no.)	AP (%) (run no.)	ANMRR (run no.)	AP (%) (run no.)
Segment				
STAT	0.206 (run 3)	78.6 (run 3)	0.579 (run 2)	40.7 (run 2)
MFCC	0.109 (run 3)	89.7 (run 3)	0.479 (run 2)	51.2 (run 2)
Δ-MFCC	0.295 (run 2)	69.8 (run 2)	0.534 (run 3)	45.2 (run 3)
ΔΔ-MFCC	0.269 (run 10)	72.4 (run 10)	0.579 (run 4)	41.3 (run 4)
LPC	0.483 (run 2)	50.9 (run 2)	0.667 (run 4)	32.3 (run 4)
LPCC	0.550 (run 2)	44.2 (run 2)	0.714 (run 1)	27.4 (run 1)
S_AUDIO	0.202 (run 5)	79.4 (run 5)	0.541 (run 2)	44.7 (run 2)
Key-frame				
MFCC + deltas	0.279 (run 1)	70.2 (run 1)	0.414 (run 1)	56.5 (run 1)
LPC + LPCC	0.669 (run 1)	32.1 (run 1)	0.794 (run 1)	19.7 (run 1)
K_AUDIO	0.277 (run 1)	69.6 (run 1)	0.504 (run 1)	47.1 (run 1)

The structure of the proposed feature synthesis technique is particularly designed for *parallel* computing, so that grid computing[d] was utilized in computing the results. Here, each FS was processed on its own, as shown in Figure 4. With the applied MD PSO and EFS parameters, the computational time of a single EFS run varied between 30 min (segment features of the basic database) and 3.5 h (key-frame features of the extended database). The corresponding computation times for the SLP and MLP classifiers trained with MD PSO were more or less the same. However, it should be kept in mind that once the synthesis (or network) parameters are found, features of any previously unseen data can be synthesized afterwards with no need for any further search or training processes.

Conclusions

A method for transforming and modifying traditional audio features by an evolutionary optimization algorithm is proposed, one that improves feature discrimination as

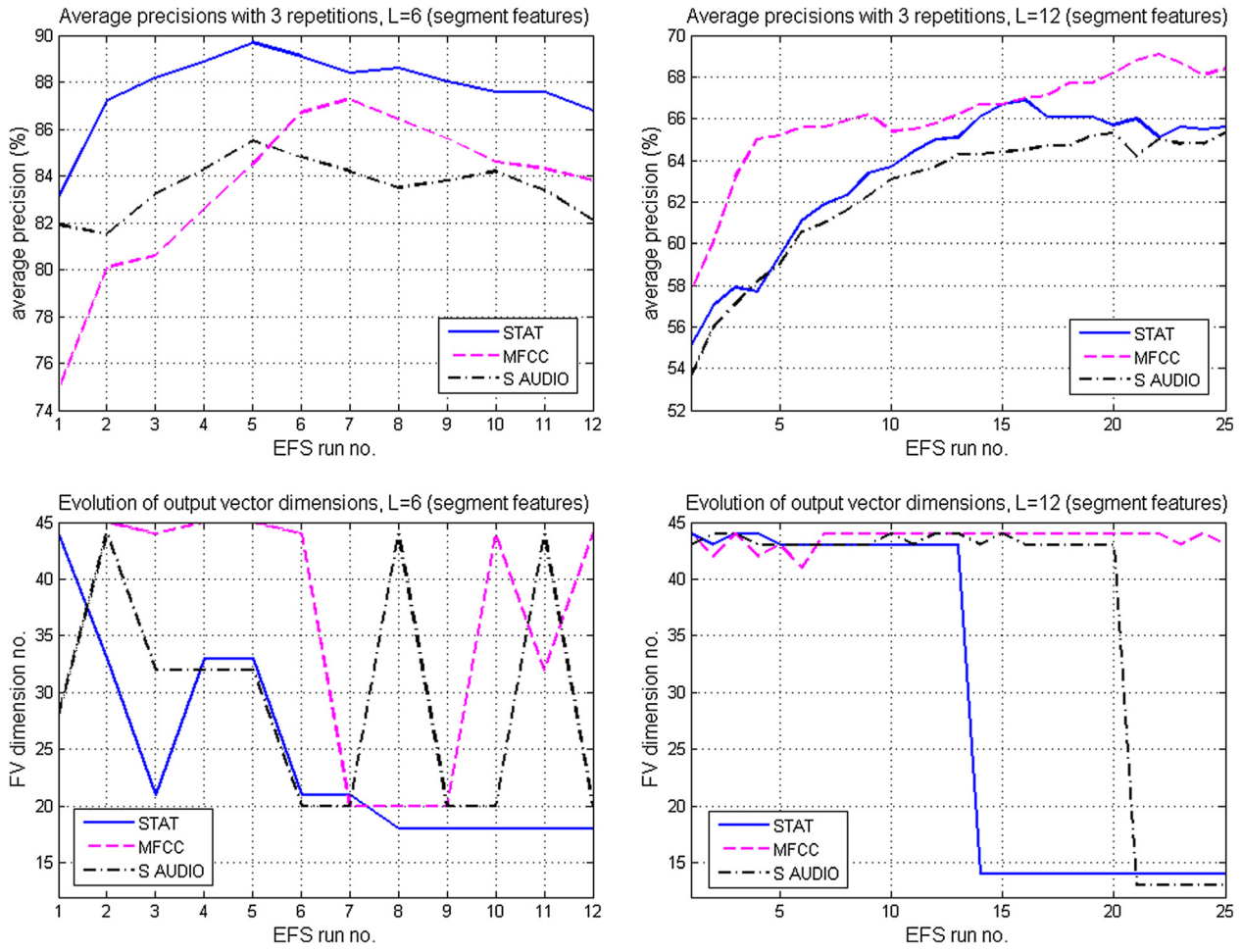

Figure 9 The consecutive EFS runs with three repetitive optimization performances within each run. Significant further improvements were obtained to the AP results also with low output FV dimensions.

well as audio classification and retrieval performance. The process, EFS, is a *generalized form* of feed-forward neural networks. In addition to traditional ANNs, the EFS technique provides (1) numerous linear/nonlinear attribute operators, (2) a built-in feature selection scheme, and (3) dynamic output (layer) dimension.

The experimental results obtained from clustering, *K*-means classification, and several audio retrieval tasks demonstrate the ability of the technique to provide substantial improvements compared with the original features. It was also shown that the audio retrieval performance can be further improved by performing several synthesis runs, whereas comparable performance could not be achieved using concatenated SLP evaluations. Based on these experiments, the synthesis approach appears capable of producing more descriptive/discriminative (artificial) features than those designed and selected by humans (i.e., the traditional low-level audio features).

The underlying optimization algorithm used to discover the optimal feature synthesis parameters is based on PSO, extended in our study by the addition of three properties previously introduced in the literature. First, an MD PSO was applied, which enables optimizing the output FV dimension during the synthesis process. Second, an FGBF technique was used to increase the probability of converging to the global optimum of the search space and third, the heterogeneous particle behavior was addressed via the swarm in order to avoid local optima in the fitness function surface.

In this study, the optimization of output vector dimension was considered only with respect to classification and retrieval performance. As a result, the best output vector dimension found by the EFS was usually higher than that of the original FS. This may be an important aspect for certain applications where there is considerable post-processing for the output vectors. However, an appropriate upper limit for the output vector dimensionality can be determined and tuned to the specific problem to be solved. Some of the experiments using the basic 6-class database suggest that when it comes to discriminating features of a relatively small and distinguishable database, the EFS technique may not be worth implementing. However, with the larger and more overlapping (in terms of the pre-determined data classes) 12-class database, regular neural networks were not able to achieve as significant improvements to the retrieval performance as those obtained with the EFS.

In general, the EFS framework can be used in any such tasks in which it is applicable to take advantage of enhanced feature discrimination. For example, content-based classification and knowledge mining are suitable venues for the proposed framework, as are those in which the quality and description power of the applied features play an essential role. In future research, the synthesis performance may be enhanced by experimenting with other optimization techniques (such as simulated annealing and GAs) or by optimizing the mathematical operators used. This would require analyzing operator selection during the synthesis process so that some statistical conclusions could be drawn about the selection behavior. A similar analysis could be applied, with caution, to the selection of the original features in order to avoid using "vain" (or very rarely selected) features. Unlike general feed-forward ANNs, where regular gradient-descent training methods (such as back-propagation) are designed to minimize fixed and differentiable fitness functions (such as MSE), the EFS technique can be used to minimize any types of fitness functions. In this way, the optimization tasks can be attuned to the specific goals of the research, such as audio retrieval, as closely as possible. Designing additional fitness functions for particular problems is thus one of the main advantages of the proposed feature synthesis scheme. Such issues will be addressed in future research.

Endnotes

[a]TIMIT (Texas Instruments, Massachusetts Institute of Technology) is a corpus of phonemically and lexically transcribed speech of American English male and female speakers of different dialects. [b]RWC (Real World Computing) Music Database is a copyright-cleared music database that is available to researchers as a common foundation for research (http://staff.aist.go.jp/m.goto/RWC-MDB/). [c]The *StockMusic.com* web shop (http://www.stockmusic.com/). [d]Techila Technologies Ltd., *Techila Grid*, http://www.techila.fi/.

Competing interests

Toni Mäkinen is a PhD student at the Tampere University of Technology (TUT), from where he receives salary. In order to graduate, scientific publications are required. The TUT will gain financially from the graduation itself in the future, but not separately from any specific publication (such as this one). The authors thus declare that they have no competing interests.

Acknowledgments

The authors would also like to thank Toni Heittola, Anna-Maria Mesaros, and Tuomas Virtanen from the Tampere University of Technology for kindly providing several audio databases for testing during this research.

Received: 13 March 2012 Accepted: 13 August 2012
Published: 11 September 2012

References

1. R.O. Duda, P.E. Hart, D.G. Stork, *Pattern Classification*, 2nd edn. (Wiley-Interscience Publication, 2001)
2. R.E. Bellman, *Adaptive Control Processes—A Guided Tour (Princeton University Press* (Princeton, NJ, 1961)
3. D.E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, 1st edn. ((Addison-Wesley Longman Publishing Co, Boston, MA, 1989)
4. J. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection* (MIT Press, Cambridge, MA, 1992)

5. C.-L. Huang, C.-J. Wang, A GA-based feature selection and parameters optimization for support vector machines. Expert Syst. Appl. **31**(231–240) (2006)
6. L. Zhuo, J. Zheng, F. Wang, X. Li, B. Ai, J. Qian, A genetic algorithm based wrapper feature selection method for classification of hyperspectral images using support vector machine. Proc. SPIE **7147**(71471J) (2008). doi:10.1117/12.813256
7. J. Kennedy, R. Eberhart, *Particle swarm optimization, in Proceedings of the IEEE International Conference on Neural Networks*, 4th edn. (, Perth, Australia, 1995), pp. 1942–1948
8. R.M. Ramadan, R.F. Abdel-Kader, Face recognition using particle swarm optimization-based selected features. Signal Process. Image Process. Pattern Recognit. **2**(2), 51–66 (2009)
9. L.-Y. Chuang, H.-W. Chang, C.-J. Tu, C.-H. Yang, Improved binary PSO for feature selection using gene expression data. Comput. Biol. Chem. **32**(1), 29–37 (2008)
10. S.-W. Lin, K.-C. Ying, S.-C. Chen, Z.-J. Lee, Particle swarm optimization for parameter determination and feature selection of support vector machines. Expert Syst. Appl. **35**(4), 1817–1824 (2008)
11. Y. Liu, Z. Qin, Z. Xu, X. He, Feature selection with particle swarms, in *Computational and Information Science, in Lecture Notes in Computer Science*, ed. by J. Zhang, J.-H. He, Y. Fu, 3314th edn. (Springer, Berlin, 2005), pp. 425–430
12. K. Geetha, K. Thanushkodi, A. Kishore Kumar, *New particle swarm optimization for feature selection and classification of microcalcifications in mammograms, in Proceedings of International Conference on Signal Processing, Communications and Networking* (, Chennai, India, 2008)
13. I. Guyon, A. Elisseeff, An introduction to variable and feature selection. Mach. Learn. Res. **3**(1157–1182) (2003)
14. D. Yang, L. Rendell, G. Blix, A scheme for feature construction and a comparison of empirical methods, in *12th International Joint Conference on Artificial Intelligence, IJCAI '91*, ed. by J. Mylopoulos, R. Reiter, 2nd edn. (Sydney, Australia, 1991), pp. 699–704
15. S. Markovitch, D. Rosenstein, Feature generation using general constructor functions. Mach. Learn. **49**(1), 59–98 (2002)
16. O. Ritthoff, R. Klinkenberg, S. Fischer, I. Mierswa, *A hybrid approach to feature selection and generation using an evolutionary algorithm* (Proceedings of the UK Workshop on Computational Intelligence, Birmingham, UK, 2002), pp. 147–154
17. K. Krawiec, B. Bhanu, *Visual learning by evolutionary feature synthesis, in Proceedings of the International Conference on Machine Learning* (, Washington, DC, USA, 2003), pp. 376–383
18. B. Bhanu, J. Yu, X. Tan, Y. Lin, *Feature synthesis using genetic programming for face expression recognition, in Proceedings of Genetic and Evolutionary Computation Conference* (, Seattle, WA, USA, 2004), pp. 896–907
19. J. Yu, B. Bhanu, Evolutionary feature synthesis for facial expression recognition. Pattern Recognit. Lett **27**(11), 1289–1298 (2006). Special Issue on Evolutionary Computation
20. Y. Lin, B. Bhanu, Evolutionary feature synthesis for object recognition. IEEE Trans. Syst. Man Cybern **35**(2), 156–171 (2005). C. Applications and Reviews (Special Issue on Knowledge Extraction and Incorporation in Evolutionary Computation)
21. F. Pachet, A. Zils, Evolving automatically high-level music descriptors from acoustic signals. LNCS **2771**(42–53) (2003)
22. F. Pachet, P. Roy, Analytical features: a knowledge-based approach to audio feature generation. EURASIP J. Audio Speech Music Process. **153017**, 23 (2009)
23. G. Barbieri, F. Pachet, M. Degli Esposti, P. Roy, *Is there a relation between the syntax and fitness of an audio feature? in Proceedings of the 11th International Society for Music Information Retrieval Conference* (, Utrecht, Netherlands, 2010), pp. 321–326
24. F. Mörchen, I. Mierswa, A. Ultsch, *Understandable models of music collections based on exhaustive feature generation with temporal statistics, in Proceedings of 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (, Philadelphia, PA, USA, 2006), pp. 882–891
25. I. Mierswa, K. Morik, Automatic feature extraction for classifying audio data. Mach. Learn. **58**(2–3), 127–149 (2005)
26. B. Schuller, S. Reiter, G. Rigoll, *Evolutionary feature generation in speech emotions recognition, in Proceedings of the IEEE International Conference on Multimedia and Expo* (, Toronto, Canada, 2006), pp. 5–8
27. S. Krakowski, F. Pachet, P. Roy, *Improving the classification of percussive sounds with analytical features: a case study, in Proceedings of 8th International Conference on Music Information Retrieval (ISMIR)* (Vienna, Austria, 2007), pp. 229–232
28. X. Shi, Y. Liang, H. Lee, C. Lu, L. Wang, An improved GA and a novel PSO-GA-based hybrid algorithm. Inf. Process. Lett. **93**(5), 255–261 (2005)
29. R.C. Eberhart, Y. Shi, Comparison between genetic algorithms and particle swarm optimization, in *Evolutionary Programming VII, in Lecture Notes in Computer Science*, ed. by V. Porto, N. Saravanan, D. Waagen, 1447th edn. (Springer, Berlin, 1998), pp. 611–616
30. S. Kiranyaz, T. Ince, A. Yildirim, M. Gabbouj, Evolutionary artificial neural networks by multi-dimensional particle swarm optimization. Neural Netw. **22**(1448–1462) (2009)
31. S. Kiranyaz, T. Ince, A. Yildirim, M. Gabbouj, Fractional particle swarm optimization in multi-dimensional search space. IEEE Trans. Syst. Man Cybern. B. Cybernetics Journal **40**(2), 298–319 (2010)
32. S. Kiranyaz, J. Pulkkinen, T. Ince, M. Gabbouj, *Multi-dimensional evolutionary feature synthesis for content-based image retrieval, in Proceedings of the IEEE International Conference on Image Processing* (, Brussels, Belgium, 2011), pp. 11–14
33. A.P. Engelbrecht, *Heterogeneous particle swarm optimization, in Proceedings of 7th International Conference on Swarm Intelligence* (, Berlin, Germany, 2010), pp. 191–202
34. K. West, S. Cox, *Features and classifiers for the automatic classification of musical audio signals, in Proceedings of 5th International Conference on Music Information Retrieval* (Barcelona, Spain, 2004)
35. S. Kiranyaz, A.F. Qureshi, M. Gabbouj, A generic audio classification and segmentation approach for multimedia indexing and retrieval. IEEE Trans. Audio Speech Lang. Process. **14**(3), 1062–1081 (2006)
36. S. Kiranyaz, M. Gabbouj, A generic content-based audio indexing and retrieval framework. IEEE Proc. Vis. Image Signal Process. **153**(3), 285–297 (2006)
37. J. Bullock, Libxtract, A lightweight library for audio feature extraction. (2012). http://www.postlude.co.uk/postlude/downloads/LibXtract:_a_lightweight_feature_extraction_library.pdf. Accessed 23 February 2012
38. F. Pachet, P. Roy, *Exploring billions of audio features, in Proceedings of International Workshop on Content Based Multimedia Indexing* (, Bordeaux, France, 2007), pp. 227–235
39. G. Peeters, A large set of audio features for sound description (similarity and classification) in the CUIDADO project. (2003). http://recherche.ircam.fr/equipes/analyse-synthese/peeters/ARTICLES/Peeters_2003_cuidadoaudiofeatures.pdf. Accessed 23 February 2012
40. J. Breebaart, M. McKinney, *Features for audio classification, in Proceedings of Philips Symposium of Intelligent Algorithms* (Eindhoven, Netherlands, 2002)
41. Y. Shi, R.C. Eberhart, Parameter selection in particle swarm optimization, in *7th Annual Conference on Evolutionary Programming, EP '98*, ed. by V.W. Porto, N. Saravanan, D.E. Waagen, A.E. Eiben, 7th edn. (Springer-Verlag, London, 1998), pp. 591–600
42. J. Kennedy, *The particle swarm: social adaptation of knowledge, in Proceedings of the IEEE International Conference on Evolutionary Computation* (, Indianapolis, IN, USA, 1997), pp. 303–308
43. J. Kennedy, *Bare bones particle swarms, in Proceedings of the IEEE Swarm Intelligence Symposium* (, Indianapolis, IN, USA, 2003), pp. 80–87
44. B.S. Manjunath, P. Salembier, T. Sikora (eds.), *Introduction to MPEG-7: Multimedia Content Description Interface (Wiley* (San Francisco, CA, 2002)
45. T.G. Dietterich, G. Bakiri, Solving multiclass learning problems via error-correcting output codes. Artif. Intell. Res. **2**(263–286) (1995)
46. W.J. Bainbridge, W.B. Toms, D.A. Edwards, S.B. Furber, *Delay-insensitive, point-to-point interconnect using m-of-n codes, in Proceedings of the 9th International Symposium on Asynchronous Circuits and Systems* (, Vancouver, BC, Canada, 2003), pp. 132–140

doi:10.1186/1687-4722-2012-23
Cite this article as: Mäkinen *et al.*: **An evolutionary feature synthesis approach for content-based audio retrieval.** *EURASIP Journal on Audio, Speech, and Music Processing* 2012 **2012**:23.

Tampereen teknillinen yliopisto
PL 527
33101 Tampere

Tampere University of Technology
P.O.B. 527
FI-33101 Tampere, Finland

ISBN 978-952-15-3075-3
ISSN 1459-2045