Audio Event Classification for Urban Soundscape Analysis

Jon Stammers

PhD

University of York
Electronics

October 2011

Abstract

The study of urban soundscapes has gained momentum in recent years as more people become concerned with the level of noise around them and the negative impact this can have on comfort. Monitoring the sounds present in a sonic environment can be a laborious and time–consuming process if performed manually. Therefore, techniques for automated signal identification are gaining importance if soundscapes are to be objectively monitored.

This thesis presents a novel approach to feature extraction for the purpose of classifying urban audio events, adding to the library of techniques already established in the field. The research explores how techniques with their origins in the encoding of speech signals can be adapted to represent the complex everyday sounds all around us to allow accurate classification.

The analysis methods developed herein are based on the zero–crossings information contained within a signal. Originally developed for the classification of bioacoustic signals, the codebook of Time–Domain Signal Coding (TDSC) has its band–limited restrictions removed to become more generic. Classification using features extracted with the new codebook achieves accuracies of over 80% when combined with a Multilayer Perceptron classifier.

Further advancements are made to the standard TDSC algorithm, drawing inspiration from wavelets, resulting in a novel dyadic representation of time–domain features. Carrying the label of Multiscale TDSC (MTDSC), classification accuracies of 70% are achieved using these features.

Recommendations for further work focus on expanding the library of training data to improve the accuracy of the classification system. Further research into classifier design is also suggested.

Contents

List of Figures

List of Tables

List of Acronyms

ANN	Artificial Neural Network
CWT	Continuous Wavelet Transform
DTW	Dynamic Time Warping
DFT	Discrete Fourier Transform
EDS	Extractor Discovery System
FEC	Feature Extractor and Classifier
FFT	Fast Fourier Transform
GMM	Gaussian Mixture Models
HMM	Hidden Markov Model
ISVR	Institute of Sound and Vibration Research, Southampton
kNN	k-Nearest Neighbour
LDA	Linear Discriminant Analysis
LPC	Linear Predictive Coding
LPCC	Linear Predictive Cepstral Coefficients
LTW	Linear Time Warping
LVQ	Learning Vector Quantisation
MFCC	Mel Frequency Cepstral Coefficients
MLP	Multilayer Perceptron
MTDSC	Multiscale Time Domain Signal Coding
NFN	Noise Futures Network
NNT	Neural Network Toolbox
PSP	Positive Soundscapes Project
SOM	Self–Organising Map
SPL	Sound Pressure Level
STBF	Spectro-Temporal Box Features
STFT	Short-Time Fourier Transform
SVM	Support Vector Machine
TDNN	Time Delay Neural Network
TDSC	Time Domain Signal Coding
TES	Time Encoded Speech
TESPAR	Time-Encoded Signal Processing and Recognition
WVD	Wigner-Ville Distribution

Acknowledgements

I would like to thank my supervisor, Dr. David Chesmore, for involving me in this research from its beginnings, and his ensuing guidance and advice through to its conclusion.

I would like to thank the other members of the Bio–Applied Systems Engineering group, in particular Ollie, Naoko and James, for their company and encouragement both during and after the time we spent together in the lab.

I am grateful to the members of the ISRIE project team for their feedback and suggestions during our joint meetings, and for their excellent hospitality.

A great deal of thanks must go out to my friends and family, especially my parents, George and Nan and Grandad, for their continued support throughout this long process. Many extra thanks to Mum for the numerous hours spent proof–reading.

And finally to Rachel, without whom this simply would not have been possible. I am eternally grateful for your belief in me and unparalleled support.

Author's Declaration

I declare that the work contained herein has not been submitted for any other degree or to any other institution. Copies of conference proceedings are included in the appendices. The bibliographic entries for these proceedings are as follows:

- Stammers, J. and Chesmore, D. Instrument for soundscape recognition, identification and evaluation (ISRIE): signal classification. In *Proceedings of the Institute of Acoustics Spring Conference: Widening Horizons in Acoustics*, Vol 30(2), Reading, UK, 2008
- Stammers, J. and Chesmore, D. Instrument for soundscape recognition, identification and evaluation (ISRIE): signal classification. In *Proceedings of the 2nd ASA-EAA Joint Conference, Acoustics '08*, Paris, France, 2008
- Bunting, O., Stammers, J., Chesmore, D., Bouzid, O., Yun Tian, G., Karatsovis, C., and Dyne, S. Instrument for soundscape recognition, identification and evaluation (ISRIE): technology and practical uses. In *Proceedings of Euronoise 2009*, Edinburgh, Scotland, 2009

Chapter 1

Introduction

1.1 Research Summary

The focus of the research detailed in this thesis was to develop a stand-alone classification system capable of differentiating between audio events typically found in an urban soundscape. The most significant developments were made in how features were extracted from audio data, these features then being classified by standard neural networks. The feature extraction methods were purely time–domain, based and built upon a technique well–established for bioacoustic signal identification.

The presented research also contributed towards work package 2.2 of the project "Instrument for Soundscape Recognition, Identification and Evaluation" (ISRIE). This work package aimed to perform analysis and classification of audio signals found in an urban environment.

1.2 The ISRIE Project

The ISRIE project arose from the EPSRC Ideas Factory 'A Noisy Future: making the World sound better' held in January 2006. The proposed outcome of the ISRIE project can be described briefly as an intelligent noise metering system. The project was a collaboration between the universities of York and Newcastle, and the Institute of Sound and Vibration Research (ISVR) in Southampton.

1.2.1 Project Collaborators

An overview of the work carried out at each institution is given below:

- Work in **Newcastle** focused on experimenting with multi-sensor techniques for source localisation. They also looked at the design and construction of an ad-hoc wireless sensor network for wide-area localisation, including a review of various network topologies. Further details and results of this work can be found in Atmoko, Tian, and Fazenda (2007) and Atmoko, Tan, Tian, and Fazenda (2008).
- The research at **York** was split into two projects. One developed methods for source separation using a B-format microphone which included the calculation of a source's direction of arrival to enable separation (Bunting, 2011). The work detailed in this thesis describes the second project involving the research of methods for analysis and classification of single–source audio signals.
- **ISVR** investigated the potential uses of ISRIE with respect to existing noise legislation. This included liaising with potential beneficiaries of an *intelligent noise meter*, such as ISRIE, to discover how it could assist their work and if they thought it should have any particular functions. ISVR's extensive knowledge of legislation also aided in the decision of which sounds should be included in the classification process (see Section 1.5). Creating and maintaining a database of audio recordings was also undertaken by ISVR. This database was made available to the other collaborators via a secure FTP server. Many of these recordings contributed to the development and testing of the classification techniques discussed in this thesis. More details of the work carried out by ISVR can be found in Karatsovis and Dyne (2008).

Collaboration between the three institutions was achieved through regular meetings. During these meetings each institution reported on progress, and discussions took place to decide how work would move forward.

The ISRIE project group also contributed to the Noise Futures Network (NFN) through attendance and presenting at meetings. The aim of the NFN is to make soundscapes a more prominent area of study by facilitating multi–disciplinary research into this field. It is hoped that as a result of the work of the NFN more aurally-conscious decisions will be made for future urban developments and that the general public will become more aware of their sonic environment.

1.2.2 Motivation for the ISRIE Project

At present, noise level measurements, such as those taken by local councils when investigating noise complaints, are made using the A-weighted sound pressure level (SPL) at a point over a period of time. This provides the investigator with limited information, such as the average and peak sound levels at that point, but the measurement does not give them any information as to the cause of the sounds. Consequently, a rural village, for example, could be deemed as 'noisy' because a train passes through close to the measurement location once or twice a day giving a high peak sound level. The village could also be deemed 'noisy' due to high peak levels caused by a dawn chorus from local birds. These two examples illustrate that both positive and negative sounds can cause an area to be labelled as 'noisy' even though the dawn chorus may not be a particular nuisance to local residents.

One method of overcoming this issue is to make recordings of the environment under investigation and perform manual analysis of the recordings; this is known as attended monitoring. If a timecode synchronisation was provided between the recordings and SPL measurements, the method would supply a detailed report of which sounds occurred at what time and how loud they were. However, this method of analysis perhaps introduces more problems than it solves. Firstly, performing such a recording over a long period of time would require a great deal of data storage; and secondly, it would require a person to listen to and review all of the recorded data, which is a time-consuming and laborious process. The ISRIE system would simplify this task by providing similar data, i.e. what sound occurred when, but with none of the problems given above.

In a study of the occurrence of over snow vehicles in the Yellowstone National Park (Burson, 2006), audio samples were recorded in the field and then analysed in an office environment. Burson described how the volume of playback for some recordings had to be increased by 10 dB to approximate the audibility that would have been available in the field. This type of study would benefit from a system such as ISRIE.

The following section discusses a case study carried out to identify the range of problems the ISRIE system could overcome.

1.2.3 Application of ISRIE: Case Study

To illustrate the potential uses of ISRIE, case studies were formulated by colleagues at ISVR. The aim of these case studies was to demonstrate how using just the conventional sound level for specific public administration applications is not always adequate (Bunting et al., 2009). One such case study is briefly described here.

The case study was concerned with a residential development proposal. The area that was to be developed was in close proximity to a number of noise sources that could cause problems to the eventual residents. Two of the main noise sources were local residential roads and a railway line.

The site was surveyed over a 24-hour period using unattended monitoring with sound level meters. The meters logged data for L_{Aeq} and $L_{Amax,s}$[1] in accordance with the guidelines set out by Planning Policy Guidance (PPG) 24 (Department for Communities and Local Government, 1994). This document also sets out a number of Noise Exposure Category (NEC) bands which determine the outcome of a noise assessment for a proposed development.

In the case study, one of the meters logged a daytime L_{Aeq} level of 65 dB placing the site into NEC band 'C', if the main cause of this reading was road traffic or mixed noise sources. Band 'C' dictates that '*Planning permission should not normally be granted*' (Department for Communities and Local Government, 1994). However, if the main source of the daytime reading was due to rail traffic, the site would be placed in NEC band 'B' where '*Noise should be taken into account*' when considering the planning application.

Another reading that could have affected the outcome of this site's NEC placement was $L_{Amax,s}$ which reached 82 dB. According to PPG 24 a site would be placed into NEC band 'C' when there are frequent events that exceed 82 dB $L_{Amax,s}$. The sound level meters were configured to make recordings of any such events to allow further analysis. It was found that out of the 70 events that reached 82 db $L_{Amax,s}$, 65 of these were due to birdsong. Therefore, these readings would not affect the PPG 24 assessment. It is worth noting that analysis of the recordings took 4 man–hours.

The case study illustrates how an instrument such as ISRIE could aid in legislative procedures and remove the need for unnecessarily spending time on analysing

[1] L_{Aeq} is the average A-weighted sound pressure level over a given measurement period. $L_{Amax,s}$ is the maximum A-weighted sound pressure level over a given measurement period based on a slow weighting.

recordings of events exceeding a predetermined level. An *intelligent* noise meter would have been able to inform the user which sound categories contributed to the reading of 65 dB L_{Aeq}. It would also have been able to indicate which sound category was responsible for the events exceeding 82 dB $L_{Amax,s}$ without the need for an extra four man-hours to analyse recordings.

1.3 Research Aims

The final ISRIE system requires the ability to separate different sound sources in an acoustic environment and to provide a system that can overcome at least some of the need for attended monitoring (see above). It was stated that the presented research contributes towards the identification of sound sources. The acoustic sources present in the environment will be captured by the ISRIE system via a microphone and converted into a digital representation of the audio signal. Consequently, the aims of the presented research were defined as:

1. Perform a comprehensive review of the literature to identify classification systems that already exist and where contributions can be made to the field of signal classification.
2. Develop and test a classification system suitable for identifying audio signals and determining the category to which the signals belong.

By reviewing the work of others and gaining an insight into the status quo of the field, the first aim allowed the research to become focused on a particular strand of signal classification rather than attempting to tackle the whole field. Inspiration was drawn from a range of sources to develop the direction taken in fulfilling the second aim.

1.4 Classification Systems

A typical classification system is shown in Figure 1.1. The two main parts of a classification system are evident; a feature extractor and a classifier. Some systems also use pre- or post-processing. Pre-processing, could include, for example, normalising

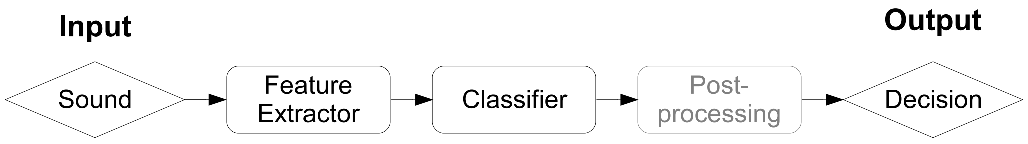


Figure 1.1: *Flow diagram showing the basic structure of a classification system. Adapted from Allegro, Büchler, and Launer (2001).*

the audio signal. Post-processing is often used after the classifier to correct for errors in classification.

When designing a classification system it is important to consider the signals that are to be classified. This will assist in deciding the most suitable feature extractor and classifier and also whether any pre- or post-processing is required. For the classification section of the ISRIE system the input will be an electronic representation of an acoustic signal. Chapter 2 discusses some of the available feature extractors and classifiers.

1.5 Audio to be Classified

During the early stages of the ISRIE project, the project partners discussed which sounds the ISRIE system should aim to classify. A set of audio categories for identification was developed based on the relevant standards – PPG 24, which relates to minimising the adverse impact of noise (Department for Communities and Local Government, 1994) and BS4142, which relates to rating the industrial noise affecting areas comprising residential housing and industrial units (British Standards Institute, 1997). It was decided to aim to place the sounds into categories rather than individual sounds because there is no real need to identify every individual sound, as highlighted by the case study in Section 1.2.3.

The categories chosen for identification are given in Table 1.1. There are many other sound categories which could be identified in an urban soundscape, but those shown in Table 1.1 were determined to be the most important for the ISRIE project to recognise.

Although the main purpose of ISRIE was always to identify sounds arising in an urban soundscape, research was also carried out in recognising other biophonic sounds. This work was undertaken to allow comparisons to be made with prior work which used similar techniques for feature extraction and classification, and to

Table 1.1: *Sound categories to be identified by ISRIE.*

Transportation	**Mechanical**	**Biophonic**
Road	Air Conditioning	Bird Song
Rail	Ventilation	
Air	Building Works	

broaden the scope of potential uses for the final ISRIE system. Further details of this work are given in Chapter 4.

1.6 Summary of Novel Research

The presented research aimed to develop a classification system capable of discriminating between signals belonging to different urban sound categories. Persuing this aim gave way to the following areas of novel research:

- A comprehensive review of the literature pertaining to the study of urban soundscapes and signal classification. The latter was considered in many fields of research including general audio classification and bioacoustic signal recognition. From this review, techniques were identified for further investigation (Chapter 2).
- The development of TDSC to provide a generic codebook suitable for extracting features from urban audio signals. TDSC had previously been used for bioacoustic and health monitoring applications (Chapter 3).
- The application of the new TDSC codebook in classifying urban audio signals as well as a bioacoustic application (Chapter 4).
- Seeking inspiration from wavelet analysis to combine the zero–crossings based analysis methods of TDSC with the dyadic representation of a signal resulting in a technique called Multiscale TDSC (MTDSC). (Chapter 5).
- The application of MTDSC to classify urban audio events in combination with MLP and LVQ networks (Chapter 5).
- The application of MTDSC in classifying bioacoustic signals and the recommendations for future uses of MTDSC as a result (Chapter 5).

Publications arising from the presented research are provided in Appendix G.

1.7 Chapter Summary and Thesis Overview

This chapter has introduced the research detailed in this thesis and set the wider context of the work. Whilst the experimental data discussed herein is valid as a stand–alone piece of research, the findings also make a contribution towards the ISRIE project. The involvement with the ISRIE project influenced the choice of audio data to be classified and the categories into which signals are classified.

The aims of the presented research were outlined and the reader introduced to what constitutes a signal classification system. Some of the underlying theory was presented, particularly in relation to the core parts of a classification system: the feature extractor and classifier. Details were given of the categories into which audio was classified. The choice of categories was based on collaboration with ISRIE project colleagues and the uses of ISRIE with regard to the relevant standards.

Finally, a summary was included of the novel contributions made by the presented research. Chapter references were provided to guide the reader towards the details of the novel work.

The remainder of this thesis is organised as follows:

Literature Review Previous work in the fields of urban soundscape analysis and signal classification are considered in detail in Chapter 2. Prior research relevant to the topic of this thesis is examined and critiqued.

Development of a Generic Codebook Chapter 3 explains the concepts and origins of TDSC. The limitations of the standard TDSC methods are discussed and developments made to produce a generic codebook.

Classification using TDSC The generic codebook is used to classify both urban audio events and bioacoustic signals in Chapter 4. Details of the training and testing methodology are also provided together with experimental results.

Further Developments to TDSC Inpiration is drawn from wavelets in developing a novel feature extraction method in Chapter 5. Results are presented for the training and testing performed with the new technique.

Discussion and Further Work In Chapter 6, the findings of the presented research are discussed in detail including its strengths and limitations. The possibilities for continuing the research in the future are also considered.

Conclusions The final chapter draws conclusions on the project.

Appendices Full code listings and additional material referred to throughout the text are included in the appendices. Also included are several publications relating to the presented research.

Chapter 2

Signal Classification

In this chapter, prior work carried out in the field of signal classification will be discussed. This will provide an overview of both the signal analysis techniques that can be used for classifying signals and the different areas of research where such techniques are applied. An intended outcome of this review of previous work was to identify suitable feature extraction and classifier techniques which could be applied in the ISRIE system.

First, an overview will be given of research that has previously attempted to analyse a soundscape. As will be seen, some of these studies have required the researcher(s) to be present in the soundscape. Their results have either been established by simply listening to what sounds are occurring around them or by asking passers-by what they can hear. These studies provide results that are predominantly subjective and are based entirely on people's perception of the sounds they hear. Ideally, the ISRIE system will provide an output that is objective. That is, the output will inform the user which sounds or categories of sound were detected in a soundscape.

A discussion is also given of prior work that has tried to achieve an objective description of the sound events that occur within a given environment. The aims of the different studies vary from security implementations to similar aims to ISRIE; identifying the sources of noises that may be of nuisance to soundscape users.

The field of signal classification is by no means restricted to the identification of general audio events. Neither is it restricted to audio events. However, for the purposes of this project the focus will be on studies relating to the identification of audio. Much signal analysis and classification work has been undertaken in the fields of health studies (both animal and human), mechanical fault diagnosis, species identification, and music and speech analysis. The latter of these, speech and music, will only be discussed very briefly because, as it will be seen, the techniques usually employed for such tasks often rely on the input signals having an alphabet, something which most sounds in an urban environment lack.

2.1 Soundscape Studies

There have been studies into both the contents of soundscapes and the importance of sound in urban environments, relating to both planning and human comfort.

Early studies of soundscapes were performed by the World Soundscape Project, established by R. Murray Schafer in the 1960s and 70s (The World Soundscape Project Website, 2007). “The Vancouver Soundscape” (Schafer, 1978) was the first detailed study performed by the group and focused heavily on the contents of the Vancouver soundscape and how it had changed in recent years. After giving some “ear-witness” accounts of how the sonic environment had developed, Schafer describes the keynote sounds, signals and soundmarks of the soundscape. Schafer also introduces the notion of Hi-Fi and Lo-Fi soundscapes. In a Hi-Fi one even the smallest of disturbances can communicate interesting information without being washed out by background noise (i.e. a high signal-to-noise ratio). Conversely, in a Lo-Fi soundscape a small disturbance would be lost in an “overdense population of sounds”. A Lo-Fi soundscape is therefore an undesirable situation for an environment.

To combat noise pollution in Vancouver, Schafer proposed the use of large-display noise meters at certain intersections in the city. These meters would give the ambient noise levels on a graded system: <40dB being “pleasant”, <60dB being “tolerable”, >70dB being “annoying” and above 85dB being “dangerous”. Such a system would only provide an indication for people at or near that intersection of what the level is, not what to do about it. For such a system to become useful these meters would need to be monitored and suggestions put forward for ways to combat any “annoying” or “dangerous” noise levels.

What Schafer illustrated in this soundscape study is that, even in the 1970s soundscape monitoring was an important part of town planning and development.

Echoing Schafer's views in a study examining the influence of sound on landscape values, Carles, Barrio, and de Lucio (1999) found

> "...a need to identify places or settings where the conservation of the sound environment is essential ..."
>
> (Carles et al., 1999, p. 191)

This need stems from the fear that a landscape can be negatively altered both by visual change (buildings and roads) and by a change to the acoustic environment.

Further studies into how users of soundscapes perceive the sonic environment have been carried out as part of the "Positive Soundscapes Project" (PSP - a sister project to ISRIE). Cain, Jennings, and Poxon (2011) used data generated by PSP to determine descriptors for assessing how positive a soundscape was. Soundscapes were plotted on axes having the labels "calmness" and "vibrancy" - soundscapes with high values of each were generally seen to be positive. The aim of the work by Cain et al. (2011) was to develop techniques that can be used to aid soundscape design.

A related study by Jennings and Cain (2012) applied automotive product design techniques to analyse soundscapes. In doing so, the authors aimed to define what is meant by a "positive" soundscape. Jennings and Cain report that this is largely dependent on the user and the activities they are carrying out within the soundscape.

Yang and Kang (2005) provide a subjective study investigating the effect of the sonic environment on those situated in it. They performed an evaluation of acoustic comfort in open public urban spaces through the use of questionnaires and sound level measurements. Their results found that subjective evaluations of sound levels correlated well to the measured level. However, considerable differences were found between the subjective level evaluation and acoustic comfort evaluation. This is actually a fairly obvious result; loud sounds are not necessarily always unpleasant. So Schafer's proposed noise meter system will not always prove valid (for example, a flock of birds singing could produce sound levels in excess of 70dB and not be considered annoying). Yang and Kang show that the quality of a soundscape can only be described as good/bad, pleasant/unpleasant, etc. with considerable human input. During meetings with the ISRIE project partners it was established that subjective classification would not be an aim for the ISRIE system as this would rely too heavily on human perception.

Other studies have also combined subjective study with accurate measurement, such as Ge and Hokao (2005) and Raimbault, Lavandier, and Bérengier (2003). These studies, like that of Yang and Kang (2005), perform their evaluation through questioning passers-by and attended monitoring, including making recordings of particular urban areas and measuring sound levels. Hall, Irwin, Edmondson-Jones, Phillips, and Poxon (2011) compared results from laboratory-based listening tests with sound level measurements of soundscapes in their study regarding the *pleasantness* of various soundscapes. Hall et al. (2011) report that acoustic and psychoacoustic variables contributed little towards the perception of how pleasant a soundscape was.

Another field that aims to quantify the urban soundscape is that of soundscape cartography. Kang and Servign (1999) propose a system which provides the user with an animated map of an urban soundscape for a given time period. It is proposed that the system could also be used for simulated sound impact for new urban developments.

A recognition tool would aid the evaluation processes in studies such as those discussed above by providing the researcher with, for example, a list indicating which sounds or which types of sound occurred and at what time they occurred. Combining a recognition tool with the animated cartography proposed by Kang and Servign (1999) could result in a very powerful system providing a great wealth of information to urban planners, dwellers and researchers. A system such as that proposed by the ISRIE project would negate the need for the researcher to be in the soundscape for extended periods of time to manually collect such information.

2.1.1 Audio Identification Instruments

The ISRIE project, as the name suggests, aims to develop an instrument for soundscape analysis. Prior studies which have had similar aims to the ISRIE project or tried to develop a similar instrument are discussed here.

The MADRAS project (Dufournet and Jouenne, 1997; Dufournet, Jouenne, and Rozwadowski, 1998) aimed to develop an instrument for real-time identification and quantification of various acoustic sources in a given acoustic environment, and assess their impact on the environment. The system comprised 5 steps:

detection of a signal (thresholding),

segmentation to identify sections of interest in the signal,

broad classification into one of 5 categories using Principle Components Analysis and a neural network,

expert classification from the broad category into a more specific category using third-octave features and a multilayer perceptron classifier,

post-processing such as calculating the contribution of the signal to the overall acoustic environment.

The stages of this system relate closely to the more generic stages introduced in Figure 1.1. However, in this system the classification process is split into two stages.

Dufournet et al. (1998) report that the broad and expert classifiers achieve 80% accuracy and propose how these results may be improved. At the time of writing, it has not been possible to find any further information on the MADRAS project beyond that given in the cited papers. It is therefore uncertain if the MADRAS project was successful in implementing the proposed system to develop "a new generation of acoustics instruments" or if the authors were able to achieve higher accuracies.

A more recent project developed an Ecological and Environmental Acoustic Remote Sensor - EcoEars (Gage, Maher, and Sanchez, 2005). The EcoEars system was proposed as a stand-alone unit (with portable and permanent variations) for monitoring noise contributions, environmental data (wind speed, temperature, etc.) and noise levels in a given location. The main interest of the EcoEars project was identifying mechanical noise and bird vocalisations in the vicinity of airfields. It is proposed that by monitoring these categories it could be possible to reduce the number of birdstrikes on civilian and military aircraft. This could result in cost savings to the civilian and military aviation industries and fewer endangered birds being killed.

Gage et al. (2005) propose short-time Fourier analysis for feature extraction and a pattern matching method for classifying. This combination results in recognition accuracy for bird vocalisations of 97.5%. However, as acknowledged by Gage et al., pattern matching as a means of classification requires a great deal of storage space. This is not a problem when the application only requires the identification of two categories. However, if there are more categories to classify into, the storage requirements become too large. Therefore, pattern matching is not an ideal candidate for the classifier stage of the ISRIE system.

2.2 General Audio Event Classification

There have been a number of studies that have also sought to accurately identify audio events that fall into the general sound category. That is, the sounds are those which might be heard in everyday environments, but not necessarily an urban environment. Inspiration can be drawn from such studies because they are trying to identify a number of sounds that, like those of an urban environment, have obviously different characteristics, either in their frequency components or temporal properties, or both. This section will discuss some of the studies in this area and identify the feature extraction and classifier techniques the researchers have chosen to implement.

2.2.1 Audio Event Classification for Security

A comparative examination of various feature extractors and classifiers for the recognition of environmental sounds is provided by Cowling and Sitte (2003). The application of their work is focused on a component of a security system. The authors state that sound recognition systems have an advantage over video surveillance cameras in a security system because line of sight is not required for a sound recognition system to function correctly. The work of Cowling and Sitte attempts to discover which feature extractor and classifier (FEC) pair provides the best classification result of non-speech environmental sounds. They also examine how well the various FEC pairs can classify speech and musical instrument sounds.

Cowling and Sitte refer to the findings of previous studies to construct comparison tables of feature extraction and classification techniques that are suitable for classifying audio events. They rely on the results of some of these earlier studies to determine which FEC pairs provide the best results for a particular type of sound. For example, a Q-cepstral coefficient feature extractor paired with a Gaussian Mixture Model classifier were used for musical instrument recognition, but not for environmental sounds. The comparison tables showed that some techniques are not suitable for non-speech sound recognition. In particular, feature extraction techniques which require an alphabet of subword features are not particularly suited to the classification of environmental sounds. It is pointed out that this is because environmental sounds do not have the phonetic structure that speech has and that there is no set alphabet for environmental sounds. Hidden Markov model-based classifiers are mentioned as being difficult to implement for environmental sound recognition as they require an alphabet.

The experiments conducted by Cowling and Sitte consisted of testing eight different sounds. The sounds were chosen for their likelihood of being classified in a sound surveillance system. All of the possible combinations of FEC pairs are tested in this study, each FEC pair being tested using a *jack–knife* method. This is a method in which a classification system is trained with all data except for the sound sample that will be tested. This sound sample is then used to test how accurately the system can classify that particular sound type.

The results of this work show that a Dynamic Time Warping (DTW) classifier with either Mel Frequency Cepstral Coefficient (MFCC) features or Continuous Wavelet Transform (CWT) features gave a correct classification rate of 70%. None of the other classifier models, including Learning Vector Quantisation (LVQ), Artificial Neural Networks (ANN) and Gaussian Mixture Models (GMM), gave comparable accuracies for classification.

The work of Cowling and Sitte provides an excellent overview of some techniques that could be applied in classifying urban audio events. The results produced by some of the combinations are promising and Cowling and Sitte suggest ways in which these results could be improved. This study will provide a useful reference when discussing the techniques that will be tested for use in the ISRIE system and when establishing the testing methods for any classification systems that are developed. Cowling and Sitte also provide a further use for the proposed ISRIE system.

Another form of security is the focus of Ghiurcau, Rusu, Bilcu, and Astola (2012). The authors propose the use of audio classification methods for protecting wild areas from intruders that may be damaging the areas (through hunting, fishing, or deforestation). Ghiurcau et al. emphasise that it is difficult to monitor large reserves with just foot patrols and suggest that remote acoustic sensors could be used to raise the alarm when an intruder is present.

Ghiurcau et al. (2012) provide a comparison between low-complexity and standard classification methods. For the low-complexity methods, they extract features using Time-Encoded Signal Processing and Recognition (TESPAR) - a method which is of significant importance to the presented research and one that is discussed in greater detail in Chapter 3. The TESPAR features were classified using pattern matching with archetypal features stored in a database. For the standard methods, Ghiurcau et al. use MFCC features and Gaussian Mixture Model (GMM) and Support Vector Machine (SVM) classifiers.

The purpose of the work of Ghiurcau et al. (2012) is to determine if the low-complexity method of signal classification is accurate enough to be used as a remote acoustic detector of intruders. They conclude that the low-complexity methods are not as robust as the standard (and more complex) methods. Using TESPAR features extracted from clean recordings, an accuracy of 97% was achieved, compared to an accuracy of ~99% for the more complex standard classifcation methods. However, Ghiurcau et al. emphasise that the more complex methods are not suitable for implementation on simple controllers that could be distributed in remote wildlife areas.

The work of Ghiurcau et al. (2012) is significant in the context of the ISRIE project because the intended outcome of ISRIE was a remote sensor for classifying acoustic signals. Ghiurcau et al. provide a strong argument for the use of low-complexity feature extraction methods in a distributed sensor using simple controllers.

2.2.2 Environmental Noise Monitoring

A novel approach to environmental sound recognition (in this case urban sounds) is presented by Defréville, Roy, Rosin, and Pachet (2006). The authors aimed to develop a system capable of giving an efficient representation of an acoustic environment by computing a noise disturbance indicator based on identification of noisy sound sources. The goal of the work of Defréville et al. is similar to that of ISRIE in that the focus is on providing an objective assessment of a urban sonic environment. Defréville et al. refer in particular to European Directive 2002/49/EC (Council Directive (EC), 2002) which is still in force.

Defréville et al. adhere to the typical classification system structure introduced in Figure 1.1 stating that any classification system follows a two-phase scheme: a feature extraction phase and a classification phase.

Defréville et al. focus on improving the features presented to the classifier by using only *problem-specific* features (i.e. features that perform particularly well for the current problem). For example, a classifier will require a different set of features to accurately identify a car in comparison to a bird. The decision on which features are required for a particular recognition task are normally selected by a signal processing expert or researcher. The authors of this study replaced that person by using the Extractor Discovery System (EDS).

EDS was developed by Zils and Pachet (2004) to produce high-level music descriptors for use in a music information retrieval system. According to Defréville et al., EDS invents

> "... features as combinations of operators that are specific to the problem to solve ... features that a signal processing expert would never have come up with..."
>
> (Defréville et al., 2006, p. 2)

A typical feature generated by EDS might combine the following mathematical and signal processing operators: splitting the signal into frames of a given length, applying a window function (such as a Hanning window), performing a fast Fourier Transform, finding the square root of the signal, and extracting the MFCC properties of the signal.

There is no reason for a signal processing expert to come up with this particular combination of signal processing techniques, or even to combine this many techniques. However, the EDS has all of these techniques available and can combine them to suit the audio signal under analysis.

Defréville et al. trained and tested the classification system using 2000 audio samples from 16 categories, with only 6 of these being focussed on for recognition (cars, mopeds, motorcycles, buses, birds and voices). The authors state that each audio sample used in the study was arbitrarily fixed at 500ms. The audio samples were split into 2 groups: 66% for training, and the remaining samples for testing. Defréville et al. used two types of classifier in their study: Gaussian mixture models (GMM) and Nearest Neighbours (kNN).

Defréville et al. performed a number of classification tasks to test the validity of the extracted features in combination with the two types of classifier. These tests include, amongst others, splitting the data into *bird* sounds versus *other* sounds, *moped* sounds versus *other* sounds, *mechanical* versus *non-mechanical* sounds, etc. The authors report average results in the region of 90%. Interestingly, results are given with a *timbre* feature added to the set of features generated by EDS. Including the timbre feature increases the recognition accuracy by as much as 5 to 6% in some cases.

Defréville et al. raise two critical issues in the discussion of their work: *completeness* and *consistency*:

- The completeness issue is explained as being the difficulty encountered when trying to design descriptors that cover every sound category and also differentiate between each category. The authors highlight that this problem will be encountered whenever trying to perform multiple class classification. To overcome the completeness issue, Defréville et al. use a hierarchy of classifiers. Figure 2.1 shows an example of this approach.
- The consistency issue discussed by Defréville et al. relates to the fact that some categories of sound are not acoustically consistent. Example is given of the motorcycle category in which the sound of a Japanese four-cylinder engine sounds very different to the two-cylinder engine of a Harley-Davidson. To a human it would be obvious that these samples are both motorcycles. However, a classification system may place them in different (and incorrect) categories. No solution to the consistency issue is offered by the authors. However, this is an issue that must be borne in mind in any sound recognition task.

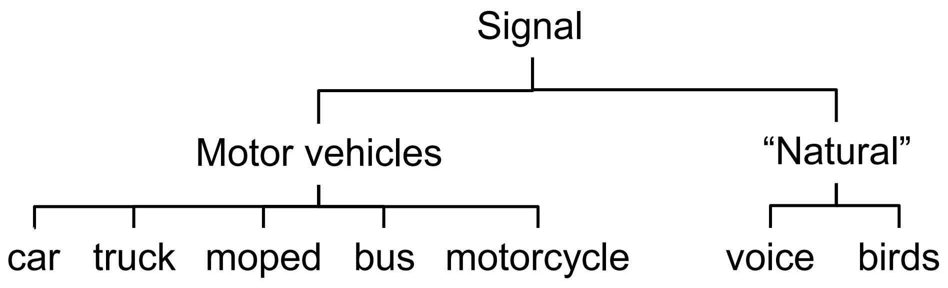


Figure 2.1: *Hierarchical classification approach adopted by Defréville et al. (2006).*

Botteldooren, Coensal, and Muer (2006) also propose a system for environmental noise monitoring, drawing on terminology more often used for describing music and examine the soundscape in terms of its *rhythm*. Botteldooren et al. examine the soundscape as a whole rather than the individual constituent parts in their study of the temporal structure of urban soundscapes. The notion of a soundscape being rhythmic is not new and this is acknowledged by the authors. Building on this, it is stated that *"...interesting, music-like temporal structure may become quite disturbing or annoying ..."* (Botteldooren et al., 2006, p. 108).

Determining if a soundscape is music-like is the main focus of the experimental work and this is achieved by comparing amplitude and pitch spectra of various soundscapes to $1/f$ noise. This is taken further to look specifically at the temporal structure of traffic noise because the authors believe this to be one of the biggest disturbances in the urban soundscape.

Their results show that many soundscapes exhibit some level of music-likeness but traffic noise is often more chaotic. The work of Botteldooren et al. provides a very different approach to soundscape evaluation when compared to the other studies in this section.

Another holistic approach to soundscape recognition is presented by Peltonen, Tuomi, Klapuri, Huopaniemi, and Sorsa (2002). The authors state that auditory scene recognition could be used to make hearing aids context-aware, thus improving the users experience of the environment. Peltonen et al. use many different soundscapes to train and test a number of combinations of features and classifiers in an attept to identify the best candidates for the task. The highest accuracy (68.4%) was obtained using a feature vector consisting of 5 different features fed into a nearest neighbour classifier. The authors report that this result is comparable to human listening test results of 70% accuracy.

The work of Peltonen et al. suggests that a complex feature extraction stage is necessary to achieve reasonably accurate recognition rates for whole soundscapes. Their approach, as well as that of Botteldooren et al., is perhaps not as useful to the ISRIE project given that it does not aim to identify the individual sounds contributing towards the soundscape. Nevertheless, the work of Peltonen et al. and Botteldooren et al. is important to the field of soundscape research as they both provide a more holistic method of analysis. Furthermore, these approaches negate the issues of completeness and consistency given by Defréville et al. (2006).

Krijnders, Niessen, and Andringa (2010) aimed to identify individual sounds within a soundscape and highlight a problem similar to the consistency issue of Defréville et al. (2006); some signals will have similar acoustic signatures but in fact will have very different meanings to a human listener. Krijnders et al. give the example of screams, singing and speech. To overcome this problem, Krijnders et al. use a technique they refer to as a “knowledge network” to provide contextual information for each sound being analysed. Classification results improve by 20% when the contextual information is combined with the features extracted from the raw signals for those categories that are very similar to others in their acoustic signature. The contextual information is applied to the system during a supervised training stage and relies on commentary of the signals being analysed for training.

2.2.3 Audio Content Analysis

Umapathy, Krishnan, and Rao (2007) discuss how the ability to classify a wide range of audio signals can play an important role in day-to-day life. Similar to the suggestion of Peltonen et al. (2002), Umapathy et al. propose that audio content analysis can be used to develop hearing aids that can adapt a noise reduction strategy depending on the noise sources present in the local environment. Such a device could significantly improve the quality of life for those with hearing impairments. Other applications of "audio environment detection" discussed by Umapathy et al. include wildlife conservation and in an educational tool for teaching children about different environments.

Umapathy et al. do not explicitly state that their work could be applied to audio event identification for soundscape analysis. However, there are aspects of this work that could be relevant to ISRIE. In particular, Umapathy et al. propose a method of heirarchical classification for their sound categories. The top tier of the heirarchy divides the sounds into *artificial* and *natural* sounds. Such a broad division is applicable in the ISRIE project because the sound categories detailed in Section 1.5 could also be split in this way; the artificial categories (traffic, building works, ventilation, etc.) of the ISRIE project are the categories most likely to be considered a noise nuisance (i.e. a sound that users of the soundscape find unpleasant). Therefore, one possible classification of sounds which are present in an urban soundscape could simply be natural vs. artificial, with the sounds identified as artificial being those causing a noise nuisance in the context of the ISRIE project.

Although this approach would work for many sounds present in an urban environment, it would not be suitable if other artificial sources were present which were not considered a nuisance. Umapathy et al. include a number of musical instruments in their classification tree on the side of artificial sounds. Furthermore, it is desirable to obtain more detail in terms of the specific category of sound which an audio event belongs to.

The classification system developed by Umapathy et al. (2007) uses a Local Discriminant Bases (LDB) technique to identify discriminatory time-frequency features. The time-frequency features are extracted using Wavelet Packets. Time-frequency features were chosen by the authors due to the non-stationarity of the signals they were attempting to classify. To validate the results of using the LDB-Wavelet combination, Umapathy et al. also extracted Mel Frequency Cepstral Coefficient (MFCC) features. To classify the features, the authors used a Linear Discriminant Analysis

(LDA) classifier. The testing method implemented is labelled as the *leave-one-out* method – one sample is excluded from the dataset during training and is used to test the trained system. This is repeated for each sample in the dataset. This method of testing is similar to the *jack-knife* method of Cowling and Sitte (2003).

Umapathy et al. (2007) tested their system using only the LDB-Wavelet features, only the MFCC features, and a combination of both sets of features. In general, the authors found that the results obtained using the LDB-Wavelet features were comparable to those obtained using the MFCC features. However, the MFCC features performed better for the natural sounds, and LDB-Wavelet features performed slightly better for the artificial sounds. Therefore, Umapathy et al. performed experiments using both sets of features to determine how well the methods complement each other. Their results show that combining the two methods of feature extraction did improve the overall classification results.

The authors report classification accuracies of 91% for the top tier of their heirarchy (using a combination of LDB and MFCC features). This result is comparable to the 96% accuracy achieved by Defréville et al. (2006) for their division of mechanical and non-mechanical sounds, as discussed above. The result reported by Defréville et al. was also achieved using a combination of features – EDS features and *timbre*. These high classification accuracies for seemingly simple divisions of sound categories are promising when considering the aims of ISRIE and the case study discussed in Section 1.2.3. An accurate recognition of natural vs. artificial, as used by Umapathy et al., or mechanical vs. non-mechanical, as used by Defréville et al., would give satisfactory results for discriminating between the traffic and bird song highlighted in the case study.

2.2.4 Further General Audio Classification Studies

The work of Cowling and Sitte (2003), Defréville et al. (2006) and Umapathy et al. (2007) described above are perhaps the most pertinent of the previous studies into general audio classification because of the breadth of the research and the areas of application. There are, however, many other pieces of prior research which aim to classify general audio, some of which are discussed here.

The work of Norris and Denham (2003) concentrates on the detection of *sound textures* which are defined as sound with local structure but no perceptually obvious long-term structure. Contrary to the typical classification system structure shown in 1.4, Norris and Denham do not use a feature extractor per se. Instead, the audio data is fed straight into a Self-Organising Map (SOM) either 2, 4, 8 or 16 samples at a time. The output from the SOM is used to construct a histogram of the redundancy of each input sample. The theory behind this approach is that if the SOM is trained for a particular audio sample, similar audio samples will give similar probability distributions.

The approach of Norris and Denham results in a SOM that is very good at recognising a particular type of sound, and each trained SOM could be considered an *expert* classifier. A system could be constructed using many SOMs, each SOM trained to identify a particular category of audio event, the overall system capable of determining which category an input belongs to. This approach would reinforce the *completeness* issue highlighted by Defréville et al. (2006).

Couvreur, Fontaine, Gaunard, and Mubikangiey (1998) discuss how earlier research into environmental noise monitoring have used *dynamic* feature extraction techniques (i.e. those which incorporate both time and frequency features) but then only used a *static* classifier to process the features. In doing so, much of the useful information gathered by the dynamic feature extractor is not fully utilised. Couvreur et al. propose a noise monitoring system for environmental sounds, the aim of the research being to show how classification results for environmental sounds can be improved by providing a dynamic classifier to complement dynamic features.

Couvreur et al. (1998) perform a frame-wise spectral analysis of the input signals and convert these to cepstral coefficients. After further data reduction using vector quantisation, the features are classified using a left-right HMM. The system was constructed using the Speech Training and Recognition Unified Tool (STRUT), a speech processing system developed by TCTS-Multitel at the Faculté Polytechnique de Mons in Belgium. Without making any significant changes to the speech processing system, Couvreur et al. achieve a recognition rate of 91%. This figure is increased to 95.3% after modifying the system.

The results given by Couvreur et al. (1998) contradict the statement given by Cowling and Sitte (2003), showing that a classifier which allegedly requires an alphabet can successfully recognise environmental sounds. Two further investigations also report excellent results using HMMs:

- Kim, Burred, and Sikora (2004) achieve excellent recognition accuracies using a continuous HMM in their comparative study of MPEG-7 and MFCC features

(∼95% accuracy) with 12 sound classes (including 2 speech classes).

- Ma, Milner, and Smith (2006) report accuracies in the range 93 – 96% using MFCC features and a left-right HMM classifier in their investigation into an acoustic environment classifier. The locations used to make the recordings were everyday environments such as an office, city centre street, railway station, etc..

These three studies which use a HMM classifier all outperform the systems proposed by Cowling and Sitte (2003), showing that classifiers typically used for speech recognition can have a place in the recognition of environmental sounds.

Ruvolo, Fasel, and Movellan (2010) provide a novel method of extracting features for classification by a Support Vector Machine (SVM). The method used by Ruvolo et al. extracts Spectro-Temporal Box Filter (STBF) features. These features analyse a signal over differing framelengths. Ruvolo et al. (2010) highlight that previous sound classification studies have often analysed signals over very short framelengths (tens of milliseconds) and then performed longer term analyses after classification to improve results. The authors provide results showing that their method (using short-, medium-, and long-term frames for feature extraction) improves classification accuracy for three different tasks - emotion detection (78.7%), detection of crying (95%), and music versus speech (98.4%).

2.2.5 General Audio Event Classification Summary

This section has provided a comprehensive overview of the research that has already been undertaken in the field of general audio event classification. Most of the reported research conforms to the standard feature extractor-classifier structure for the classification system. For each of these stages, many techniques have been identified and tested.

Of the reported results, the most promising are those given by Defréville et al. (2006) and Gage et al. (2005). However, both of the systems developed by these authors have their drawbacks. The system proposed by Defréville et al. requires some very complex feature extraction which would necessitate significant computing power. Gage et al. achieve high accuracy rates by using a pattern matching classifier. This type of classifier needs large amounts of data storage if there are many categories of sound to be classified.

Each of the studies using a HMM classifier report excellent classification accuracies. Contrary to Cowling and Sitte (2003), HMMs should not be ruled out as a potential candidate for the classifier of the ISRIE system.

One of the most interesting proposals found in the literature is that of expert classifiers and classifiers that differentiate between low numbers of categories. Umapathy et al. (2007) use a heirarchical classifier arrangement and report accuracies of up to 99% when selecting between two categories. In discussing the *completeness* issue, Defréville et al. suggest that hierarchical classification could be the solution to overcoming the issue. The idea of expert classifiers is also proposed by Dufournet and Jouenne (1997); Dufournet et al. (1998). Therefore, the hierarchical method of classification should be investigated in this thesis if using a single classifier does not yield acceptable results.

As discussed previously, the work of Ghiurcau et al. (2012) is significant to this project because they have shown that simple feature extraction methods can yield robust classification results. Using feature extraction methods with lower complexity would result in less computational power required for processing signals detected in the field. Consequently, a processor would have lower power requirements and a battery-operated sensor would require less maintenance.

The field of general audio event classification has yielded many techniques for feature extraction and classification. The following section delves into some of the other areas where audio signal classification is applied, providing an overview of these areas of application and identifying the techniques used.

2.3 Signal Classification in Other Research Fields

2.3.1 Human Health

Systems for automated classification have been developed for human health analysis. Their main purpose is to provide an objective and reliable method for assisting diagnosis. Often their application relates to an illness that requires a great deal of observation by a specialist to diagnose. An automated system can reduce the amount of time required from the specialist.

Distinguishing between voluntary and spontaneous cough is the focus of the research conducted by Van Hirtum and Berckmans (2002). Coughs are an important symptom in many respiratory diseases and often continuous observation of an individual is required to determine how the cough is affecting them. The methods proposed in this research aim to solely distinguish between voluntary and spontaneous cough with the eventual aim being to automatically determine if an individual has a particular disease from their cough. The feature extraction methods used are power spectral density analysis, principle component analysis and Euclidean spectral distance finding. These are then classified using nearest neighbour methods, learning vector quantisation, and fuzzy C-means clustering. A classification result of 96% is stated for distinguishing between the two types of cough but it is unclear which combination of methods provided this.

Dimoulas, Kalliris, Papanikolaou, Petridis, and Kalampakas (2008) discuss methods for unsupervised monitoring of bowel sounds captured via analysis of abdominal surface vibrations. The direction of the research is toward non-invasive methods for prolonged observation of patients to aid diagnosis of bowel functional disorders. This work introduces an interesting de-noising method referred to as Wavelet Domain Wiener Filtering. This stage of pre-processing is necessary to remove other sounds picked up by the sensors, such as patient movement and other internal organ sounds (in particular the lungs and heart). A number of features are extracted from the signals including temporal power features, spectral features and features gained from wavelet analysis. A neural network in the form of a multilayer perceptron is used as the classifier. The combination of wavelet features with the MLP network gave the best results averaging 95% recognition performance between 5 classes of illness.

Another application of non-invasive diagnosis is discussed by Güler and Übeyli (2006). In this study wavelets and probabilistic neural networks are used for classification of Doppler ultrasound blood flows. The motivation is similar to that of Dimoulas et al. (2008) in that observation over long periods of time is necessary for diagnosis.

Linder, Albers, Hess, Pppl, and Schnweiler (2008) propose a method for screening the voice disorder *Dysphonia*. Usually this can only be diagnosed at specialised centres where individuals with a great deal of training and experience can analyse the patient. Linder et al. aim to develop a portable system that will allow non voice-specialists in primary care to diagnose voice problems. Four features are used as inputs to a multilayer neural network: period correlation, jitter (period-to-period variability), shimmer (peak-to-peak amplitude variability) and glottal to noise excitation ratio. An improved backpropagation training algorithm is used to train the network and a correct classification result of 80% is achieved.

The health applications briefly discussed above all share an aim with the ISRIE project in that they are trying to remove long periods of observation of signals by humans. They all demonstrate that automated classification has wide-ranging applications and can help improve many aspects of human comfort. It should be noted that research also takes place to help diagnose animal illness (for example, Chedad, Moshou, Aerts, Hirtum, Ramon, and Berckmans (2001) study pig coughs).

2.3.2 Speech Recognition

Speech recognition is a large field of research with much interest. The applications of speech recognition include security (Dieckmann, Plankensteiner, and Wagner, 1997), improvement of hearing aids (Allegro, Büchler, and Launer, 2001) and spoken dialogue systems for interacting with users via speech (López-Cózar and Callejas, 2003).

Cowling and Sitte (2003) discussed how hidden Markov Models (HMMs) were not suitable for environmental sound recognition due to the lack of an alphabet for these sounds. However, López-Cózar and Callejas (2003) state that most automatic speech recognition systems use HMMs which are trained for the specific speech units under consideration - that is, a specific alphabet is developed for the sounds under consideration. HMMs are also applied in consumer applications, such as converting speech to text (Takaguchi and Nishimura, 2010).

An extensive discussion of HMM application in speech recognition, and an introduction to the theory of HMMs, is given by Rabiner (1989).

2.3.3 Vehicle Noise

The recognition and classification of vehicle noise has a number of applications. It is used for monitoring of vehicle health (He, Feng, and Kong, 2007; Rafiee, Rafiee, and Tse, 2010), assessing vehicle interior noise quality (Wang, Lee, Kim, and Xu, 2007) and for monitoring traffic levels in urban environments (Nooralahiyan, Kirby, and McKeown, 1998). Vehicle noise classification also has its uses in surveillance (Evans, 2010).

Schclar, Averbuch, Rabin, Zheludev, and Hochman (2010) recognise different vehicle types in their surveillance application. They state that vehicle recognition often relies on the premise that similar vehicles will emit similar sounds on a given road surface. These sounds can be used to assign an *acoustic signature* to a particular type of vehicle.

The methods of feature extraction used for vehicle noise analysis are as varied as those discussed in Section 2.2 in relation to general audio event classification. For example, Rafiee et al. (2010), Schclar et al. (2010), Wang et al. (2007), and Wu and Liu (2008) all use some form of wavelet analysis to generate their features. He et al. (2007) use independent components analysis to extract features from gearbox acoustic signals. Nooralahiyan et al. (1998) rely on Linear Predictive Coding (LPC) in monitoring road traffic. The work of Mazarakis and Avaritsiotis (2007) relies on a time-domain feature extraction method which is discussed further in Chapter 3.

2.3.4 Bio-Acoustic Species Identification

Automated recognition is a useful tool for the monitoring of species. A system that has been trained to recognise or distinguish between different animals or species can often perform this task more successfully than a human can if the sounds made are very similar to one another. The use of an automated system for species recognition also reduces the need for attended monitoring or recording large quantities of audio data. The work examined in this section looks at how feature extraction techniques and classifiers have been combined to perform recognition tasks.

Crickets are a regular target for automated classification. Chesmore (2001), Chesmore and Ohya (2004), Dietrich, Palm, Riede, and Schwenker (2004) and Lee, Chou, Han, and Huang (2006) all look at ways of classifying different species of cricket. Each piece of research applies different methods of feature extraction and classifiers. Chesmore (2001) and Chesmore and Ohya (2004) use time-domain signal coding for feature extraction and a multi-layer perceptron for the classifier. Under low-noise conditions this system produces a 100% classification accuracy for 13 species of cricket.

Dietrich et al. (2004) extract six different features from the audio streams (frequency contour, energy contour and temporal structure of the pulses; pulse distance density, pulse length and pulse frequency). Fusion methods are then used to combine these features for a more robust system. The classifier used is a fuzzy-k-nearest-neighbour method. Through the combination of features a classification accuracy of 93.5% is achieved.

Lee et al. (2006) produce a system which is used for identification of both Crickets and Frogs. During feature extraction, syllables are segmented and then the Mel-frequency cepstral coefficients (MFCC) are calculated for each. MFCCs are widely used in speech recognition tasks as they have the ability to represent the signal spectrum in a more compact form than traditional methods. A Linear discriminant analysis (LDA) method is used to derive a lower-dimensional feature vector, and a number of different classifiers are tested on this data for each species. Average classification results are 96.8% for 30 different frog calls and 98.1% for 19 different cricket calls.

Each of these 3 different studies show that it is possible for a classification system to distinguish between sounds that are very similar to one another. Both time- and frequency-domain techniques have been used together with a variety of classifiers. Although these studies are only looking at one particular type of animal with a lot of prior knowledge assisting system design, the principles can still be carried through to a more generic classification system.

Selin, Turunen, and Tanttu (2006) study a method of classification of bird sounds from eight different species. Five of these species produce inharmonic sounds and the remaining three produce harmonic calls. Wavelets are used as the feature extractor in this study with both a self-organising map and a multilayer perceptron tested as classifiers. The motivation for this work is to remove the subjectivity of bird sound classification when performed by humans through the use of an automated system and to improve on previous works by developing a system that can cope with inharmonicity and transients. The results for this study show a 78% correct recognition value for the SOM and 96% accuracy for the MLP. The work of Selin et al. (2006) shows that wavelet analysis can perform as an excellent tool for extracting features from inharmonic sounds with transients.

Bardeli, Wolff, Kurth, Koch, Tauchert, and Frommolt (2010) also classify bird sounds to monitor the populations of species that are of conservation interest. The authors use autocorrelation methods to classify time-frequency representations of bird sounds for two different species. Bardeli et al. achieve good results for each species of bird (∼90%). The authors discuss how detectors should be customised for particular sounds if they are to be used in areas where many sounds will be present. Bardeli et al. propose that this will increase the recognition accuracy for the target sounds which can be significant when monitoring species populations.

The studies discussed above focus their work on discriminating between different animal vocalisations. Even though some of these sounds are very similar to each other, systems have been developed that give excellent classification accuracies. This is a promising result in the context of the ISRIE project as there will undoubtedly be occurrences of very similar sounds that require classification.

2.3.5 Other Applications

Automated classification of signals is not limited to the areas discussed above. Further areas of application include:

- Non-invasive inspection of tile-wall bonding integrity (Tong, Tso, and Xu, 2006);
- Audio fingerprinting for music and sound effect database searching (Haitsma, Kalker, and Oostveen, 2001; Haitsma and Kalker, 2002; Verfaille, Guastavino, and Traube, 2006);
- Melody classification with expert networks (Ray and Hsu, 1998); and
- Classification of Chilean wine (Beltrán, Duarte-Mermoud, Bustos, Salah, Loyola, Peña-Neira, and Jalocha, 2006).

This list is by no means exhaustive. However, it does illustrate that the field of signal classification is very diverse and encompasses many different areas of application.

2.4 Techniques Identified

The studies discussed in Section 2.2 use a number of different feature extraction and classifier techniques to accomplish their goals. Table 2.4 below provides an overview of the various studies and the techniques used, along with how accurately the overall system performed. A more detailed discussion of some of the techniques identified is given below.

Table 2.1: *Summary of prior research in the various fields of signal classification.*

Author(s)	Area of Application	Feature Extraction	Domain	Classifier	Best Result
Allegro et al. (2001)	Hearing aids, speech recognition	Amplitude variations, spectral profiles, harmonicity	t,f	HMM	90% (clean speech)
Bardeli et al. (2010)	Bird species monitoring	Spectral profiles	f	Autocorrelation	∼90%
Botteldooren et al. (2006)	Whole soundscape identification	Amplitude/pitch spectra	t,f	none	not reported
Chesmore (2001); Chesmore and Ohya (2004)	Bioacoustic, cricket species identification	Time-domain signal coding	t	MLP	100%
Couvreur et al. (1998)	Noise monitoring system	Framewise spectral analysis combined with MFCC	t-f	HMM	95%
Cowling and Sitte (2003)	Security system	Continuous wavelet transform	t-f	DTW	70%
		MFCC	pseudo-f	DTW	70%
Defréville et al. (2006)	Noise nuisance indicator	Extractor Discovery System w/ Timbre	t,f,t-f	GMM	96.60%

continued on next page...

Table 2.1

. . . continued from previous page

Author(s)	**Area of Application**	**Feature Extraction**	**Domain**	**Classifier**	**Best Result**
Dieckmann et al. (1997)	Security, speech recognition	Fourier transform	f	Vector quantisation (MELT algorithm)	99% (combined w/ video identification)
Dietrich et al. (2004)	Bioacoustic, cricket species identification	Energy & frequency contours, pulse features	t,f	fuzzy kNN	93.50%
Dimoulas et al. (2008)	Health, bowel sound analysis	Energy, spectral analysis, wavelet	t-f	MLP	95%
Dufournet et al. (1998)	Acoustic environment classifier	Third-octave bands, wavelet	f	MLP	80%
Gage et al. (2005)	Acoustic environment sensor	Short-time fourtier transform	t-f	Pattern matching	97.50%
Ghiurcau et al. (2012)	Acoustic detection of intruders in wild areas	TESPAR	t	Pattern matching	97%
		MFCC	pseudo-f	GMM	99%
Güler and Übeyli (2006)	Doppler ultrasound blood flow monitoring	Wavelet	t-f	Prob. NN	95-97%
He et al. (2007)	Vehicle health monitoring	Independent component analysis	t,f	manual	Not reported

continued on next page. . .

Table 2.1

. . . continued from previous page

Author(s)	**Area of Application**	**Feature Extraction**	**Domain**	**Classifier**	**Best Result**
Kim et al. (2004)	Video soundtrack content analysis	MFCC	pseudo-f	Cont. HMM	94.20%
Krijnders et al. (2010)	Environmental sound recognition	Cochleogram	t-f	Bayes classifier	44%
Lee et al. (2006)	Bioacoustic, cricket & frog species identification	MFCC with linear discriminant analysis	pseudo-f	Various	96-98%
Linder et al. (2008)	Health, dysphonia screening	Various time & frequency features	t,f	MLP	80%
Ma et al. (2006)	Acoustic environment classifier	MFCC	pseudo-f	Left-right HMM	93-96%
Mazarakis and Avaritsiotis (2007)	Vehicle classification	TESPAR	t	ANN	100%
Nooralahiyan et al. (1998)	Traffic level monitoring	Linear predictive coding	f	TDNN	86%
Norris and Denham (2003)	Sound texture detection	Self-organising map	t	none	not reported
Peltonen et al. (2002)	Hearing aid improvements	Various time & frequency features	t,f	1-NN	68.40%

continued on next page. . .

Table 2.1

... continued from previous page

Author(s)	**Area of Application**	**Feature Extraction**	**Domain**	**Classifier**	**Best Result**
Ruvolo et al. (2010)	Emotion analysis, cry detection, music vs. speech	STBF	pseudo-f, f	SVM	~78-98%
Schclar et al. (2010)	Acoustic-based vehicle detection	Wavelet packet	t-f	kNN	100% (vehicle vs. non-vehicle)
Selin et al. (2006)	Bioacoustic, bird sound identification	Wavelet	t-f	SOM / MLP	78% / 96%
Umapathy et al. (2007)	Hearing aid improvements	Linear discriminant basis w/ MFCC	t-f, pseudo-f	LDA	83-99%
Van Hirtum and Berckmans (2002)	Health, identification of respiratory diseases	Power spectral density	f	Fuzzy NN	96%
Wang et al. (2007)	Vehicle interior noise monitoring	Wavelet packet	t-f	MLP	not reported

2.4.1 Feature Extractors

A feature extractor in a classification system is concerned with reducing the amount of data the classifier has to process by finding the important features of the input data. Beltrán et al. (2006) explain how optimal classification is achieved when the classifier is kept simple, implying that the complexity of the input data to the classifier should be minimal. This view is also shared by Defréville et al. (2006).

Methods of feature extraction for audio signals are generally considered in groups related to the domain in which they operate. For example, spectral profiles are a frequency-domain (F-D) technique, periodicity can be determined in the time-domain (T-D) and Wavelet analysis allows access to both time-domain and frequency domain characteristics (TF-D). Table 2.2 lists some of the techniques that can be used for analysis in each of the domains. These techniques are discussed in detail below.

Table 2.2: *Feature extraction techniques available and their corresponding domains.*

Technique	**Domain**		
	t-d	**f-d**	**tf-d**
TDSC	●		
Fast Fourier		●	
Short–time Fourier			●
Wavelet			●
Wigner-Ville			●

Time-frequency analysis does have slightly more complex computation than just time- or frequency-domain. However, this complexity is sometimes out-weighed by the extra information provided. The type of sounds that will be present in an urban environment are unlikely to be stationary and there will also be a mix of transient and steady-state signals. Time-frequency techniques provide detailed information about how a signal changes over time with respect to frequency and can detect both transient and steady-state features within the signal.

Time-Domain Signal Coding

Time-Domain Signal Coding (TDSC) is a purely time-domain technique which is computationally inexpensive. It is based on a speech compression method known as Time-Encoded Speech (TES). TDSC and TES both exploit the fact that any band-limited signal can be described by its real and complex zeros (Chesmore, 2001).

They differ in that TDSC only uses the real zeros for analysis and has added functionality in the form of matrix normalisation and scaling, and automated codebook generation. TDSC is discussed in much greater detail in Chapter 3 as it was the main feature extraction technique experimented with, leading to novel implementations of TDSC.

Disrete and Fast Fourier Transforms

The Fourier transform is a well-established method of signal analysis and the most often used method for transforming a signal into the frequency domain (Marchant, 2003). Based on the theory developed by J. Fourier, it is used to break up a function into the frequencies from which it is made, much like a prism will break up light into separate colours (Hubbard, 1998). The original function is represented by an infinite sum of sine or cosine functions, each a multiple of the fundamental frequency (Marchant, 2003). Signals are usually discretised before applying a Fourier transform. Equation 2.1 defines the discrete Fourier transform (DFT) for a finite duration sequence *x(n)*, $0 \leq n \leq N$ - 1,

$$X(k) = \sum_{n=0}^{N-1} x(n)W^{nk} \tag{2.1}$$

where $W = e^{-j(\frac{2\pi}{N})}$ and N is the length of the signal. Given the sequence *X(n)*, $0 \leq k \leq N$ - 1, the inverse-DFT can be found using Equation 2.2.

$$x(n) = \sum_{n=0}^{N-1} X(k)W^{-nk} \tag{2.2}$$

(Gold and Morgan, 2000)

The standard computation of the DFT takes N^2 computations. However, the fast Fourier transform (FFT) reduces this number to $NlogN$ computations (Hubbard, 1998). While the FFT provides a powerful tool for frequency analysis of signals, it cannot describe what is happening to the signal over time. This is acceptable for a signal that does not change over time (a stationary signal) but that situation is unlikely in an urban setting. A more useful method perhaps is the short-time Fourier transform.

Short-Time Fourier Transform

The Short-Time Fourier transform (STFT) is the most widely used method for analysis of non-stationary signals (Cohen, 1995). It is calculated by performing a sliding window FFT on a signal (Marchant, 2003), hence the technique is sometimes known as the windowed Fourier transform. The signal is effectively sliced into small segments and then each of these has a FFT applied to it. This gives frequency information about the signal in each segment and a time-frequency picture of the whole signal can be built up from this. Equation 2.3 describes the STFT:

$$X_{STFT}(t,f) = \int x(\tau)w(\tau-t)^{*}e^{-j2\pi ft}\,d\tau \tag{2.3}$$

Where t is the point of calculation for the power spectrum, $w(\tau)$ is the sliding window with length τ, and $w(\tau-t)^{*}$ is the complex conjugate of $w(\tau-t)$ (Marchant, 2003). Although the STFT can provide a computationally efficient method for analysing a signal in both time and frequency, a compromise must be made. For small changes to be seen in a signal (i.e. high frequency information) a small window size must be used and little information is given about low frequency features. But if a larger window size is used high frequency information is lost. There will always be a trade-off between time localisation and frequency localisation. In an urban soundscape sounds will be made up from frequencies all across the spectrum so it is necessary to use a technique capable of handling a broad range.

Wigner-Ville Distribution

The Wigner-Ville Distribution (WVD - sometimes referred to as just Wigner Distribution) is an approach to time-frequency analysis that defines an "energy density" for a signal (Hubbard, 1998). It was developed by Wigner to calculate the quantum correction to the second virial coefficient of a gas to indicate how the gas deviates from the ideal gas law. For this a joint distribution of position and momentum was required. Wigner developed such a distribution and this was introduced into signal analysis by Ville (Cohen, 1995). The WVD can be expressed mathematically in terms of a time-domain signal, $x(t)$, as

$$W(t,\omega) = \frac{1}{2\pi}\int x^{*}(t-\frac{1}{2}\tau)x(t+\frac{1}{2}\tau)e^{-j\tau\omega}\,d\tau \tag{2.4}$$

(Cohen, 1995)

Although the WVD has excellent time-frequency domain properties, its application is limited by interference cross-terms (Mallat, 1999; Marchant, 2003). To overcome these it is necessary to average the Wigner-Ville transform and lose some time-frequency resolution. WVDs also suffer when a signal is subject to significant noise. The cross-terms can make the time-frequency distribution almost impossible to analyse (Marchant, 2003).

Wavelet Analysis

The Wavelet, or mathematical microscope as it is sometimes called, can give an approximate image of a signal but also zoom in on the smaller details (Hubbard, 1998). Its functionality can be related to that of the Fourier transform where a signal can be described by adding together different sines and cosines. In a wavelet transform the same operation can be performed by adding together wavelets of different sizes and positions (Hubbard, 1998). The *transform coefficients* describe how the *mother wavelet* should be manipulated. These wavelet *kernels* (the various time-shifted and dilated *mother wavelets*) are localised in both time and frequency unlike a Fourier transform where the sines and cosines are localised only in frequency (Marchant, 2003). The *continuous* wavelet transform is so called because the signal is studied at all possible resolutions using wavelets displaced by all values. This would give an infinite number of wavelet coefficients allowing perfect reconstruction of the signal. In reality this is not practical so the number of coefficients is usually in the order of 10,000 (Hubbard, 1998). The continuous wavelet transform introduces a lot of redundancy because the wavelets overlap each other and therefore some of the same information is contained in more than one wavelet. The *discrete* wavelet transform can describe a signal using a set of translated and scaled wavelets that are orthogonal to one another and therefore greatly reduces the redundancy (Marchant, 2003).

Wavelet transforms do suffer from some drawbacks. Similar to a windowed Fourier transform, at the start and end of a signal there may be unwanted artefacts appearing simply because the wavelet is being compared to “non-data”. Assumptions have to be made as to what the signal will do at its start and end. A wavelet transform will also suffer from the time vs. frequency resolution trade-off encountered with the STFT. However, the wavelet transform behaves differently in that it gives good time resolution and poor frequency resolution for high frequency components; low frequency components are well defined in terms of frequency resolution but poorly defined in terms of time (Marchant, 2003). This problem can be overcome using a wavelet packet decomposition[1].

2.4.2 Classifiers

In a classification system the classifier will take the output of the feature extractor and determine which category that set of data belongs to. Classifiers can broadly be split into two types; those that employ a neural network and those that do not. Determining if a technique lies in one category or the other can be decided from whether it uses any form of neuron model to process data (see Beale and Jackson (1998) for an introduction on neural computing). Figure 2.2 shows a basic McCulloch-Pitts neuron, one type of neuron model. From this diagram it can be seen that a neuron has a number of weighted inputs, a summation and a threshold value (μ_i). If the sum of the weighted inputs exceeds the threshold value then the neuron will generate an output signal. The MCP neuron model mimicks the behaviour of the neurones of the human cerebral cortex.

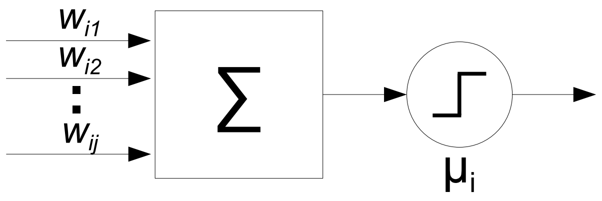


Figure 2.2: *A scematic diagram of a McCulloch-Pitts neuron model. The unit will generate an output only if the input weighted sum $\sum \omega_{ij} n_j$ is greater than or equal to the threshold μ_i for all inputs units (j). Adapted from Hertz et al. (1991).*

Neural network (NN) techniques can learn to adapt to a problem with minimal human input. The techniques that do not fall into the NN category often employ statistical analyis or pattern comparison techniques for classification.

Multilayer Perceptron

The Multilayer Perceptron (MLP) is a neural network consisting of more than one layer of neurons. This type of network is said to have one or more *hidden* layers. That is, these layers do not directly link to the outside world, they only accept an input and give an output. All neurons in one layer will be connected to all layers in

[1]For a discussion on the wavelet packet transform see Addison (2002), pages 133–140.

adjacent layers via *unidirectional* connections: links that (in a feed-forward network) can only transmit in the forward direction (Ham and Kostanic, 2001).

A number of studies have achieved successful pattern recognition rates using MLP networks with a feature extractor. Dimoulas, Kalliris, Papanikolaou, Petridis, and Kalampakas (2008) achieve results averaging 95% using a combination of time-domain and wavelet features input to an MLP with one hidden layer.

Self-Organising Map

The Self-Organising Map (SOM — sometimes referred to as a Kohonen Map) was developed by Tuevo Kohonen and is an unsupervised, competitive learning, clustering network (Ham and Kostanic, 2001). Whether it can be termed a neural network or not varies from one source in the literature to another. The main reason for this is that the units in a SOM do not behave like a perceptron would in an MLP network, for example. There is no threshold associated with each unit but they do have weighted inputs.

Learning Vector Quantisation

Vector quantisation is used for the compression of data by producing a *codebook* of quantisation vectors (at least one for each class of data) which provide a good approximation to the input vectors (Ham and Kostanic, 2001). Then any input vector can be encoded using the codebook. If both transmitter and receiver have the codebooks then only the codebook(s) for each input vector needs to be transmitted thus reducing network traffic. *Learning* Vector Quantisation (LVQ) is a supervised learning technique (also developed by Tuevo Kohonen) which builds on vector quantisation to classify input data (Hertz, Krogh, and Palmer, 1991). In training an LVQ network an input vector is given to the network along with its correct class. If the network classifies it correctly the weight vector associated with that class is updated according to one rule (which moves the vector towards the input vector). If the network misclassifies the update rule, it moves the weight vector away from the input vector. According to Kohonen (1990), this is known as LVQ1.

There are other training methods which can be used with LVQ networks which are explained by Kohonen (1990). The other training methods improve on the idea to include neighbourhood functions and move closer to Bayes decision theory (Hertz et al., 1991).

Time Warping

Time warping is a method of signal analysis often associated with speech processing and recognition tasks. *Linear* time warping (LTW) is used to allow comparison of one signal with another where the signals are of different lengths but the proportions are similar. According to Gold and Morgan (2000), if the proportions are different one of the signals will need to be warped dynamically to allow a comparison to be made. The technique then becomes known as *Dynamic* time warping (DTW). For example, imagine a car passing the point where a recording is made from. The features of the sound (car approaching, brief moment where the car is level and car moving away) are present in a recording of both the car travelling at 30mph and the car travelling at 60mph. If the approach rate and moving away rate were the same for both speeds LTW could be used for analysis. If however the car was accelerating as it approached and moved away LTW would not allow a comparison of the two signals but DTW would. In the application of ISRIE it is likely that DTW will be more useful than LTW as it is unlikely that two similar sounds will be directly proportional to one another.

2.5 Techniques for Investigation

The available literature relevant to signal classification is extensive. Many of the techniques identified in the above sections have been tried and tested in a number of different areas of application including the field of general audio event classification. The techniques that have been researched previously are mostly well-known techniques and are often found in signal processing texts dealing with signal classification as a whole.

To avoid repeating work that has already been done by numerous other researchers, the research detailed in the remaining chapters of this thesis are concerned with developing and applying a TDSC feature extractor to the task of classifying audio events into the categories identified in Section 1.5.

There are good reasons for choosing TDSC as a feature extractor for general audio event classification:

1. as has been identified already, TDSC is a computationally light-weight technique and would therefore be suited to use in a portable device running on batteries, for example, and
2. TDSC has not been applied in this context before and therefore this field of research would benefit from some experimental data which validates TDSC as a useful technique.

The first of these reasons is an important consideration for the final ISRIE system and has been a consideration of other projects developing remote sensors (see Ghiurcau et al. (2012)). The second reason highlights how the work detailed in this thesis will contribute to the field as a whole.

2.6 Chapter Summary

This chapter has given a review of research and work carried out previously into the field of sound identification and soundscape evaluation. The literature has not been limited to urban or general sound classification but has also included other areas of signal classification as inspiration for the ISRIE project. A discussion on some of the techniques for feature extraction and classifying has also been provided to give the reader some foundation on what is available in this field of research.

Time Domain Signal Coding has been identified as a prime candidate for developing a signal classification system, and this choice has been justified accordingly. In the next chapter TDSC is discussed in detail to explain its origins, how it has previously been used and implemented, and the initial experimental work and development carried out to modify TDSC for the task in hand.

Chapter 3

Adapting TDSC for Urban Audio Event Classification

In the previous chapter, it was discussed how time domain signal coding (TDSC) was a suitable candidate as the feature extractor for urban audio event classification. This chapter will expand on the history of TDSC, the processing required in TDSC analysis and how it was initially applied in this research for classifying urban audio events.

It will be shown in this chapter how the standard TDSC algorithm required significant modification for it to be appropriate for urban audio events. This chapter also details how a generic code book was developed.

3.1 Background and Applications of Time Domain Signal Coding

Time domain signal coding (TDSC) was introduced in the previous chapter as a purely time-domain technique based on a speech compression method known as Time Encoded Speech (TES). TES was proposed by R.A. King in the 1970s as a simple method of compressing speech data for digital transmission (King and Gosling, 1978). TESPAR (Time-Encoded Signal Processing and Recognition) is discussed by King and Phipps in their 1999 paper. TESPAR uses the same processes as TES to perform a zero-based analysis of signals.

King and Phipps (1999) propose how zero-based analysis can be used to classify a variety of signals. The authors use the widely-known Shannon sampling theorem and work by other authors to develop TESPAR. In their discussion, King and Phipps provide a history of zero-based signal analysis and highlight how it is a technique that has been available to mathematicians for a considerable time but has had relatively little application in the field of engineering. Correcting this disparity between the fields was started by Voelcker, who brought together the ideas and applied them in studying the modulation of signals, and continued by Requiche, who demonstrated the application of zero-based techniques to engineering problems (King and Phipps, 1999).

Zero-based analysis of signals has its origins in the amateur radio community around the same time that Shannon was developing his sampling theorem (circa 1949). Licklidder and Pollack experimented with removing amplitude information from waveforms leaving only the zero-crossing information. Licklidder and Pollack found that the intelligibility of a speech signal was maintained when the signal was rebuilt from only the zero-crossing information indicating that amplitude information was not necessarily required when transmitting speech data (King and Phipps, 1999). Later work by Bond and Cahn formalised the use and application of zero-based signal analysis, including the synthesis of a band-limited signal from a given set of zeros, and from this Voelcker and Requiche brought the technique into the engineering field.

King and Phipps explain how the extraction of zeros from a signal to allow the signal to be adequately represented is far from trivial. This is mainly because gathering complex zero information requires very involved calculations. Finding the real zeros of a signal, however, is very straightforward as these correspond to the time-domain waveform zero-crossings of the signal, as shown in Figure 3.1.

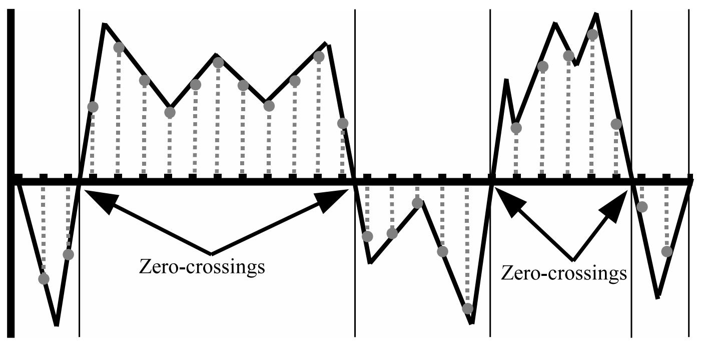


Figure 3.1: *The real zeros of a signal can easily be found from the time-domain signal.*

.

King and Phipps provide a comprehensive description of how TESPAR coding is carried out and this is summarised here. The portion of signal between a pair of successive zero-crossings is termed an *epoch*. Figure 3.1 shows the epoch divisions with vertical lines. King and Phipps explain that the perturbations of a signal are caused by the complex zeros. Therefore, to overcome the difficulty in finding the complex zeros of a signal, King and Phipps propose that the *shape* of the signal is extracted as a feature in TESPAR coding. Each epoch of the signal can be described by two properties: its duration, D, in samples and its shape, S. It is worth noting that King and Phipps are keen to highlight that not every complex zero can be represented through the shape of the signal, and the TESPAR coding technique only gives an approximation of the complex zeros.

Figure 3.2 gives an example of a signal that has had its TESPAR coding features highlighted. In this example, as with all TESPAR coding, the shape of an epoch is determined by the number of positive minima or negative maxima it contains. Samples in Figure 3.2 are represented by the vertical lines that terminate with a small circle. These allow the duration of each epoch to be easily seen. In this example, epoch 2 has 11 samples between its successive zero-crossings and 2 positive minima, D=11 S=2.

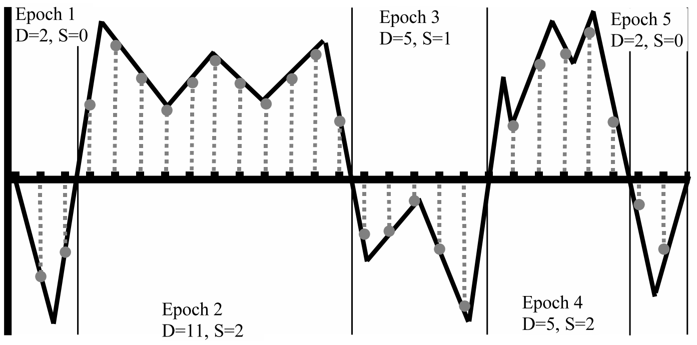


Figure 3.2: *A simple example of TESPAR/TDSC analysis.*

.

This coding process produces a D-S pairing for every epoch contained in a signal. Such an analysis would result in a very large amount of data for each signal. Therefore, King and Phipps (1999) propose the use of a vector quantisation process which results in an alphabet of symbols, each symbol representing one or more D-S pairing. A *codebook* is used to map the epoch D-S pairings on to the symbols contained in the codebook. Table 3.1 shows an example mapping of D-S pairings using a symbol set. In this example there are 8 codes which are shown in italicised font. This example shows how more than one D-S pairings can be mapped onto a single code. For example, code 5 has the D-S pairs 6-1, 7-1, 8-1, 9-1 and 10-1.

Table 3.1: *An example of a codebook that could be used in TESPAR coding.*

Duration (samples)	**Shape 0**	**1**	**2**	**3**	**4**	**5**
1	*1*	–	–	–	–	–
2	*2*	–	–	–	–	–
3–5	*3*	–	–	–	–	–
6–10	*4*	*5*	–	–	–	–
11–18	*6*	*7*	*8*	–	–	–

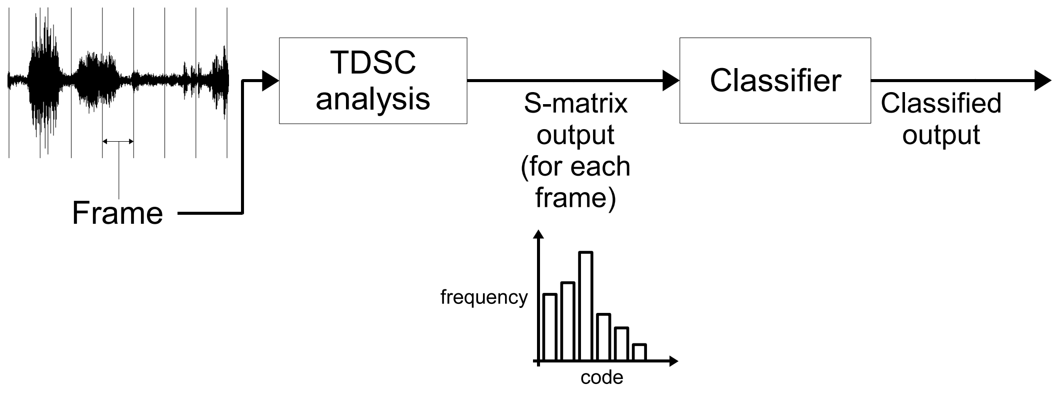
The example in Table 3.1 also shows that there are some D-S pairings which are not possible. As King and Phipps (1999) put it *"...short epochs cannot exhibit a multiplicity of minima..."* (p. 448). King and Phipps relate this statement to the fact that TESPAR was used for compressing and analysing band-limited signals and therefore multiple minima in short epochs would suggest there were frequency components present in the signal that fall outside of the band limits. It is also not possible for the shortest epochs to exhibit any minima.

At this juncture, the link between TESPAR and TDSC will be made. TDSC (the focus of this research) is a term coined by Dr. David Chesmore for an adaptation of TESPAR. TDSC uses the same extraction of D-S pairings as TESPAR discussed above but TDSC has been further developed to include techniques such as matrix normalisation, matrix scaling and automated codebook generation (Chesmore, 2001).

The output for both TESPAR and TDSC can be either a 1- or 2-dimensional histogram. The 1-dimensional histogram is referred to as the S–matrix and the 2-dimensional histogram is referred to as the A–matrix.

Figure 3.3 shows an example S–matrix and its associated codebook. In this example, the D-S pairings are 1-0, 2-0, 3-0, 3-1, 4-0, 4-1, 4-2, 5-0. Each element of the S–matrix provides the frequency of occurrence of each of the D-S pairings in the signal under analysis. The A–matrix describes how often an epoch with code i is followed by an epoch with code j by a lag L in the signal under analysis. Figure 3.4 shows an example A–matrix.

Once a signal is represented by either the S- or A–matrix it can be used as an input to one of the classifiers discussed in Chapter 2. The outputs of TDSC and TESPAR analysis are of a fixed size once the dimensions of the codebook have been decided. In the original band-limited applications of TDSC and TESPAR the number of codes required depended on the upper and lower frequencies of the signals



Code	D	S	Frequency
1	1	0	5
2	2	0	6
3	3	0	8
4	3	1	4
5	4	0	3
6	4	1	2
7	4	2	0
8	5	0	0

Figure 3.3: *An overview of how features are extracted using TDSC. The histogram shows an example of an S–matrix. The table shows the codes that would be used to generate the histogram.*

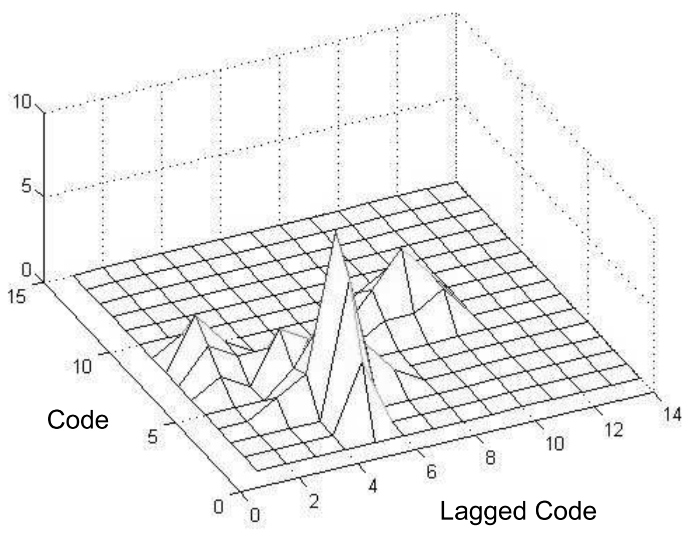


Figure 3.4: *An example A–matrix.*

being analysed. Therefore, the dimensions of the output of these analysis methods is dependent on the application. King and Phipps (1999) proposes a fixed size of 29 codes and Chesmore (2001) uses a codebook with a size of 28. Having a fixed size output for a feature extractor, such as TDSC and TESPAR, is common across most of the signal analysis and classification techniques discussed in Chapter 2. A fixed size output of the feature extractor ensures that extracted features from a variety of signals are directly comparable by a classifier.

3.1.1 Previous Applications of TESPAR and TDSC

It has already been discussed how TES was originally developed to allow the transmission of speech over a channel using a low data rate. King and Phipps (1999) took the initial TES concept and developed it further to include recognition and classification of signals, applying TESPAR to speech processing tasks.

Chesmore (2001) cites prior applications of TES in monitoring the condition of machinery using an acoustic signal and analysis of heart sounds to identify defects. The work detailed in Chesmore (2001) was discussed in Section 2.3.4: using TDSC to extract features from recordings of Orthoptera for classification by a multilayer perceptron neural network.

Farr (2007) extended the TDSC framework in his application which aimed to develop an automated bioacoustic detection and identification system for insects larva detection in wood. Farr argued that the need for a small codebook, as proposed by King and Phipps (1999), was no longer required for identification tasks because of the significant increase in computing power available to identification systems. Farr found that the codebooks and natural alphabets used by King and Phipps and Chesmore (2001) were too restrictive for his work and developed the *distributed code matrix* or "D–Matrix". This coding scheme represents every code that could be generated from a waveform and is therefore scalable according to the input signal. With this enhanced TDSC algorithm, the features being fed into LVQ neural networks, Farr achieved species identification rates in excess of 97%.

In an attempt to deliver a more computationally light-weight system, Mazarakis and Avaritsiotis (2007) applied the methods of TESPAR and TDSC in a wireless sensor network vehicle classification application. Mazarakis and Avaritsiotis used customised codebooks for the signals they aimed to classify, focusing on both the acoustic and seismic signals generated by vehicles. By customising the codebooks, the authors were able to minimise the number of symbols needed. Mazarakis and Avaritsiotis achieved classification results comparable with existing methods but highlight that the computational cost is greatly reduced.

Moca, Scheller, Mureşan, Daunderer, and Pipa (2009) implemented the TESPAR feature extraction algorithms for their investigation into automated depth of anesthesia (DOA) estimation based on electroencephalogram (EEG) recordings. Moca et al. used A–Matrices derived from the EEG signals to train multilayer perceptrons to map the EEG signals to different DOA classes. The classification results of the artificial system were compared with classifications made by anesthesiologists (human experts). An example of the S– and A–Matrices used by Moca et al. is shown in Figure 3.5. There is a clear distinction between the S– and A–Matrices for the two different states of anesthesia shown in the figure. Moca et al. found a close match between the classifications made by the system they had developed and the human expert classifications (a 2% difference is reported).

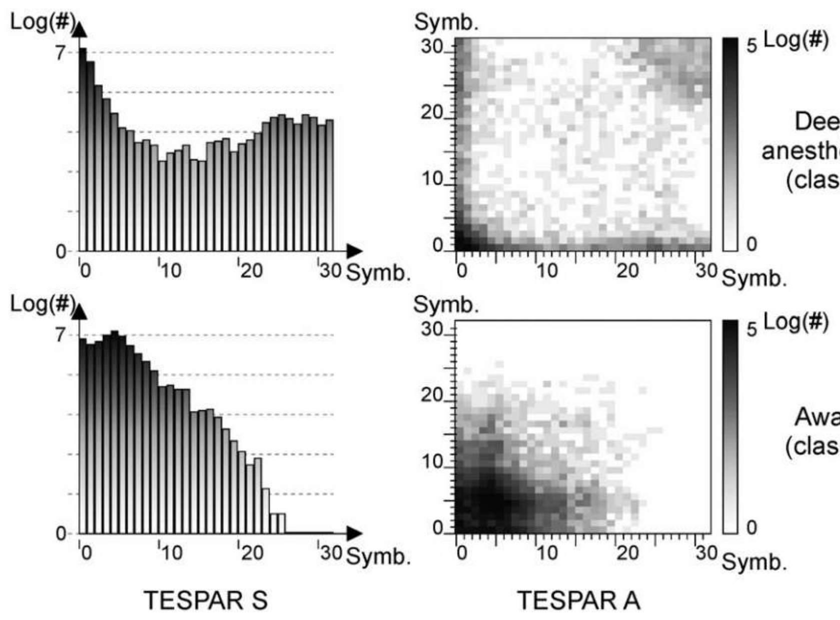


Figure 3.5: *Examples of the S- and A–matrices used by Moca et al. (2009).*

The prior applications of TESPAR and TDSC discussed here (and earlier in the thesis) show that these time-domain methods are valid tools for analysing a variety of signals and have been applied in many fields. Many of the authors (Chesmore, 2001; Chesmore and Ohya, 2004; Farr, 2007; Mazarakis and Avaritsiotis, 2007; Ghiurcau et al., 2012) emphasise how computationally inexpensive TESPAR and TDSC are. Whilst this may not be of benefit in a system implemented on a modern desktop or laptop computer, it can certainly benefit a portable module designed to spend significant amounts of time out in the field, possibly running from a battery power source. Chapter 1 introduced the ISRIE system and explained that the eventual aim for the project was to have a self-powered module capable of performing identification tasks and communicating wirelessly with other devices. This is similar to the aims of Mazarakis and Avaritsiotis (2007) who state that the low computational cost of time-domain feature extraction techniques are of clear benefit.

3.2 Implementing TDSC using MATLAB®

3.2.1 Mathworks MATLAB®

All development work discussed in this thesis was undertaken using MATLAB®[1]. This software allows for quick development of algorithms without the need of compilation at run time (unlike languages such as C, C++, Java, etc.). So if small changes are made to a function it is not necessary to recompile a whole function which could be a time consuming process. Programs and functions written in the conventional languages are often able to process data more quickly than a higher-level language such as MATLAB® but for the purposes of development this would be an unnecessary advantage.

MATLAB® contains many powerful functions built into it for the purposes of data manipulation and signal processing, and these can be added to through the use of toolboxes. However, it is necessary to write functions for a particular purpose which MATLAB® will not cover. Functions are straightforward to write in MATLAB® if the purpose of the function is known, and decisions simply need to be made on the inputs and outputs. The MATLAB® Neural Network Toolbox was used extensively for designing, training and testing a variety of classifiers.

The following section provides an overview of the process involved in designing and writing the functions created for the presented research. All of the functions are presented in full in Appendix C.

3.2.2 Designing and Implementing MATLAB® Functions

The process of designing and implementing a piece of code to perform a function in any programming language often differs from one person to the next. This section will describe the skeleton process used to write MATLAB® functions for the presented research. MATLAB® functions can be written and tested in a relatively short amount of time because there is no need to compile the functions. Consequently, the

[1]Mathworks MATLAB® version 7.3.0.267 (R2006b)

functions presented in Appendix C are the final versions arrived at after a number of iterations. The iterations will have come about as a result of immediate testing and the realisation that additional functionality was required.

To illustrate the process of design and implementation, the `multiSMatReturn.m`[2] function is used as an example, taken from conception to the final code listing.

Initial conception of the function

The function `sMatReturn.m` analyses individual WAV files to return an S–matrix for each file. Given that there were over 60 files used for training networks, analysing each WAV file individually to generate the training data would have been a laborious and time-consuming process. Therefore, it was decided that a function should be written that could accept a path to a folder on a computer hard disk and analyse all of the WAV files located in that folder.

Identifying the purpose of the `multiSMatReturn.m` function allows some initial requirements to be determined: the function needs a path as an input, the function should return S–matrices for all of the WAV files as its output, and at some point in the function `sMatReturn.m` needs to be called. Therefore, the input to `multiSMatReturn.m` will be a path to the WAV files and the output will be a collection of S–matrices.

Having determined the purpose of the function and what it will take as its input and output, how the function will achieve its purpose can be considered.

Function flow diagram and code specifics

A flow diagram for a function gives an overview of the individual processes that need to take place for the function to achieve its goal. The flow diagram devised for `multiSMatReturn.m` is presented in Figure 3.6. Each stage of the flow diagram is discussed in detail below.

[2]All developed functions used a naming convention similar to that used in the Java programming language: the first letter of a function is lowercase and subsequent whole or part–words are capitalised. This allowed the functions written for the purposes of this research to be differentiated from the functions included with MATLAB®

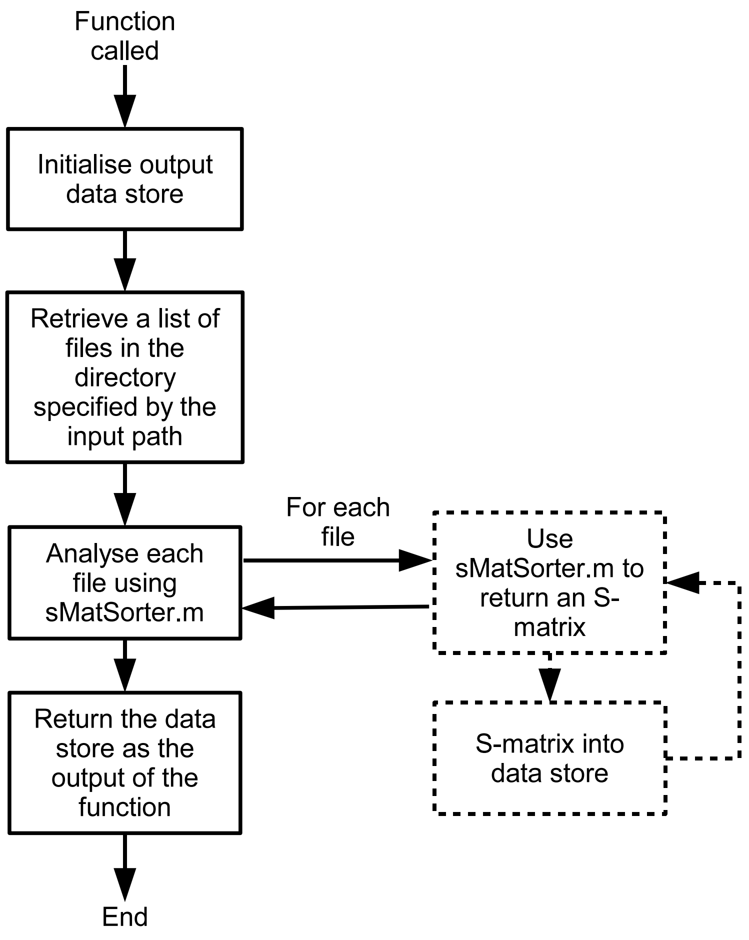

Function called
Initialise output data store
Retrieve a list of files in the directory specified by the input path
Analyse each file using sMatSorter.m
For each file
Use sMatSorter.m to return an S-matrix
S-matrix into data store
Return the data store as the output of the function
End

Figure 3.6: *The function flow diagram devised for `multiSMatReturn.m`.*

Function declaration All MATLAB® functions must start with a line of code similar to that shown below:

```
function SmatData = multiSMatReturn(path, framelength)
```

The first word of this line informs MATLAB® that this is a function (as opposed to a script). The second word defines what the output of the function is. At some point in the function a variable must be declared with this name and it must be set to a value. After the equals sign the name of the function is given (this must be the same as the name of the file where the function is saved) along with the inputs to the function in parentheses.

Initialise output data store There are many options in the MATLAB® environment for storing data. One of these options is a structure array which allows specific fields to be defined when creating the array. A key benefit of using a structure array is that any type of variable can be stored in a structure array (e.g. integers, floating point numbers, vectors, matrices, other arrays, etc.). For this particular function, it was desirable to store the name of each file being analysed together with the S–matrix data for that filename. The initialisation code for the structure array is as follows:

```
1 % Create the output Struct with 2 fields; "filelist" - a list
2 % of the filenames, and "sMat" - the S-matrix.
3 SmatData = struct('filename',[],'sMat',[]);
```

Lines one and two of this code begin with an ampersand which denotes a comment. When a function is used, any lines that begin with an ampersand are ignored. It is good practice to include commentary in code to provide guidance for future users and adapters of the code. The above code snippet also shows the naming convention used for structure arrays in the presented research. Most variable and function names begin with a lower case letter but structure arrays always started with a capital letter to differentiate them from other variables. The code shows the fields for each item in the array: the first is a string that simply says *filename* to indicate that the second field contains the filename of the WAV file whose S–matrix is stored in the fourth field.

Retrieve a list of files in the directory specified by the input path Functions built-in to MATLAB® allow easy retrieval of a list of files in a specified directory. After retrieving a list of the files, it was found to be necessary to remove the first two entries of the list as these entries are references to parent directories and not names of files to be analysed.

```
1 % File handling
2 filesTemp = dir(path);
3 % The first 2 of these are "." and ".." respectively. I.e. not
4 % wav files. Remove.
5 files(1:length(filesTemp)-2,1) = filesTemp(3:length(filesTemp),1);
6 % Matrix to store filenames
7 filenames = [];
```

Lines two and six actually perform the file handling, storing the list of files in a matrix called *files*. Line 7 creates a matrix in which to store the filenames. This matrix was used for development purposes only.

Analyse each file using `sMatSorter.m` To analyse all of the files in the specified directory, the function is made to repeat a section of code by using a `for` loop.

```
1  for i = 1:length(files)
2  ...
3  end
```

Line one of the above code shows the start of the `for` loop. A `for` loop repeats the code contained within it as many times as specified. In this case, the loop is repeated as many times as the number of files listed in the variable *files*. Another feature of the way the loop is initiated is the use of index variable i which will increment from a value of 1 to the total number of files listed in the variable *files*. The variable i can be used throughout the loop to address the i^{th} item in a matrix, for example. The number of files is obtained by using the MATLAB® function `length` on the variable *files*.

Use `sMatSorter.m` to return an S–matrix Prior to calling the function `sMatSorter.m`, a number of checks are performed on each file to be analysed. These are illustrated in the following code:

```
1  % Check that the file is a file and not a directory
2     if files(i,1).isdir == 0
3         % Generate string for wav file to analyse
4         filename = strcat(path,files(i,1).name);
5         % Check that this file is in fact a wav file
6         [p,n,e,v]=fileparts(filename);
7         if strcmp(e,'.wav')
8             % Perform TDSC analysis using sMatReturn
9             datasmat = sMatReturn(filename,framelength);
10        ...
11        end
12    ...
13    end
```

Line two of the code performs a check to see if the filename located in position i of the *files* matrix is a file and not a directory. The code contained within the `if` statement (lines 3 to 12) will only run if the filename is not a directory. Otherwise, the function skips straight to line 13.

Line four uses a MATLAB® function which concatenates the string variables given in the parentheses into a single string. In this case, the path of the files being analysed is concatenated with the name of the file located at position i in the *files* matrix. This step is necessary because `sMatReturn.m` requires the full address of a WAV file to analyse it.

Lines six and seven are used to check that the i^{th} filename in the matrix *files* is the name of a WAV file. Passing a non-WAV file to `sMatReturn.m` would cause an error to occur. If the file is not a WAV file lines 8 to 10 of the code will be skipped over.

Line nine of the above code calls the `sMatReturn.m` function, passing over two variables - the full location and name of the file to be analysed (generated from within this function), and the framelength to be used during TDSC analysis. At this juncture, a correction is needed to the input requirements of the current function being written. It was stated earlier that the input to the function `multiSMatRetun.m` was a string containing the path of the files to be analysed. However, if `sMatReturn.m` requires a framelength variable and is called from within `multiSMatRetun.m`, a framelength variable must be specified as an input to `multiSMatRetun.m`. The S–matrix returned by `sMatReturn.m` is stored in the variable *datasmat*.

S–matrix into data store The S–matrix returned from `sMatReturn.m` for the i^{th} filename in the variable *files* is now stored in the structure array. The S–matrix is inserted into the structure array at the i^{th} position along with the filename of the corresponding WAV file.

```matlab
% Add the S—matrix to the output data structure
SmatData(i).sMat = datasmat;
% Add the filename to the output data structure
SmatData(i).filename = n;
% Add the filename to a matrix of filenames (for test purposes)
filenames(length(filenames)+1:length(filenames)+length(n)) = n;
```

Return the data store as the output of the function After all of the WAV files contained in the variable *files* have been analysed, the structure array *SmatData* will contain S–matrices for each file. This structure array is the output of the function. There is no need to write any code to inform MATLAB® that this is the returned data as it was specified in the function declaration discussed earlier.

Writing functions summary

In this section there has been a discussion of the procedure followed during the presented research for designing and writing MATLAB® functions. All of the functions presented in Appendix C were designed in this manner. It has been mentioned that most functions will have been through a number of iterations before arriving at the presented code. The reasons for having iterations of code are numerous - in the case of `multiSMatReturn.m` if the inputs to `sMatReturn.m` were altered for whatever reason, where this function is called from within `multiSMatReturn.m` a further alteration would be required.

It is hoped that this overview of designing and implementing a function will allow the reader to gather an insight into one possible method of writing MATLAB® functions if they are unfamiliar with the program. The commentary given in the code listings should also assist the reader in following how the functions are laid out and what the purpose is of each line of code.

3.2.3 Basic TDSC Feature Extraction

Figure 3.7 gives an overview of how TDSC feature extraction is performed. A signal is presented as a whole to the TDSC analysis system. The signal is split into frames of the same length and each frame is analysed to extract features, as described in Section 3.1. It is often helpful to consider each frame of the whole signal as the waveform shown in Figure 3.2.

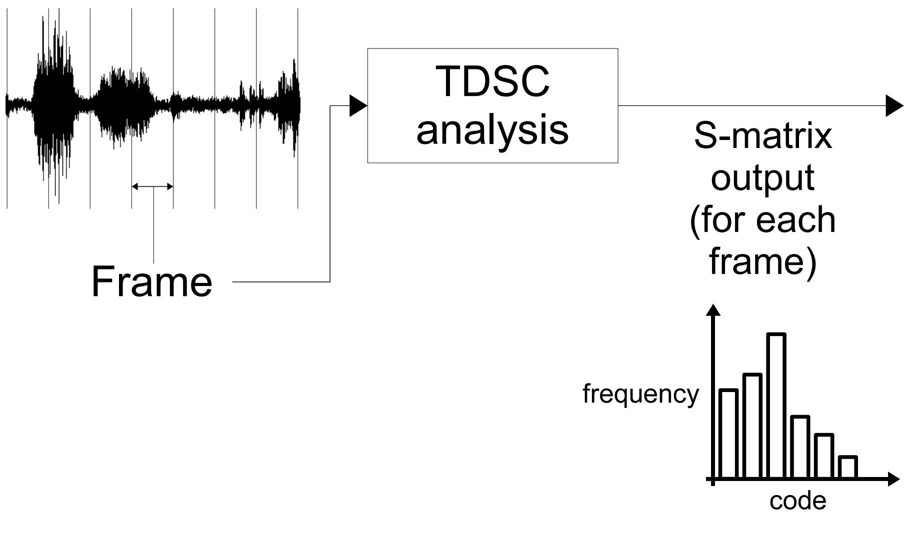


Figure 3.7: *S–matrix generation using TDSC.*

In a basic implementation of TDSC, each frame will have an S–matrix produced describing the frequency of occurrence of each D-S pair in the predetermined codebook. The data collected for each frame can be used to examine the time-domain features of the signal as time varies, or they can be summed together to represent the signal as a whole. The S–matrix is developed according to Equation 3.1.

$$s(i) = \sum_{j=1}^{N} x(j) \; 1 \leq i \leq M \tag{3.1}$$

where: $s(i)$ = element i of matrix s; N = number of epochs in the signal; M = number of codes in the codebook; $x(j) = 1$ if $x(j) = j$, 0 otherwise (Chesmore and Ohya, 2004).

The MATLAB® function developed for the purpose of controlling and performing TDSC analysis is called `sMatReturn.m`. Figure 3.8 shows the functions which are invoked for the purpose of extracting S–matrix data from a signal. The function `sMatReturn.m` takes as its inputs a path for the location of a signal to be analysed (normally this will be a WAV[3] file) and the framelength to be used during TDSC feature extraction. The function returns a M×N matrix of D-S pair frequencies; where M is dependent on the number of D-S pairs in the codebook, and N is the number of frames in the signal. The following section discusses development of the codebook used in TDSC analysis.

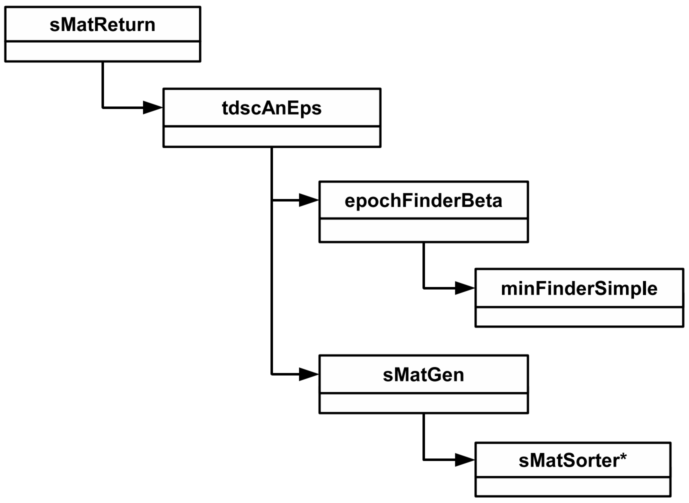


Figure 3.8: *Function dependencies for basic TDSC analysis. The function `sMatSorter` is adorned with an asterisk to denote that it can be one of three different functions which each have "sMatSorter" in their name. These functions are discussed throughout this chapter.*

[3]Waveform Audio.

3.3 Classification Methodology

Figure 3.9 shows an overview of how a network was trained and tested with features extracted using TDSC. The first box of the diagram in Figure 3.9 is representative of the TDSC feature extraction shown in Figure 3.7 and described in Section 3.2.3.

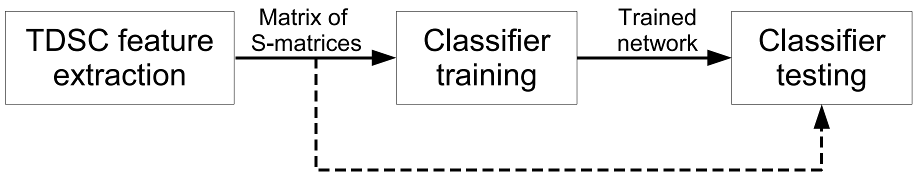


Figure 3.9: *Flow diagram of training and testing a network*

The output of the TDSC feature extractor is a matrix of S–matrices, each S–matrix representing one frame of audio data. The function `sMatReturn.m` will return a matrix of S–matrices for a single WAV file. To simplify the process of analysing multiple WAV files, the function `returnSMatrices.m` was developed to accept as an input a path for a folder of WAV files. Each of the WAV files are analysed sequentially by `returnSMatrices.m` and a matrix of S–matrices for all of the WAV files is returned.

As well as returning a matrix of S–matrices, `returnSMatrices.m` also returns a vector containing a target for each S–matrix. The target is a number representing which output of a classifier the corresponding S–matrix should be associated with. For example, if the S–matrix located at position 30 belongs to an audio sample from the *building* category and the *building* is to be associated with output 5 of a classifier, then the target at position 30 will be a 5.

A target vector is required for networks that employ supervised training such as Learning Vector Quantisation (LVQ) and Multilayer Perceptron (MLP) networks. The target vector allows the training algorithm for such a network to adjust the weight vectors of the internal network connections. A simplified explanation of training for an LVQ network is as follows:

- Present the network with an input dataset (one S–matrix) and a target vector. The target vector will contain all zeros except for a 1 at the location of the winning output (e.g. a target vector for a 6-output network with a target winning unit of 5 would be [0 0 0 0 1 0]).
- Allow the network to classify the input dataset. If the winning output unit corresponds to the target vector (e.g. output unit 5 is the winner and the target

vector specifies that the output should be unit 5) then the weight vectors of the network are adjusted to move the input dataset (and similar datasets) towards unit 5. However, if the winning unit does not correspond to the target vector (e.g. unit 3 is the winning unit) the weight vectors are adjusted to move the input dataset (and datasets that are similar) away from unit 3.

- This process is repeated for all input datasets.
- One training cycle of the network is completed once all input datasets have been passed through the network and weight adjustments have been made. A training cycle is termed a *training epoch*, which should not be confused with an epoch as used in TDSC feature extraction.
- A network can repeat the training process for a specified number of training epochs or until a training target is reached.

The training process described above is very similar for MLP networks and other types of network that require supervised training. Detailed explanations of training methods for various networks can be found in Abdi (1994), Hertz et al. (1991) and Kohonen (1990).

After training of a network was complete, the network could then be tested using S–matrix data extracted from the test audio data. These features were extracted using `multiSMatReturn.m`, a function that will return S–matrices for all of the WAV files located in a given folder. These S–matrices are not returned with a target vector because a target is not required for *simulation* of a network. MATLAB® uses the term *simulation* to mean presenting a network with some data for the network to classify. Network weights are not adjusted during simulation of a network.

The output of a simulation of a network with N outputs for a single dataset is a N-by-1 vector. The values of the network outputs are presented in this vector. For each test WAV file there were 15 S–matrices representing 15 frames of audio data (3 seconds / 0.2 seconds per frame = 15 frames). Therefore, the output of the simulations for each WAV file returned an N-by-15 matrix containing network output values for each frame of TDSC data. From this matrix, the mean values for network outputs was calculated to give an N-by-1 vector.

Throughout the research detailed in this thesis, the approach adopted for classifying audio data was similar to the approaches seen in the literature. From a set of audio samples the majority are used for training a network and the remaining few are used to test the network. Cowling and Sitte (2003) use a *jack–knife* method in which all sound samples are used to train a network except for one sample which is used to test the network. Similarly, Umapathy et al. (2007) use the *leave–one–out* training and testing method which is similar to the method used by Cowling and Sitte.

A final point worthy of discussion in relation to the classification methodology is the number of hidden units used in each network. There is little guidance in the literature as to how many hidden units a network should have in relation to the number of inputs and outputs. Hawickhorst, Zahorian, and Rajagopal (1995) recommends that the number of hidden units in a network is less than the number of inputs to the network. This recommendation was followed in the experimental work detailed in this thesis.

3.3.1 Audio Data

Details of the audio files used for training and testing can be found in Appendix A. In total, 82 samples of urban audio events were used for training and testing. Of these, 62 were used for training and the remainder were used for testing. All audio samples were encoded at a sampling rate of 44.1 kHz and a bit depth of 16.

Each WAV file used for training and testing was 3 seconds in length. This length of sample was arrived at after consulting the literature and performing some brief initial tests. Many of the previous studies discussed in Chapter 2 do not provide any details of the length of the audio samples used. Of those that do provide details, Defréville et al. (2006) used a sample length of 0.5 seconds, Umapathy et al. (2007) used 5 seconds, and Norris and Denham (2003) used 6 seconds. No reasoning was given in these earlier works as to why these sample lengths were used.

3.4 Early Codebook Development

In the previous applications of TDSC, the codebook was tailored to suit the signals being analysed. The size of the codebooks were relatively small ($\sim$30 codes) because the range of signals being analysed did not require anything larger. That is, the signals under analysis were bandlimited and could therefore be described by a small number of codes.

King and Phipps (1999) recommend the use of 29 codes, this being a sufficient number of codes for any bandlimited signal. Adjusting the frequency band capable of being represented by a TESPAR/TDSC algorithm is achieved by changing the rate at which the signal is sampled (Farr, 2007). By examining the codebook used by Chesmore (2001) to identify different species of Orthoptera (presented in Table 3.2) this aspect of TESPAR/TDSC can be examined more closely.

Table 3.2: *The codebook used by Chesmore (2001) in his work identifying different species of Orthoptera.*

Duration	**Shape**					
(samples)	**0**	**1**	**2**	**3**	**4**	**5**
1	*1*	–	–	–	–	–
2	*2*	–	–	–	–	–
3	*3*	–	–	–	–	–
4	*4*	–	–	–	–	–
5	*5*	–	–	–	–	–
6–7	*6*	–	–	–	–	–
8–10	*7*	*8*	–	–	–	–
11–13	*9*	*10*	–	–	–	–
14–18	*11*	*12*	*13*	–	–	–
19–23	*14*	*15*	*16*	*17*	–	–
24–30	*18*	*19*	*20*	*21*	*22*	–
31–33	*23*	*24*	*25*	*26*	*27*	*28*

The codebook illustrated in Table 3.2 allows a minimum and maximum epoch duration of 1 and 33 samples respectively. The duration of an epoch in seconds ($\mathrm{D}_{seconds}$) can be found by dividing the duration in samples ($\mathrm{D}_{samples}$) by the sampling frequency (f_s):

$$\mathrm{D}_{seconds} = \frac{\mathrm{D}_{samples}}{f_s} \tag{3.2}$$

Chesmore (2001) used $f_s = 44.1$ kHz. Using Eq. 3.2 gives a possible $\mathrm{D}_{seconds}$ range of 2.3×10^{-5} s to 7.5×10^{-4} s.

Finding the minimum and maximum frequencies that can be represented by this codebook can be accomplished using Eq. 3.3. This is a standard formula for finding the frequency, f, of a wave given its period, T, where T is equal to $2(D_{seconds})$.

$$f = \frac{1}{T}\mathrm{Hz} = \frac{1}{2 \times D_{seconds}}\mathrm{Hz} \tag{3.3}$$

In the present example, Eq. 3.3 gives a frequency range of 666 Hz to 22 kHz for the codebook illustrated in Table 3.2. This frequency range is acceptable for the application of TDSC by Chesmore (2001).

In using the techniques of TESPAR/TDSC in the research detailed in this thesis, a broader range of duration and shape pairs was required because a much broader range of sounds and more complex sounds exist in an urban environment. This was verified through analysis of sample audio signals from the various categories shown in Table 1.1 and is discussed further in Section 3.7.

During the early stages of TDSC development, the use of different codebooks for each category of sound was considered. These codebooks would be tailored to the category, only using the D-S pairs found in audio files of that category. However, this was discounted as an option for two reasons:

- even within a single category there are considerable variations in the various signals that could be present – recall the *consistency* issue highlighted by Defréville et al. (2006) in Chapter 2; and
- having different codebooks for each category of different sizes introduces unnecessary complexity at the classifier stage because the classifier will need to be able to accept any size of codebook and make comparisons between these.

Consequently, a generic codebook was sought which could be used to represent a signal from any of the sound categories given in Table 1.1. The first stage in developing a generic codebook was to determine the maximum epoch duration and shape values that would be needed to adequately represent the sound categories.

3.4.1 Duration and Shape Pair Distribution

In developing a generic codebook capable of being used with all of the sound categories, it was important to first establish the distribution of D-S pairings for signals from the different categories. Analysing the distributions reveals the most significant D-S pairings that should be included in a generic codebook. Mazarakis and Avaritsiotis (2007) performed a similar procedure in their work to develop codebooks suitable for the signals they were analysing.

Equation 3.4, adapted from Mazarakis and Avaritsiotis (2007), shows how the distributions were constructed:

$$L_{ij} = \sum_{n=1}^{N} x(D_n, S_n) \tag{3.4}$$

with

$x(D_n, S_n) = 1$ if $i = D_n$ and $j = S_n$

$x(D_n, S_n) = 0$ otherwise

where $L_{ij} = (i, j+1)$ element of the D-S distribution ($j+1$ is used because a shape of 0 cannot be used as an index), $N =$ the total number of epochs in the signal(s) being analysed, n is the nth epoch of the signal, D_n is the duration of the D-S pairing ij of the nth epoch, and S_n is the shape of the D-S pairing ij of the nth epoch. Once the D-S distribution matrix had been completed, it was normalised to the highest value of L_{ij}.

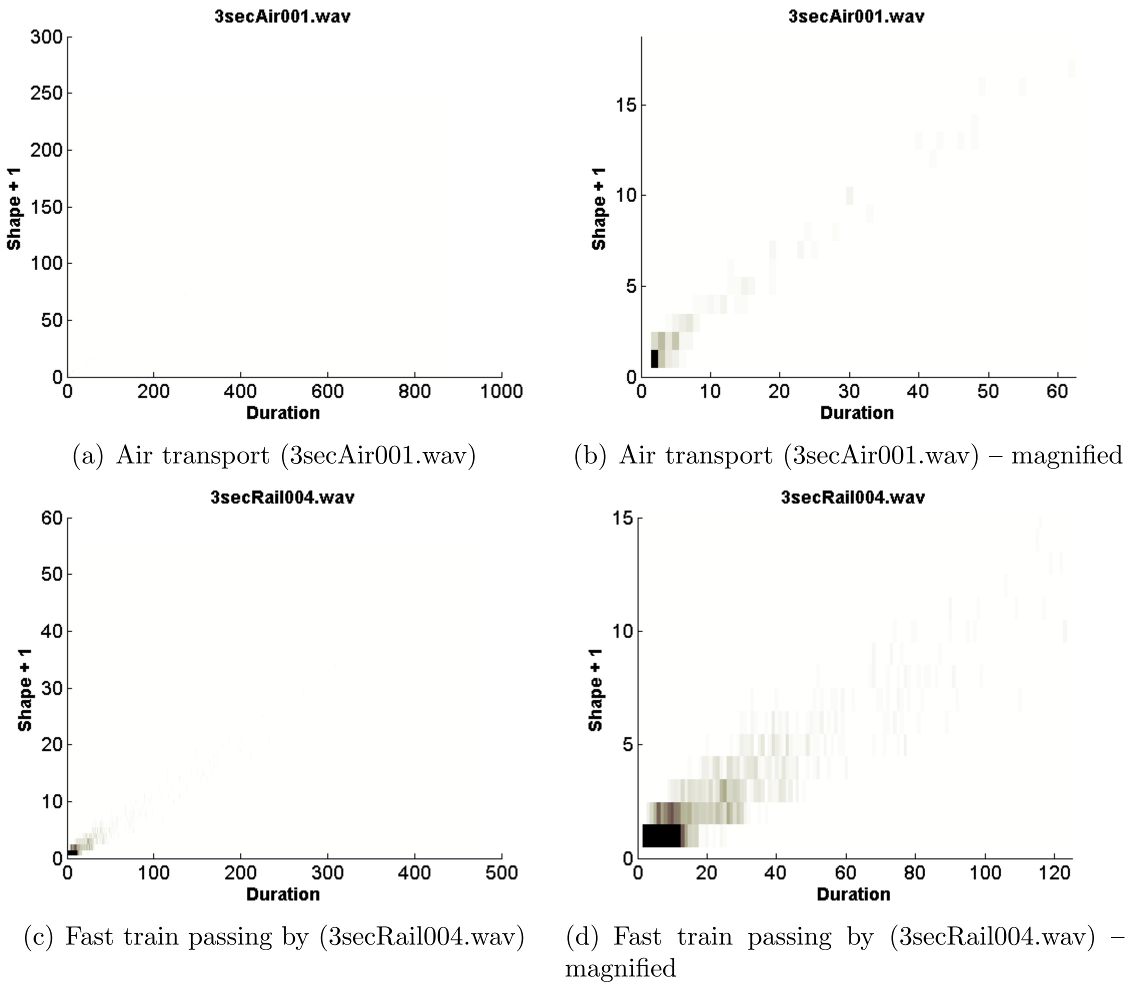
Figure 3.10 shows a selection of D-S distribution plots for a variety of sounds. The D-S distributions are plotted using the MATLAB® `image` function with a 1−`bone` colourmap. The `image` function plots the data contained in a matrix directly. Therefore, the darker areas show a higher concentration of that D-S pairing being present in the signal.

Figure 3.10(a) does not appear to show any dark areas, which would indicate that there are few repetitions of epochs with identical D-S characteristics. However, when the plot is magnified close to the origin (Figure 3.10(b)), it is possible to see that the D-S pairings with the highest frequency of occurrence are very close to the origin. The reason for the plot in Figure 3.10(a) being plotted with a maximum duration of ∼1000 and a maximum shape of ∼300 is that the signal may have had a single epoch with either or both of these values as one of its D-S pairings. The plots of `3secRail004.wav` in Figures 3.10(c) and 3.10(d) also exhibit this property of the D-S distribution for that signal.

Figure 3.11(a) shows a D-S distribution plot for all of the WAV files labeled as training files[4], with Figure 3.11(b) showing a magnified portion of the distribution close to the origin.

The D-S distribution data matrix used to generate the plot shown in Figure 3.11(a) had a maximum duration value of 5236 and a maximum shape value of 981. If all possible D-S pairs were used in a codebook using these figures, the codebook would have ∼4.5 million codes. Clearly this would not have been an appropriate size for

[4]see Appendix A for a summary of the audio files used for experimentation.



(a) Air transport (3secAir001.wav)

(b) Air transport (3secAir001.wav) – magnified

(c) Fast train passing by (3secRail004.wav)

(d) Fast train passing by (3secRail004.wav) – magnified

Figure 3.10: *D-S distribution plots for example signals from various sound categories.*

an input to a classifier, particularly when recalling that a feature extractor should help to minimise the complexity of a classifier (see Section 1.4).

It was therefore decided to reduce the maximum duration, D_{max},to 1000 and the maximum shape, S_{max}, to 75. Also, rather than use every duration value, a set of duration ranges were devised (see Appendix B for a table of the ranges used). With these reductions to the possible codes, the output of TDSC analysis had only 1716 codes.

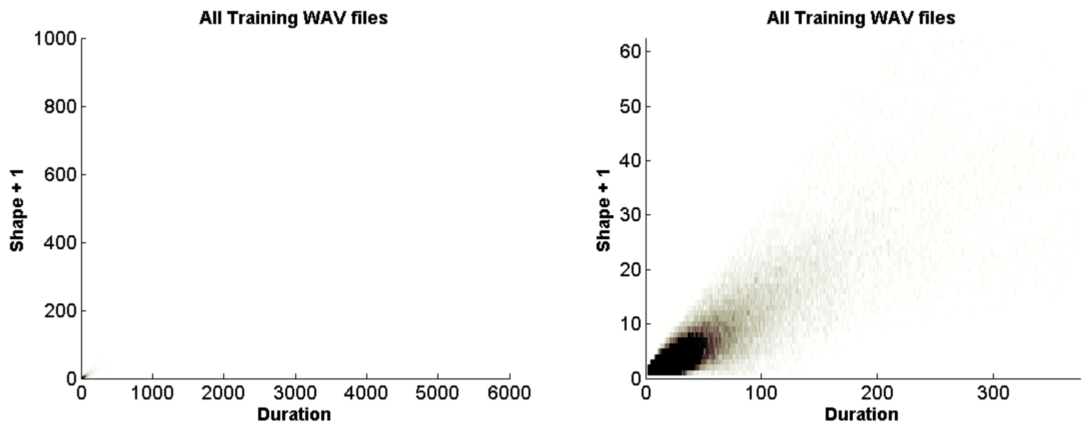


Figure 3.11: *D-S distribution plots for all training WAV files – normal and magnified.*

3.5 Classifying S–matrices with D$_{max}$=1000 and S$_{max}$=75

3.5.1 Classification Using Self–Organising Map Networks

During the initial stages of the presented research, it was necessary to determine the viability of the generic codebook described in Section 3.4. In the first instance, a Self–Organising Map (SOM) network was selected for training with TDSC features.

Self–organising maps were introduced in Chapter 2 as an unsupervised classifier. Previous studies that made use of SOMs were also introduced in Chapter 2. In particular, the work of Selin et al. (2006) and Norris and Denham (2003) showed that SOMs can produce good classification results. These prior results and the ease of implementing SOM networks provided good reason for using SOMs in the first instance.

A SOM network shares many similarities with other classifier networks in that it has input and output units, and weight vectors associated with these. The difference between a SOM and other types of network (for example, a multi-layer perceptron) is that SOMs are unsupervised during network training, making their own decisions as to how input patterns become associated with output units. This is in contrast to a supervised network where the user informs the network of which input patterns should be associated with which output units.

The experimental work during the early stages of the presented research is discussed in detail in the conference proceedings included in Appendix G. A summary of that work is included here.

Two types of audio signal were tested using TDSC features and SOM networks; general audio recordings and field recordings of *Tibicen* cicada. The aim of the experimental work was to see how well SOM networks could form associations between the different categories of sound and the outputs of the networks.

It was found that for the cicada recordings, a SOM network was easily able to form clear associations between different species of cicada and groups of network outputs. For example, with a 10–output SOM network, outputs 1 to 3 were primarily associated with *T. flamatus*, outputs 4 to 6 with *T. japonicus*, and outputs 8 to 10 with *T. bihamatus*. It would have then been possible to apply a simple decision rule to the output of the SOM to form a final classification for the cicada recordings. Overall classification accuracy using cicada recordings was 95%.

Classification results using general audio sounds were not as promising during the early stages of the presented research. It was found that SOM networks were unable to form consistent associations between signals from each sound category and specific output units or ranges of output units. Unlike the cicadas, it would not have been possible to use simple decision rules on the outputs of a SOM network. Using visual analysis of SOM output plots over time, classification accuracies were achieved only as high as 30 to 40%.

After the low classification accuracies were encountered using the SOM networks, it was decided to try the same classification task using a network having supervised training. Learning Vector Quantisation (LVQ) networks were selected for this task as a logical progression from SOM networks. An LVQ network has two stages to its processing - a self–organising stage, which functions very much like a SOM, and a linear stage that forms associations between the first stage and the network outputs, based on the output targets specified by the user (Demuth, Beale, and Hagan, 2008). The results of using LVQ networks are discussed in the next section.

3.5.2 Classification Using Learning Vector Quantisation Networks

An LVQ network was trained using TDSC features extracted from the training audio data given in Appendix A. The codebook in use at this stage was the codebook described in Section 3.4. TDSC analysis was controlled from the function `returnSMatrices.m` which repeatedly calls `sMatReturn.m` to assemble S–matrices for all of the training data.

Referring to Figure 3.8, it can be seen that one of the `sMatSorter*` functions are required during TDSC feature extraction. For the codebook based on D_{max}=1000 and S_{max}=75, the function `sMatSorter1716.m` was called by `sMatGen.m`.

The function `multiLVQTrainer.m` was developed to train four different LVQ networks, each with a different number of hidden units. This was done to determine how the number of hidden units affected the classification results. The networks had 10, 50, 100 and 300 hidden units, 1716 inputs (as the codebooks represented 1716 D-S pairs), and 6 output units. Table 3.3 shows the assignments for the output units.

Table 3.3: *The output unit assignments used for the LVQ networks.*

Output	**Sound Category**
1	Air transport
2	Air conditioning \Ventilation
3	Birds
4	Building works
5	Rail transport
6	Road transport

3.5.3 D_{max}=1000 and S_{max}=75 Results

The classification results for the four different LVQ networks are presented in Table 3.4. Also included in this table is the time taken to train each network for 50 epochs.

These results show that for the codebook with 1716 D-S pairs, an LVQ network with 300 hidden units (300HU) produces the most accurate results with an accuracy of 55%. The other networks only managed to achieve an accuracy of 15%. When inspecting the results in detail, the 10HU, 50HU and 100HU networks had the same winning output unit for all test audio data meaning these networks were not able to differentiate between the sound categories at all. The 300HU network classified all of the category 1 (air transport), 3 (bird) and 6 (road transport) test recordings accurately but did not classify any of the other categories successfully.

Table 3.4: *LVQ classification results for TDSC analysis using D_{max}=1000 and S_{max}=75.*

Number of hidden units	Time taken to train (seconds)	Overall accuracy (%)
10	350	15
50	1260	15
100	2649	15
300	6960	55

The most accurate network (300HU) is also the network that took longest to train. Given the nature of training a network (calculating new weight vectors after each training dataset) it is not surprising that more hidden units resulted in longer training times. However, it took nearly 2 hours to train the 300HU network to achieve an accuracy of only 55%. Requiring this amount of time to train a network for this low an accuracy was not acceptable. Further experimentation using the 1716 D-S pair codebook would have resulted in long periods of waiting for networks to train with no guarantee of an improvement in accuracy. Therefore, it was decided to determine if the codebook size could be reduced further.

3.6 Further Codebook Development

In determining how the large codebook with 1716 D-S pairs could be reduced in size, all of the S–matrices used for training the LVQ networks were manually examined. It was noted that all of the S–matrices were mostly empty, containing a large amount of zeros in comparison to non-zero values. An average was calculated for how many non-zeros there were in each S–matrix. This value was less than 2% meaning that on average each S–matrix contained ∼1680 zeros. This provided a reason as to why the LVQ network struggled to differentiate between the various sound categories – i.e. all of the input data appeared to be very similar to the network.

3.6.1 Codebook Size Reduction

The D-S distributions shown in Figures 3.10 and 3.11 were re-examined to determine how the codebooks, and hence the S–matrices, could be reduced in size. Looking again at Figure 3.11, which shows the D-S distribution for all training files, there are no D-S pairs with a significant frequency of occurrence in the area above D≈200 and S≈30. Closer inspection of the frequency distribution revealed that 93.6% of all D-S pair data points are within the area D≤150 and S≤15. Using the values D_{max}=150 and S_{max}=15 the codebook was reduced in size to only 343 codes.

The new codebook also made use of duration ranges to reduce the total number of codes. Appendix B shows the duration ranges for the original codebook with 1716 codes. The new codebook, with 343 codes, used the same duration ranges but only went up to duration range 28 ($101 \leq D \leq 150$).

3.6.2 Classification Results for D_{max}=150 and S_{max}=15

Training and testing of four LVQ networks was carried out in exactly the same way as that described in Section 3.5. To generate TDSC features using the reduced-size codebook, the function `sMatSorter343.m` was called from `sMatGen.m` (see Figure 3.8 for function dependencies). Again, `multiLVQTrainer.m` was used to train the four LVQ networks, each with a different number of hidden units, and the output assignments were identical to those shown in Table 3.3. Table 3.5 shows the overall classification results for the test audio files as well as the time taken to train each network.

Table 3.5: *LVQ classification results for TDSC analysis using D_{max}=150 and S_{max}=15.*

Number of hidden units	Time taken to train (seconds)	Overall accuracy (%)
10	220	15
50	418	45
100	583	55
300	1658	50

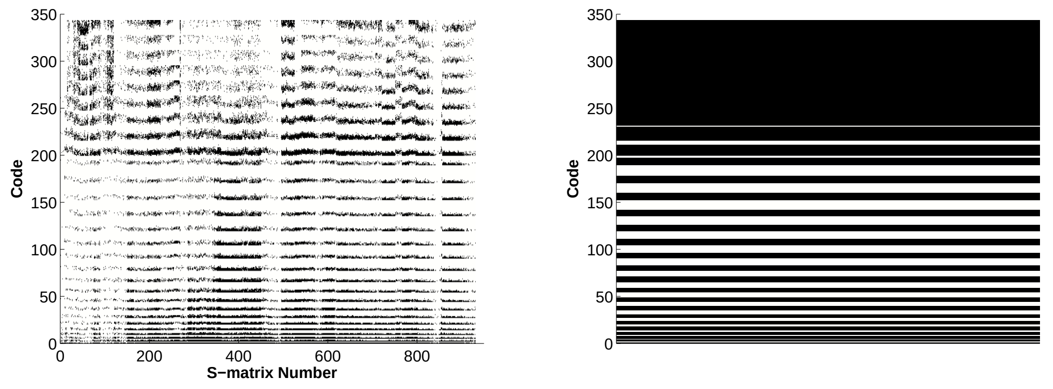
The results show a distinct improvement compared to the results achieved with the previous, larger 1716 codebook. The highest accuracy is 55%, which is the same as the 1716 codebook. However, this was achieved with fewer hidden units, and there was also significant improvement in the 50HU network. The time taken for the networks to complete training with the 343 codebook data is significantly shorter than with the 1716 codebook data. This is to be expected because there are considerably fewer network weights to be adjusted during training.

Overall, the results were satisfactory for the 343 codebook feature extraction. Although the accuracy using the 343 codebook was not an improvement over the

1716 codebook, the training times were significantly lower. The following section discusses how further developments were made to the codebook with the aim of improving the network accuracy and lowering the time taken to train the network.

3.6.3 Empty Band Discovery

Figures 3.12 shows two plots which demonstrate a banding phenomenon encountered when inspecting S–matrices generated using the 343 codebook. These plots show black where a value greater than zero was found in the resultant S–matrix. Figure 3.12(a) shows all 930 S–matrices generated for all of the test data (62 WAV files, each 3 seconds long using a framelength of 0.2 seconds). The plot shows distinctive banding of non-zero data in the S–matrices. The banding is more clearly shown in Figure 3.12(b) which is a normalised plot of a summation of all 930 S–matrices.



(a) Plot showing all 930 S–matrices for the test data files.

(b) Normalised plot of a summation of all 930 S–matrices.

Figure 3.12: *Plots showing the banding phenomenon found in S-matrices generated using the 343 codebook. Where the plots are black indicates that the S–matrix contained a value greater than zero.*

In Section 3.6.2 it was mentioned that `sMatSorter343.m` was used to generate S–matrices with the 343 codebook. Durations were grouped together into ranges to help with minimising the total number of codes in the codebook (see Appendix B). The durations from 2 through to 20 are not grouped with any other durations. For $2 \leq D \leq 20$, the code number is generated using Equation 3.5.

$$\text{Code number} = \frac{(D-1)^2 + (D-1)}{2} + S \tag{3.5}$$

The first element of this summation is the formula for calculating the triangle number of duration, D. This method of calculating the code number was originally chosen because it guarantees a minimum *allocation* of codes for a given duration. For example, if D=5, codes 10 through to 14 are all used for D-S pairs where D=5; if D=10, codes 45 through to 54 are all used for D-S pairs where D=10.

Equation 3.5 was used because it was originally assumed that an epoch could have a maximum shape of $D - 1$. However, as demonstrated in Figure 3.12, this is not the case. Table 3.6 shows durations $2 \leq D \leq 20$ and the maximum shape found in the data used to produce the plots in Figure 3.12.

Table 3.6: *Maximum shape values found for $2 \leq D \leq 20$ upon close inspection of S–matrices.*

Duration	**Max. Shape**	**Duration**	**Max. Shape**
2	0	12	5
3	1	13	6
4	2	14	7
5	2	15	7
6	3	16	7
7	3	17	8
8	4	18	8
9	4	19	8
10	4	20	8
11	5		

From Table 3.6 it can be seen that the maximum shape in any epoch for $2 \leq D \leq 20$ never exceeds $\frac{D}{2}$. The function `sMatSorter262.m` was written to take advantage of this knowledge. No formula was available for generating the correct code number given the duration for $2 \leq D \leq 20$. Therefore, the code allocation for a given duration in this range was hard-coded into the function.

TDSC feature extraction using `sMatSorter262.m` resulted in S–matrices with 262 codes, a reduction of 81 codes from the 343 codebook.

3.6.4 Classification Results using the 262 Codebook

To verify if the further reduction in codebook size improved classification accuracy, LVQ networks were trained and tested in the same manner as that described in Section 3.5 and Section 3.6.2 using the 262 codebook during TDSC feature extraction.

Table 3.7 shows the classification results for the four different LVQ networks and the time taken to train each network. As was observed with the 343 codebook, there was a reduction in the time taken to train each network because there were fewer network weights to be calculated during training.

Table 3.7: *LVQ classification results for TDSC analysis using the 262 codebook.*

Number of hidden units	Time taken to train (seconds)	Overall accuracy (%)
10	188	20
50	317	45
100	499	60
300	1395	55

The classification accuracies for the 262 codebook show an improvement over those achieved with the 343 codebook: the 10HU, 100HU and 300HU are all 5% more accurate. As with the 343 codebook, the 100HU network is the most accurate. These results show that close inspection of the S–matrices to determine how further reductions could be made were worthwhile. The 262 codebook was used for further classification tasks which are discussed in detail in Chapter 4.

3.7 Practical Considerations for TDSC Analysis

3.7.1 Framelength

The experimental work detailed throughout this chapter used audio files that were 3 seconds in length, as discussed in Section 3.3.1. It was mentioned that when performing TDSC the signal under analysis is segmented into frames. The length of these frames used in the experimental work was 0.2 seconds. This length was not selected arbitrarily.

During the early stages of the presented research, initial classification experiments were carried out, using the 262 codebook and an LVQ network, to determine how long each frame should be. A summary of these experiments is given in Table 3.8. The results show that there was little change in the classification accuracy for most framelengths. Nevertheless, the highest accuracy was achieved using a framelength of 0.2 seconds and this framelength was used for the rest of the experimental work.

Table 3.8: *Results from testing the effect of framelength on classification accuracy.*

Framelength (seconds)	**Accuracy** (%)
0.1	51
0.2	55
0.3	51
0.4	51
0.5	50
0.6	51
0.7	52
0.8	49
0.9	49
1.0	48

3.7.2 Processing Time

Throughout this chapter, and in the proceeding chapters discussing experimental work, the time taken to train classifier networks is often stated. These times allow comparisons to be made on the performance of each network during training. The experimental work was undertaken on a laptop computer running the Microsoft Windows® XP operating system. The laptop computer was not a dedicated machine for running experimental work. It was also used to perform other computing tasks. Therefore, there was a possibility that experimental work could have been performed with varying levels of demand on the processors of the laptop, depending on the other tasks the laptop was being used for. This may have resulted in large variations in the time taken to train networks.

To overcome this obstacle, the dual-core nature of the computer processor was exploited. Under the Microsoft Windows® XP operating system it is possible to assign one core of the processor to a specific task using the Task Manager. Whenever experimental work was being carried out (such as training a neural network), one core of the processor was assigned to MATLAB® . To further minimise disruptions, the laptop was left unused whilst experimental work was running in MATLAB® .

3.8 Chapter Summary

In this chapter the purely time-domain feature extraction techniques TESPAR and TDSC have been discussed. An overview of the techniques has been provided together with their history and some of the previous signal processing applications of

TESPAR and TDSC. The prior research using these techniques have demonstrated excellent classification accuracies. Although the prior works have been in different fields to the present application, their results are encouraging.

The process of designing and implementing a function in MATLAB® has been discussed, using one the functions written for the presented research as an example. An explanation of how TDSC has been implemented in MATLAB® was also given, including flow diagrams to show the different stages in TDSC analysis. All of the functions discussed throughout the chapter are reproduced in Appendix C.

The classification methodology adopted for the presented research was discussed to show the reader that the procedures used did not differ from the methodologies found in the literature. In particular, using a training data set and a testing data set is similar to the *jack–knife* and the *leave–one–out* methods used by Cowling and Sitte (2003) and Umapathy et al. (2007) respectively.

The remainder of the chapter discussed the early work carried out to arrive at a *generic* codebook which could be used with any one of the audio categories which the presented research aims to identify. The various iterations of this generic codebook were discussed along with some initial classification results. The 262 codebook showed the most promising results with an accuracy of 60% when combined with a 100 hidden unit LVQ network.

The next chapter discusses the more extensive classification tasks carried out using the 262 codebook. Experimental work using both an MLP and an LVQ classifier, in conjunction with the 262 codebook, is presented. The different classifier structures which were experimented with are described together with classification results. A bioacoustic classification problem is also discussed to allow comparison with the original uses of TDSC.

Chapter 4

Classification Using TDSC S–Matrices

The previous chapter introduced TDSC as a relatively new, but established feature extraction technique, which has had success in a variety of classification problems. Throughout the previous chapter the developments made to the generic codebook were discussed culminating in a codebook with 262 codes down from 1716.

In this chapter, experimental work carried out with the 262 codebook will be presented. The classification problem focused on is that of differentiating between the 6 urban audio sound categories using two classifier structures. The 262 codebook is also applied in a bioacoustic classification problem.

The discussion begins by examining the two different classifier structures used in the experimental work. An overview of Multilayer Perceptron (MLP) networks is then given along with the results of initial testing with MLP networks to identify the optimum design for the task at hand.

4.1 Classifier Design

In designing a classifier for an identification or classification task, there are many factors to be taken into consideration. Three factors will be discussed here: the classifier structure, the network architecture used for the classifier, and the network topology.

The *classifier structure* is how the classifier is arranged. A single classifier may be employed with multiple outputs, each output corresponding to a different category. This was the approach taken with the experimental work in the previous chapter. An alternative is to use multiple classifiers with fewer outputs, each classifier attempting to differentiate between broader categories. Defréville et al. (2006) implemented the latter approach in their work, as discussed in Chapter 2. Defréville et al. also implemented classifiers which were trained to identify only a single category, so called *expert* classifiers. The other prior pieces of research discussed in Chapter 2 either used slight variations on the structures mentioned above or do not report the structure.

The term *network architecture* is used to mean the type of network (MLP, LVQ, Radial Basis Function, etc.) being used. Closely linked to the architecture is the *topology* of the network. The architecture used in the prior works discussed in Chapter 2 is as varied as the structures described above. However, the authors of the prior works do not discuss the network topologies used.

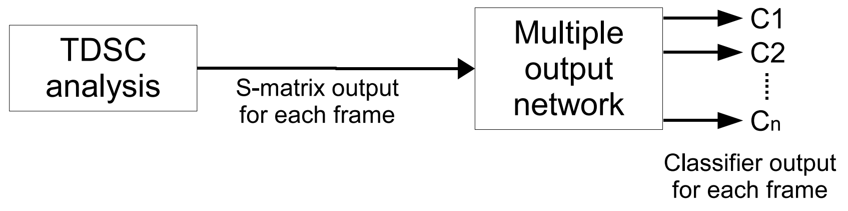
It should therefore be clear that there are many options for the structure, architecture and topology of a classifier in a classification system. Unfortunately, there is little in the literature to assist in deciding on a classifier structure prior to starting a classification task. Some authors report results based on trying a number of different classifier structures and different architectures (for example, see Hawickhorst et al. 1995, Cowling and Sitte 2003, Umapathy et al. 2007 and as already mentioned Defréville et al. 2006). Books on neural network design (such as Hagan et al. 1995) go into great detail of how a particular type of network can be optimised, in terms of topology, for various tasks but provide little in the way of advice on selecting a structure to use. However, even determining a network topology is not straightforward. Under the heading 'Selecting a topology for an MLP network' Rafiq, Bugmann, and Easterbrook (2001) state that

> "Every stage of any [Neural Network] project requires a little trial and error to establish a suitable and stable network…" (p.1545)

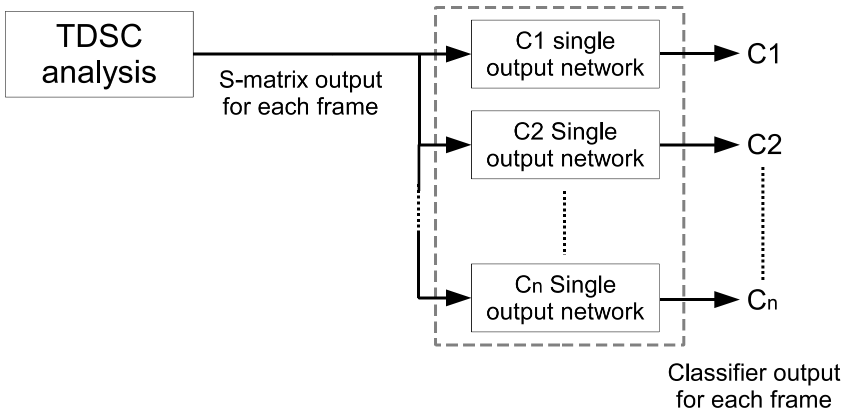
The advice of Rafiq et al. (2001) has already been observed in the experimental work presented in Chapter 3 where a variety of LVQ topologies were trialled as it was not previously known which would provide the best classification accuracy. For the 262 codebook, it was observed that an LVQ network with 100 hidden units gave the best results. Later in this chapter results from similar topology testing with MLP networks is presented.

4.1.1 Classifier Structures

Two different classifier structures were used in the work presented in the rest of this chapter. These two structures are shown in Figure 4.1.


TDSC analysis
S-matrix output for each frame
Multiple output network
C1
C2
Cn
Classifier output for each frame

(a) Using a single multiple–output classifier.



(b) Using multiple single–output expert classifiers.

Figure 4.1: *The two classifier structures used during experimental work.*

Figure 4.1(a) shows a classifier structure which employs a single network. The network is trained to differentiate between the classes C1,C2...C$_n$. Recalling that the input audio file is analysed in frames, an S–matrix is generated for each frame and the classifier is trained to map S–matrices for the different categories of audio to one of C1,C2...C$_n$. The ideal result for this structure is that for a given S–matrix belonging to a signal from class C1, for example, the output for C1 comes out as a '1' and the rest of the outputs come out as '0'.

The structure shown in Figure 4.1(b) employs a series of single-output classifiers, each trained to identify only a single class from the set C1,C2...C$_n$. For this structure, an ideal result would be for only one of the single-output classifiers to score a '1' on its output and all others to score a '0'. In this configuration the collection of networks performs as the overall classifier stage of the classification system.

Using these two different classifier structures allows the experimental work from the presented research to be compared with earlier works where authors have trialled a number of classifier architectures, topologies and structures.

4.2 Multilayer Perceptron Theory

MLP networks were introduced in Chapter 2 as a neural network having more than one layer of neurons. A standard neural network unit (such as the McCulloch-Pitts model shown in Figure 2.2) or group of units are usually called a perceptron. Perceptrons consist only of inputs and output unit(s), as shown in Figure 4.2, and have 2 layers. A simple perceptron can learn the association between an input data set and its outputs if the output is a linear transformation of the input (Abdi, 1994). An example problem often quoted in the literature to show the capability of a simple perceptron is that of the logical OR operation because it is a linearly separable function (Abdi provides an excellent account of this example).

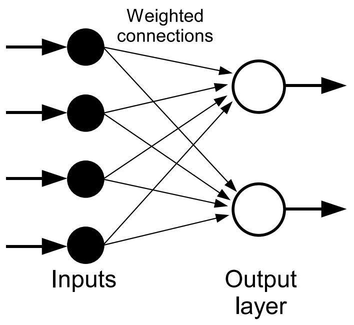


Figure 4.2: *A simple perceptron network. The network has two layers - the inputs and the output layer. The inputs are connected to the outputs via a set of weighted connections.*

Many classification problems do not present a linearly separable relationship between input and output. The logical XOR operation is an example of this (the reader is again directed toward Abdi (1994) to see this example). For a perceptron network to be able to build associations between inputs and outputs for non-linearly separable relationships, a hidden layer of neurons must be introduced into the network. Figure 4.3 shows a perceptron network which includes a hidden layer.

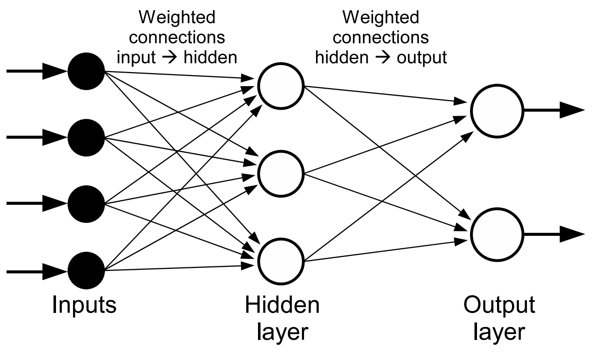

Weighted connections input → hidden
Weighted connections hidden → output
Inputs
Hidden layer
Output layer

Figure 4.3: *A perceptron network with one hidden layer. In this network there are no direct connections between the inputs and the output layer. The inputs are connected to the hidden layer units with a set of weighted connections and the hidden layer units are connected to the output layer units with another set of weighted connections.*

In both of the network topologies shown in Figures 4.2 and 4.3 the connections between the layers are weighted. To illustrate the purpose of the weighting, let Y_i be the output value of output unit i in the network of Figure 4.2, w_{ij} is the connection weight from input unit j, X_j is the input fed into unit j, and $g(f_i)$ is the activation function of O_i. The output of unit i can be expressed as:

$$Y_i = g(f_i) = g\left(\sum_j w_{ij} X_j\right) \tag{4.1}$$

Equation 4.1 shows that each connection in the network can have a different weighting and will therefore have different effects on the values being passed between units. For example, input 1 may have a weighting of 0.5 for its connection to output unit 1 but a weighting of 0 for its connection to unit 2. The value of input 1 will have a greater effect on the output of output unit 1 than that of output unit 2. During training of a network the weights are optimised such that a particular input pattern to the network will trigger a particular output pattern.

The calculations and equations which back up the theory of MLP computation are very well documented in the literature covering the subject of neural networks. See, for example, Hertz et al. (1991), Abdi (1994) and Ham and Kostanic (2001).

4.3 Implementing MLP Networks with MATLAB®

The MATLAB® Neural Network Toolbox (NNT) includes many different network architectures. The LVQ networks discussed in Chapter 3 were generated using the NNT. As well as offering many architectures, the NNT offers many options which affect the internal behaviour of a network during training and simulation.

In the MATLAB® 'Neural Network Toolbox User's Guide', Demuth, Beale, and Hagan (2008) provide excellent explanations for many of the network options. This reference was consulted extensively throughout the experimental work reported here. The MLP network architecture comes under the title of "Backpropagation" in Demuth et al. (2008), this being a particular method of training an MLP network[1]. Backpropagation learning is so called because it involves back-propagating the network error (the difference between the expected and actual outputs) through the network and adjusting the weights accordingly (Rafiq et al., 2001). Backpropagation is a form *supervised* training, and for it to work successfully the network needs to know the expected output for each input dataset.

When creating MLP networks with MATLAB®, the following options were considered carefully prior to choosing a particular network design:

- topology,
- layer transfer functions,
- training function, and
- performance function.

Choosing the number of inputs and output units for each network is usually determined by the size of the input data and the number of classes the network was to differentiate between. For the presented research, the size of the input data was 262×1 for each frame of TDSC data. The number of output units depended on the network structure under consideration (see Section 4.1.1).

4.3.1 Network topology

The subject of network topology has already been discussed in terms of whether an MLP network has a hidden layer or not. The number of units in the hidden layer is

[1]More specifically a *feed-forward* MLP network which are the only MLP networks the presented research is concerned with.

one of the options available when creating an MLP network as well the number of hidden layers.

With regard to the number of hidden layers, Hertz et al. (1991) offer a non-rigorous proof that two hidden layers are enough. Rafiq et al. (2001) state that "*. . . a single [hidden] layer with an optimum number of neurones will be sufficient for modeling many practical problems*" (p.1545).

In Chapter 3 a variety of LVQ topologies were tested to determine the optimal number of hidden units. A similar series of tests were carried out with MLP networks which are discussed further in Section 4.4. The exact number of hidden units used in LVQ and MLP networks varied depending on the classification problem. For example, in classifying urban audio data using a multiple–output MLP network, it was found that 100 hidden units gave the highest accuracy.

4.3.2 Layer transfer functions

Each neuron in an MLP network will have a transfer function associated with it. The transfer function of a unit determines the scalar output of the unit for a given weighted input. The NNT offers a selection of different transfer functions that can be used. When creating a network, the transfer functions for the hidden layer(s) and output layer are selected and applied to all units in those layers. Figure 4.4 shows the most common transfer functions used with MLP networks according to Rafiq et al. (2001).

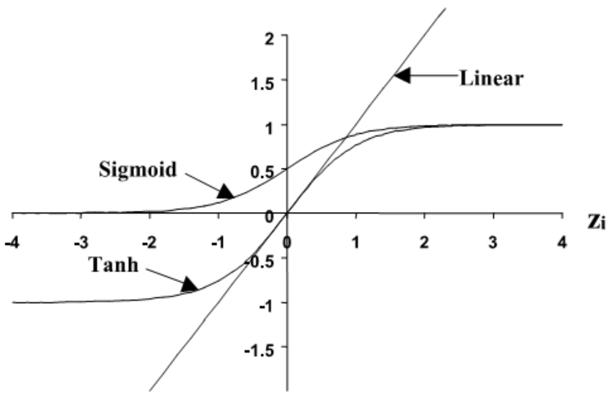


Figure 4.4: *Common transfer functions that are used in MLP neural networks. Taken from Rafiq et al. (2001).*

The sigmoid transfer function (`logsig` in the NNT) will transform an input with a value between plus and minus infinity into a number in the range 0 to +1. The tanh function (`tansig` in the NNT) performs a similar operation but its output range is -1 to +1. The output of a linear transfer function (`purelin` in the NNT) is identical to its input and can have any value.

4.3.3 Training function

The training function for a network is selected when the network is created. The chosen function determines the algorithm used during training. In the basic back-propagation training algorithm *"...the weights are moved in the direction of the negative gradient..."* (Demuth et al., 2008, p.5-15). The gradient referred to here is the gradient of the performance function and is used to determine how the weights are adjusted.

For an MLP network, the NNT offers 13 different training functions. The training function can have a significant effect on the network, both in terms of its accuracy after training and the time taken to train the network. A table summarising the different training functions is provided in Appendix D.1.

4.3.4 Performance function

The performance function acts as a target for the network. The weights are adjusted during training to try and minimise the performance function (Demuth et al., 2008). There are two options for performance functions discussed by Demuth et al. under the topic of backpropagation. The standard performance function is `mse` – the mean square error between the network outputs and the target outputs. The other performance function option is `msereg` – a modified mean square error calculation which takes into account the mean of the sum of the squares of the network weights (Demuth et al., 2008). The latter of these functions is used for improving generalisation and is more useful for function approximation problems as opposed to pattern recognition problems. Therefore, the standard `mse` performance function was used during experimental work.

4.3.5 Selecting Network Options

To determine which network options were most suitable for the present classification task, a series of tests were carried out with different network configurations, making changes to the transfer and training functions used in each. Trialling all of the possible combinations of transfer and training functions would have been a needless and time-consuming task.

Highlighting the issue of the complexity of choosing network options, Demuth et al. (2008) provide a speed and memory comparison for nine of the training functions detailed in Appendix D. In their comparison they present results for six classification problems: three pattern recognition problems and three function approximation problems. The pattern recognition problems are relevant to the presented research so these were focused on. In summary, Demuth et al. state that `trainrp`, a *resilient* backpropagation training algorithm, provides the fastest results for pattern recognition problems. Another training algorithm of note from the authors is the Levenberg-Marquardt algorithm (`trainlm`) which demonstrated excellent results and fast training but only for function approximation problems.

Studying the results allowed some preliminary decisions to be made on the design of the MLP networks that would be tested. Three training algorithms were identified that would be suitable for the present application; variable learning rate with momentum (`traingdx`), resilient backpropagation (`trainrp`), and scaled conjugate grading (`trainscg`). These training functions were chosen based on their performance across the different pattern recognition problems presented by Demuth et al.. Also, according to Kurt, Ture, and Kurum (2008), any of the training functions with variable learning rate and momentum will generally perform better than standard gradient descent algorithms.

It was also noted from the results of Demuth et al. (2008) that the MLP networks used for pattern recognition all used sigmoid transfer functions in both the hidden and output layers. This echoes the design suggestions from Rafiq et al. (2001). For the output layer, it was decided to use a log–sigmoid function because the desired response from each output was a 0–1 value for ease of comparison.

A summary of the network options used throughout this research is given in Table 4.1.

Table 4.1: *Summary of the network options used throughout the presented research. During the preliminary testing for each classification task, all of the possible topologies were tested for MLP and LVQ networks. Preliminary testing for MLP networks also included all possible combinations of transfer function and training function.*

Topology (no. of hidden units)	**Transfer function**	**Training function**	**Performance function**
10	log–sigmoid	`traingdx`	mean squared error
50	tan–sigmoid	`trainrp`	
100		`trainscg`	
300			

4.4 Preliminary MLP Testing

In order to determine the most appropriate network design for classifying the audio signals detailed in Appendix A.2, preliminary testing was carried out on the network designs shown in Table 4.2.

The data used to train and test the networks used audio from only the *air conditioning/ventilation*, *bird* and *road* categories. Complete results of the testing are presented in Appendix D.2 and are summarised in Table 4.2.

The MLP networks were created, trained and simulated in the same manner as that described in Section 3.3. Each network was trained with S–matrix features extracted from the training audio set and simulated with S–matrix features extracted from the test audio set.

Each network was given a maximum number of training epochs of 4000. This figure was based on short initial tests. Each network had a performance goal of 0.001 meaning that when the mean squared error (`mse`) reaches this value, training will stop. The value of 0.001 was chosen based on the example networks used by Demuth et al. (2008). The learning rate chosen for all networks was 0.05. This was the default value for each network and was left unchanged because Demuth et al. comment that for the faster networks the default learning rate is usually adequate.

Table 4.2: *Summary of results from preliminary testing with MLP networks. The top performing network for each topology is presented in bold typeface.*

Topology	Training function	Hidden layer transfer function	Training time (seconds)	M–value mean
10–3	`traingdx`	`tansig`	35.95	0.751
		`logsig`	49.72	0.748
	`trainrp`	`tansig`	2.06	0.776
		`logsig`	1.00	0.748
	`trainscg`	`tansig`	8.95	0.501
		`logsig`	**6.25**	**0.793**
50–3	`traingdx`	`tansig`	133.24	0.762
		`logsig`	133.59	0.725
	`trainrp`	`tansig`	2.10	0.754
		`logsig`	**2.15**	**0.764**
	`trainscg`	`tansig`	21.23	0.761
		`logsig`	23.72	0.518
100–3	`traingdx`	`tansig`	243.06	0.769
		`logsig`	250.07	0.735
	`trainrp`	`tansig`	**5.91**	**0.790**
		`logsig`	4.41	0.773
	`trainscg`	`tansig`	46.83	0.753
		`logsig`	43.41	0.749
300–3	`traingdx`	`tansig`	750.08	0.734
		`logsig`	749.51	0.709
	`trainrp`	`tansig`	**16.22**	**0.791**
		`logsig`	13.57	0.764
	`trainscg`	`tansig`	164.42	0.786
		`logsig`	167.38	0.745

The results presented in Table 4.2 show that the resilient backpropagation algorithm is the fastest to train with all network topologies, training in as little time as 1 second for a 10–3 network[2]. It was observed with the LVQ networks in Chapter 3 that as the number of hidden units increased the time taken to train also increased. This was also true for the MLP networks, as the results show.

The right–most column of Table 4.2 shows the mean of the slope gradients for post–simulation regression calculations for each network output. If the slope is

[2]The h–o nomenclature is used throughout the thesis to describe the layers of the network, where h denotes the number of hidden units and o denotes the number of output units. A 10–3 network has 10 hidden units and 3 output units.

represented by $y = mx + c$, then the slope is given by m. Hence, the regression slope values are referred to as *M–values*. The regression was performed on the simulated network output for the S–matrices extracted from the test audio data and the targets (the expected outputs) for each of the S–matrices. The slope gradient from the regression analysis in MATLAB® (with the function name `postreg`) provides an indication of how close the simulated output of a network is to the expected output. If the slope gradient is a 1 there is a perfect match between the simulated and expected outputs.

The results in Table 4.2 indicate that the network with the closest match between the simulated outputs and the expected outputs was the 10–3 SCG network (scaled conjugate gradient). There is a noticeable difference between the results for the two 10–3 SCG networks, with the `logsig` hidden layer network performing much better than the `tansig` one. Such a difference in results was not expected and it will be seen later in this chapter that such a difference is not always observed.

Although the 10–3 SCG `logsig` network had the highest overall accuracy, the RP (resilient backpropagation) networks were more consistent in their results. The 100–3 RP `tansig` network had a mean slope gradient which was only 0.003 lower than that of the 10–3 SCG `logsig` network, a negligible amount. The training times of all of the RP networks were much lower than those of the other two networks.

To determine which network design was to be used in the experimental work involving the complete audio data set, the network designs were considered in topology groups and mean accuracies were calculated. The 100–3 topology group had the highest mean accuracy of 0.762. Given the relatively short training times of the MLP networks (compared to the LVQ networks presented in Chapter 3), it was decided that all six of the designs with 100 hidden units would be used for experimental work.

4.5 Urban Audio Event Classification

This section presents classification results obtained from experimental work using various network structures. Both LVQ and MLP networks were generated for each of the classifier structures shown in Figure 4.1 resulting in four experiments being carried out for the urban audio data (see Appendix A).

The results presented here are summaries of the results obtained during the experimental work. Full results can be found in Appendix E.

4.5.1 Experiment 1: Multiple–output MLP Network

In Section 4.4 results were presented of the preliminary testing performed to determine the optimal MLP network structure. It was seen that an MLP network with a 100–3 topology outperformed all other topologies. Presented below are the experimental results for MLP networks of this topology in all 6 combinations of training and transfer functions.

Training and testing was repeated 5 times with a new network initialisation for each run. This method of validation is recommended by Demuth et al. (2008) as each initialisation results in different initial weight vectors for the networks. Repeating the training and testing process also provides more robust results for the accuracy of the networks because the results from each session all contribute towards the mean results.

Mean regression M–values were calculated for each network design. A summary of these is presented in Table 4.3. The full results for all 5 training and testing sessions can be found in Appendix E.1.

Table 4.3: *Summary of performance results for 100–6 MLP networks classifying urban audio data. The training times and regression slope means are the averages for 5 training and testing sessions.*

Training function	**Hidden layer transfer function**	**Training time (seconds)**	**M–value mean**
`traingdx`	`tansig`	431.10	0.668
	`logsig`	427.86	0.669
`trainrp`	`tansig`	13.01	0.624
	`logsig`	12.07	0.607
`trainscg`	`tansig`	**229.83**	**0.707**
	`logsig`	235.92	0.671

Table 4.3 shows that the MLP network trained using a `trainscg` (conjugate gradient backpropagation) training function with a `tansig` hidden layer transfer function produced the best regression results. This contrasts the results observed during the preliminary testing in which the networks trained using resilient backpropagation were the best performers. The mean of the regression M–values are overall considerably lower than the M–values seen in the preliminary testing. The

increase in categories (and hence outputs) is the most likely cause of this reduction in network accuracy.

Whilst the regression M–values provide an indication of how the network is performing, further tests were carried out to determine how well the `trainscg-logsig` 100–6 network could identify individual sounds. The data used for this testing comprised longer audio files than those used for training and testing. The audio files did not contain any of the same data that had been used to train the networks.

The winning output of the network was retrieved for each test file. Results for these tests are presented in Table 4.4. The `trainscg-tansig` network from the second training and testing session was used during this process as it had the highest overall M–value.

Table 4.4: *Individual WAV file classification results for the multiple–output 100–6 MLP network. The fourth column shows how many frames of the audio data were correctly classified as a percentage.*

Sound category	**File**	**Desired output**	**% correct desired output**
Air transport	`AirTrans002`	1	90
Air conditioning	`AirCon001`	2	73
Bird	`Soundrec036_Bird`	3	81
Building works	`digger_drill`	4	100
Rail transport	`train_idle`	5	50
Road transport	`Soundrec024_Road`	6	93
Mean			81

The results in Table 4.4 show that the combination of the 262 S–matrix feature extraction and the 100–6 `trainscg-tansig` network can accurately classify audio data that it has not previously encountered. The poorest result was achieved when classifying a recording of rail transport. The variety of sounds encompassed in each category can be quite large so achieving an accuracy of only 50% for the rail transport category is not wholly unexpected.

4.5.2 Experiment 2: Multiple Single–output MLP Networks

The results presented in this section relate to using an MLP network with only one output for each of the six sound categories, as shown in Figure 4.1(b). The intention was to train the MLP networks to become expert classifiers for only one category:

an output of a 1 indicating that the input frame of audio belongs to the category for which the classifier has been trained; an output of a 0 indicating that the input data does not belong to the category.

Preliminary Tests

Prior to initialising, training and testing six networks, preliminary tests were carried out to determine which network architecture was most appropriate for this application. The tests were of a similar vein to the preliminary tests that had been carried out for the other network architectures (i.e. networks of all combinations of training and transfer functions were trialled with a reduced data set). Appendix E.2 presents the results of the preliminary tests.

The preliminary test results (presented in Table E.2) showed that the best performing network overall across all of the three test categories was a 100–1 network trained using the conjugate gradient algorithm with `logsig` transfer functions in the hidden layer (M–value mean 0.7817). Therefore, training and testing of MLP single–output networks for all six sound categories was carried out using this network architecture.

Test Results for All Categories

Training and testing was repeated five times with new network initialisations for each session, as with the 100–6 network discussed earlier. Regression analysis results for all networks in all five training sessions are presented in Appendix E.3. Summarised results are presented in Table 4.5. It is to be noted that the results from the fourth training and testing session were omitted when calculating the mean values presented in Table 4.5. The results from the fourth session were considerably different from all of the other sessions and were therefore considered anomalous results.

Table 4.5 shows that after training with the training audio data (detailed in Appendix A), all of the single–output networks achieved at least a mean M–value of 0.55 across all training sessions. The networks were choosing between two possible results so this result is better than would be achieved through guesswork.

Table 4.5: *Summary of results for six 100–1 MLP networks. Each network was trained as an expert classifier to recognise only one category of urban audio event and reject all others.*

Category	**Time to train (s)**	**M–value mean**
Air transport	7.23	0.824
Air conditioning	17.98	0.589
Bird	10.76	0.701
Building works	17.46	0.702
Rail transport	10.63	0.551
Road transport	30.79	0.719

Results were presented earlier of how well the 100–6 MLP network performed with audio data it had not previously seen. To carry out similar tests with six 100–1 MLP networks, a function was written to analyse the outputs of all six networks and determine the winner (see the function `singleOpNetTest.m` in Appendix C).

Table 4.6 shows the results of the tests performed with previously unseen audio data. The audio files used were the same as those from the 100–6 MLP network tests to allow a comparison to be made. It is clear to see from the results in Table 4.6 and Table 4.4 that the classifier constructed from the single–output networks did not perform as well as the multiple–output network. In most categories the percentage of correctly classified frames is lower for the single–output network classifier. Only the *bird* category is higher, and the *air conditioning* category is identical.

Table 4.6: *Individual WAV file classification results for the six single–output 100–1 MLP networks. The fourth column shows how many frames of the audio data were correctly classified as a percentage.*

Sound category	**File**	**Desired winning network**	**% correct desired winning network**
Air transport	`AirTrans002`	1	87
Air conditioning	`AirCon001`	2	73
Bird	`Soundrec036_Bird`	3	85
Building works	`digger_drill`	4	97
Rail transport	`train_idle`	5	48
Road transport	`Soundrec_024_Road`	6	88
Mean			80

4.5.3 Experiment 3: Multiple Output LVQ Network

In Chapter 3 a multiple–output LVQ network was developed and implemented for the initial codebook development. In particular, Section 3.6.4 discusses the initial test results achieved using LVQ networks with different numbers of hidden units and six output units. The results showed that an LVQ network with 100 hidden units had the best overall performance. Therefore, an LVQ network of this topology was used to generate the experimental results presented below.

As with earlier experiments, training and testing was repeated five times with a new network initialisation each time. A summary of the results is presented in Table 4.7. Full results for all five training and testing sessions can be found in Appendix E.4.

The fourth column of Table 4.7 shows the M–value mean across all six outputs of the LVQ network. Recalling that an M–value of 1 indicates a perfect match between the simulated network outputs and the desired outputs, the results displayed for the 100–6 LVQ network are very poor. The full results for the five sessions show that outputs four and five always had an M–value of 0 indicating that none of the *building* or *rail* category test files were correctly classified. This correlates with the results observed during the initial testing discussed in Section 3.6.4. None of the *building* or *rail* category test files were correctly classified during those tests either.

Table 4.7: *Mean results for all five training and testing sessions for a 100 hidden unit, 6 output LVQ network.*

Session	**Topology**	**Time to train (s)**	**M–value mean**
1	100-6	501.48	0.372
2		511.03	0.262
3		489.03	0.314
4		484.94	0.383
5		496.80	0.413
Mean		496.66	0.349

To verify the M–values from the regression analysis post–training, individual audio files previously unseen by the network were analysed and classified. The number of frames accurately classified was calculated for each sound. The results of this test are presented in Table 4.8. The classifier was not able to correctly classify any frames of audio data from the *building works* or *rail transport* categories. The results for the *air transport*, *air conditioning* and *road transport* categories are all very high. However, the overall accuracy of this network is only 52% due to poor results in three of the categories.

Table 4.8: *Individual WAV file classification results for the multiple–output 100–6 LVQ network. The fourth column shows how many frames of audio data were correctly classified as a percentage.*

Sound category	**File**	**Desired winning network**	**% correct desired winning network**
Air transport	`AirTrans002`	1	90
Air conditioning	`AirCon001`	2	87
Bird	`Soundrec036_Bird`	3	38
Building works	`digger_drill`	4	0
Rail transport	`train_idle`	5	0
Road transport	`Soundrec_024_Road`	6	99
Mean			52

4.5.4 Experiment 4: Multiple Single–output LVQ Networks

The aim of this experiment was to test a classification system having the structure shown in Figure 4.1(b) where each output was an individual LVQ network with only one output. Each LVQ network was to be trained as an expert classifier for only one category.

In accordance with previous experiments, preliminary testing was performed to determine the most suitable network topology for the application. The results of the preliminary testing are not presented here because all of the networks tested gave an M–value very close to zero (of the order 1×10^{-15}).

A modification of the classifier structure was tested in an attempt to improve the accuracy. The modification involved using two–output LVQ networks instead of single output networks. If the first output of a network was the winning output then the input data belonged to the category associated with that network. If the second output of a network was the winning output then the input data did not belong to the category associated with that network. Preliminary tests with the modified classifier also yielded M–values very close to zero (again, of the order 1×10^{-15}).

Given the very poor results achieved during the preliminary tests, it was decided not to carry out training and testing with the full data set.

4.5.5 Urban Audio Event Classification – Results Summary

In this section results for experimental work using the 262 codebook with a variety of classification system structures have been presented. For all of the structures preliminary testing was first carried out to determine the most suitable classifier topology and, in the case of the MLP networks, the most suitable training/transfer function.

The highest accuracy (81%) was achieved with an MLP network having 100 hidden units and 6 outputs (one for each of the sound categories) which was trained using the `trainscg` algorithm and used a `tansig` output layer transfer function. Using single–output MLP networks also presented promising results (80% accuracy).

Using LVQ networks as classifiers was less successful with the highest accuracy of 52% being achieved by an LVQ network with 100 hidden units and 6 outputs. No results were gathered using single–output LVQ networks as the preliminary test results indicated they would perform poorly.

Overall, the results were very positive when using MLP networks to classify urban audio data that has had features extracted using the 262 codebook. In the next section the 262 codebook is used as a feature extractor in a classification system which attempts to classify bioacoustic audio data. One of the original uses of TDSC was to identify bioacoustic signals. Therefore, it was decided to investigate how TDSC using the 262 codebook compares to standard TDSC.

4.6 Bioacoustic Signal Classification

Time-Domain Signal Coding has previously been used successfully to identify bioacoustic signals. Some of these applications were discussed in Chapter 2. In particular, the work of Dr. David Chesmore (Chesmore, 2001) demonstrated that TDSC can extract features which allow 100% classification accuracy of *Orthoptera* sounds under low–noise conditions. The work of Farr (2007) showed that TDSC has its uses in identifying insect bites inside wooden materials.

The experimental work discussed in this section relates to an identification problem previously studied by E. Ohya (Ohya, 2004). In this study, the author aimed to discriminate between the songs of three species of Japanese cicada: *Tibicen bihamatus*, *T. flammatus* and *T. japonicus*. These particular species of cicada are known to be difficult for humans to distinguish between due to their very similar sounding songs and their habit of singing high in trees.

Ohya (2004) extracted the peak and mean frequencies of each song recording and used Principal Components Analysis (PCA) to compare these features to those of a training set. The training set consisted of three known recordings. Ohya's results showed PCA to be a reliable method of classifying the songs. However, of 22 recordings made by Ohya, 10 suffered from low signal–to–noise ratios and leakage from other noise sources so were not used in his study.

Chesmore (2004) used a data set of 25 cicada recordings (made by Ohya) to test an MLP network. The test data set included five low–quality recordings. The network had been trained using the same three known recordings used by Ohya. Chesmore's results showed the trained network was capable of correctly classifying the training data. With the test data, 17 of 25 recordings were correctly classified.

The same data set used by Chesmore (2004) was used to test how accurately a classification system using the 262 codebook could identify *Tibicen* cicada songs. The classifier structures shown in Figure 4.1 were implemented using both MLP and LVQ networks.

4.6.1 Preliminary Testing

Preliminary tests were performed to identify the network topologies which suited the identification problem. These tests were performed in the same manner as the preliminary tests carried out for the urban audio signals. That is, MLP and LVQ networks with different topologies were tested using both classifier structures shown in Figure 4.1.

The data used for the preliminary testing was the same as the data used to generate the actual test results presented below. There was not enough data available to allow a reduced–size data set to be made for the preliminary tests. Details of the *Tibicen* recordings can be found in Appendix A.3.

Full results from preliminary testing with 3–output MLP networks are presented in Appendix E.5. The results show that an MLP network with 10 hidden units trained using a conjugate gradient descent algorithm (`trainscg`) with a `tansig` hidden layer transfer function had the highest percentage (86.41%) of correctly classified TDSC frames[3].

Preliminary tests were also performed to find the ideal topology for a 3–output LVQ network. A table of these results can be found in Appendix E.6. The results show that an LVQ network with 100 hidden units had the highest percentage of correctly classified TDSC frames (86%).

Further preliminary tests were performed to identify network topologies for MLP and LVQ networks with only single outputs, as shown in Figure 4.1(b). However, the preliminary results showed accuracies that were very low for both the MLP and LVQ networks tested, based on regression M–values. Percentages for correctly classified frames were calculated and these were also very low (∼5%). Therefore, full tests were not carried out with either MLP or LVQ single–output networks.

4.6.2 Experiment 1: Multiple Output MLP Network

The preliminary testing showed that the MLP network with the highest performance results had 10 hidden units and was trained using the `trainscg` algorithm using a `tansig` output layer transfer function. Full testing was carried out using an MLP network of this architecture. As with previous tests, training and testing was repeated five times with a new network each time, thus allowing a new set of network weights to be established each time. The results of the five training and testing sessions are presented in Table 4.9. The performance for the network is based on the percentage of correctly classified frames[3].

Table 4.9 shows that over the 5 sessions the MLP network architecture achieves a mean correct percentage of 69.02%. To allow a more thorough analysis of the classifier accuracy, the network from session 5 was used to generate accuracy results for each test file individually. That is, the network was simulated with TDSC data for each test file and the simulation results were directly compared with the target values. Table 4.10 presents the results of this analysis.

[3]For all of the *Tibicen* tests, the percentage of correctly classified TDSC frames was used to determine the best performing topology/network. In previous tests regression analyses were used for this purpose. The mean M–values produced by the MATLAB® `postreg` function were in the region of 0.25–0.35 indicating that all of the network topologies were performing poorly. However, the `postreg` function always output M–values of 0 for output 2 (*T. flammatus*) because none of the test files belonged to this category. It was therefore decided that the percentage of correctly classified frames would be a more reliable indicator of network accuracy.

Table 4.9: *Performance results for a 10–3 MLP network classifying* Tibicen *recordings.*

Session	**Topology**	**Time to train (s)**	**Frames correctly classified (%)**
1	10-3	0.39	56
2		0.32	73
3		0.39	65
4		0.29	71
5		0.38	80
Mean		0.36	69

The results in Table 4.10 show that 14 out of the 25 (56%) test files have 100% of their frames correctly classified. If the highest percentage of correctly classified frames for each test file is taken as the winning category, then 22 out of the 25 (88%) test files are classified into the correct category. Using the 262 codebook with a 10–3 MLP network outperformed the results presented by Chesmore (2004).

4.6.3 Experiment 2: Multiple Output LVQ Network

Based on the preliminary test results, an LVQ network with 100 hidden units was used to perform full tests on the *Tibicen* data. Like all of the previous experiments, training and testing was performed five times with a new network initialised for each session. The results are presented in Table 4.11 and show that the mean number of frames correctly classified was 72%. This result is an improvement over the mean for the MLP network discussed above.

To allow further comparisons between the results achieved with the MLP and LVQ networks and the work of Chesmore (2004), all of the *Tibicen* test files were analysed individually by an LVQ network. The network from session 4 had the highest percentage of frames correctly classified and was therefore used for the individual file analysis. The results are presented in Table 4.12.

The results presented in Table 4.12 show that the LVQ network was able to correctly identify 19 out of 25 (76%) of the test files when the highest percentage of correctly classified frames is used as the winning category. This is fewer than was achieved with the MLP network. However, the LVQ network achieves a higher number of correct classifications with 100% of the frames correctly classified (15 out

Table 4.10: *Individual classification accuracy results for* Tibicen *test files using a 10–3 MLP network. To identify where mis–classifications occurred, the percentage of frames classified in all three categories were calculated. Results for the training files are also included to show how well the classifier can differentiate between the categories.*

File	**Target**	**Frames classified as BH (%)**	**Frames classified as FL (%)**	**Frames classified as JP (%)**
Test files				
1.wav	JP	25.00	6.25	68.75
2.wav	BH	66.67	33.33	0.00
3.wav	BH	80.00	0.00	20.00
4.wav	BH	6.25	93.75	0.00
5.wav	BH	0.00	31.25	68.75
6.wav	BH	100.00	0.00	0.00
7.wav	BH	75.00	12.50	12.50
8.wav	BH	100.00	0.00	0.00
9.wav	BH	100.00	0.00	0.00
10.wav	BH	100.00	0.00	0.00
11.wav	BH	100.00	0.00	0.00
12.wav	BH	100.00	0.00	0.00
13.wav	BH	100.00	0.00	0.00
14.wav	BH	100.00	0.00	0.00
15.wav	BH	100.00	0.00	0.00
16.wav	BH	68.75	25.00	6.25
17.wav	BH	0.00	93.75	6.25
18.wav	BH	100.00	0.00	0.00
19.wav	BH	93.75	6.25	0.00
20.wav	BH	100.00	0.00	0.00
lq1.wav	BH	60.00	26.67	13.33
lq2.wav	BH	100.00	0.00	0.00
lq3.wav	BH	100.00	0.00	0.00
lq4.wav	BH	93.75	0.00	6.25
lq5.wav	BH	100.00	0.00	0.00
Training files				
BH.wav	BH	100.00	0.00	0.00
FL.wav	FL	0.00	100.00	0.00
JP.wav	JP	0.00	0.00	100.00

of 25 compared to 14 out of 25 for the MLP network). It is interesting to note that the LVQ network did not identify any frames as belonging to the *T. japonicus* category. This includes the training file frames. The MLP network was able to identify both the test and training *T. japonicus* files.

Table 4.11: *Performance results for a 100–3 LVQ network classifying* Tibicen *recordings.*

Session	**Topology**	**Time to train (s)**	**Frames correctly classified (%)**
1	100–3	54.91	70
2		52.86	72
3		54.43	72
4		54.09	74
5		53.87	71
Mean		54.03	72

4.6.4 Bioacoustic Signal Classification – Results Summary

In this section results have been presented of experimental work using the 262 codebook in a variety of classification system structures for a bioacoustic classification problem. Preliminary testing was carried out to find the most suitable network topology.

The highest overall accuracy was achieved with an LVQ network having 100 hidden units and 6 output units (71.80%). An overall accuracy of 69.02% was achieved using a multiple–output MLP network (100 hidden units, `trainscg` training function, `tansig` hidden layer transfer function). No results were achieved using single–output LVQ or MLP networks because preliminary testing results indicated that the overall results would be very poor.

Table 4.12: *Individual classification accuracy results for* Tibicen *test files using a 100–3 LVQ network. To identify where mis–classifications occurred, the percentage of frames classified in all three categories were calculated. Results for the training files are also included to show how well the classifier can differentiate between the categories.*

File	**Target**	**Frames classified as BH (%)**	**Frames classified as FL (%)**	**Frames classified as JP (%)**
Test files				
1	JP	25.00	75.00	0.00
2	BH	0.00	100.00	0.00
3	BH	93.33	6.67	0.00
4	BH	0.00	100.00	0.00
5	BH	0.00	100.00	0.00
6	BH	100.00	0.00	0.00
7	BH	68.75	31.25	0.00
8	BH	100.00	0.00	0.00
9	BH	100.00	0.00	0.00
10	BH	100.00	0.00	0.00
11	BH	100.00	0.00	0.00
12	BH	100.00	0.00	0.00
13	BH	100.00	0.00	0.00
14	BH	100.00	0.00	0.00
15	BH	100.00	0.00	0.00
16	BH	43.75	56.25	0.00
17	BH	6.25	93.75	0.00
18	BH	100.00	0.00	0.00
19	BH	93.75	6.25	0.00
20	BH	100.00	0.00	0.00
lq1	BH	53.33	46.67	0.00
lq2	BH	86.67	13.33	0.00
lq3	BH	100.00	0.00	0.00
lq4	BH	100.00	0.00	0.00
lq5	BH	100.00	0.00	0.00
Training files				
BH	BH	100.00	0.00	0.00
FL	FL	0.00	100.00	0.00
JP	JP	75.00	25.00	0.00

4.7 Discussion of Results

There have been many results presented in this chapter for a variety of classification problems and classifier architectures. The salient results are presented in Table 4.13 to allow ease of reference when they are being discussed.

Table 4.13: *Summary of experimental results using the 262 codebook.*

Classification problem	**Network architecture**	**Frames correctly classified (%)**
Urban audio events	MLP 100–6 `trainscg--tansig`	81
	MLP six 100–1 `trainscg-logsig`	80
	LVQ 100–6	52.33
	LVQ six single–output	no result
Tibicen songs	MLP 10–3 `trainscg--tansig`	69
	LVQ 100–3	71.80
	MLP three single–output	no result
	LVQ three single–output	no result

4.7.1 Urban Audio Event Results

For the multiple–output MLP network (100–6 topology — see Figure 4.1(a)), it was found that the conjugate gradient descent backpropagation training function (`trainscg`) with a `tansig` hidden layer transfer function gave the most accurate results with a mean M–value across five training and testing sessions of 0.707. When tested with individual audio files that were previously unseen by the network, 81.2% of frames of audio data were accurately classified.

The classifier structure shown in Figure 4.1(b) was also tested using MLP networks. A set of six single–output MLP networks were trained to identify only one category of audio data each. For this topology (100–1) it was found that the conjugate gradient backpropagation method was again the better training function but was paired with a `logsig` output layer transfer function. Across five training and testing sessions, a mean M–value of 0.681 was achieved. The tests using novel audio data carried out with the multiple–output MLP network was performed using the single output networks. In this test the six 100–1 networks achieved an overall accuracy of 79.7% which is not very different to that of the 100–6 MLP network.

Test results using LVQ networks were not as promising as those using MLP networks. The multiple–output LVQ network only achieved an overall M–value of 0.349 across five training and testing sessions, much lower than the results achieved with the MLP networks. The highest performing 100–6 LVQ network was only able to accurately classify 52% of audio frames from novel data. The poor results for the overall accuracy were due to the LVQ network being unable to correcly classify any frames of data from the *building works* and *rail transport* categories.

Results for tests using six single–output LVQ networks were not retrieved due to very poor preliminary testing results. All of the single–output LVQ networks used during preliminary testing had M–values very close to zero indicating that they were unable to correctly classify any of the input data.

4.7.2 Bioacoustic Results

The results achieved with both the MLP and LVQ multiple–output network structures (as shown in Figure 4.1(a)) are very promising for classifying *Tibicen* cicada songs. The MLP network achieved an overall accuracy of 69.02% and the LVQ network achieved an accuracy of 71.80%. Compared to earlier works of Ohya (2004) and Chesmore (2004), the combination of the 262 codebook with either an MLP or LVQ network offers an improvement in the classification accuracies and ability to work with low–quality recordings.

The MLP network topology with the highest performance had 10 hidden units, used the `tansig` hidden layer transfer function and was trained using the conjugate gradient descent backpropagation algorithm (`trainscg`). With this network topology, 22 out of the 25 test files were correctly classified (using the highest percentage of correct frames as the winning output), 14 of which had 100% of their frames correctly classified.

An LVQ network having 100 hidden units and three outputs gave the highest performance overall for the percentage of correctly classified TDSC frames. However, this network was only able to correctly classify 19 out of the 25 test files. Nevertheless, this is still an improvement over the results reported by Chesmore (2004).

It was observed with the urban audio event classification problem that the single–output MLP networks had similar performance results to the equivalent multiple–output MLP network. However, for the bioacoustic classification problem presented here, neither of the single–output network architectures were able to produce sufficient results in the preliminary tests to warrant further testing being carried out.

It is to be noted that performing post–regression analyses of results was not a reliable method of determining network accuracy with the *Tibicen* recordings. The MATLAB® `postreg` function returned M–values of 0 for output 2 of the networks (*T. flammatus*) for all of the test files. This is because none of the test files belonged to that category so regression analyses could not be performed. The `postreg` function also returned 0 for output three (*T. japonicus*) for many of the test files because only one test file belonged to that category.

4.8 Chapter Summary

In this chapter, results have been presented for the extensive testing of the 262 codebook carried out to determine how well it performs as a feature extractor. Results for both urban audio events and bioacoustic signals were presented. The results for both of these classification problems were promising. These are summarised throughout the chapter and presented in full in Appendix E, showing that the simple S–matrix output of TDSC is suitable to the task in hand.

The results also showed which classification system structures and classifier topologies are best–suited to each classification problem. Multiple–output MLP networks were the classifiers of choice in the majority of cases.

The importance of preliminary testing was demonstrated throughout the work presented in this chapter. When using an MLP network, the difference that topology, training function and even transfer function can make to the final results was evident. Little guidance on network design was found in the literature, and using systematic trial-and-error methods for the preliminary tests allowed the most suitable network architectures to be found.

The next chapter provides a discussion of a completely novel addition to the TDSC arsenal of techniques. Termed "Multiscale–TDSC", the method of sorting and presenting data to the input of networks is inspired by Wavelets and allows a further reduction to the size of the feature set.

Chapter 5

Further Development of TDSC Features

The results presented in Chapter 4 showed that Time-Domain Signal Coding can be applied quite successfully to the classification of urban audio events. The 262 codebook has been shown to be generic enough to allow a wide range of audio signals to be identified. The codebook has also been shown to be capable of discriminating between bioacoustic signals to determine the species of an animal, this being one of the original applications of TDSC.

This chapter discusses development work which attempted to improve the accuracy of the classification system using TDSC as its feature extractor. The work was centred on reducing the size of the output from the TDSC algorithm to further simplify the data input to a classifier. The codebook developments discussed in Chapter 3 focused on finding redundant sectors of the codebook to reduce the amount of data presented to a classifier. The developments discussed in this chapter were inspired by wavelet analysis.

The chapter begins by explaining the codebook developments. The latter part of the chapter presents and discusses the data gathered from carrying out experimental work with the new TDSC features.

5.1 Modifying the Output of TDSC

In Chapter 3 the procedure carried out for codebook modification and reduction was discussed. This process involved a large amount of manual analysis of the codebook to identify any redundancies in the features being generated. Early analyses revealed that most of the D–S pairs (∼90%) for the types of signal being examined did not have a duration larger than 150 samples or a shape larger than 15. Further manual analyses revealed large areas of zero data caused by impossible combinations of duration and shape. The removal of the gaps and the limits placed on the durations and shapes resulted in the 262 codebook being developed.

The output of TDSC analysis using the 262 codebook is an S–matrix for each frame of data. The S–matrix is one of the original representations of D–S pairs as discussed in the work of King and Gosling (1978). The length of each frame of TDSC data is predetermined by the user. The framelengths used in earlier applications of TDSC, such as Chesmore (2001), were chosen dependent on the signal under analysis. The signals studied in these earlier works were very similar in nature and the use of equal framelengths for each signal was not unreasonable.

All of the results presented in Chapter 4 used a fixed framelength of 0.2 seconds during TDSC analysis; an S–matrix based on the 262 codebook was generated for each 0.2 seconds of signal. Whilst this framelength allowed satisfactory classification accuracies for both urban audio signals and bioacoustic signals, the question arose as to whether different framelengths for different signals could improve the achievable accuracies.

A significant hurdle in using a variable framelength for the application of urban signal identification is that the ideal framelength for each signal is unknown because of the stark differences in the signals. It would be unreasonable to expect the system to analyse each signal to find the ideal framelength prior to extracting features for classifying.

The solution to this problem was to set the framelength to a fixed number of epochs rather than a fixed amount of time in seconds. The duration of each epoch will be dependent on the prominent frequency features at any given point in a signal. Therefore, the framelength used during analysis will also vary with the signal.

In deciding how many epochs should be used per frame, consideration was also given to how the extracted data would be best represented and delivered to a classifier. Inspiration was sought from outside the realms of TDSC resulting in wavelet-inspired data representation.

5.1.1 Wavelet–Inspired Data Representation

The wavelet transform was introduced in Chapter 2 as a mathematical microscope which can give an approximate image of a signal, but also zoom in on the smaller details (Hubbard, 1998). Through the work of Bunting (2011), a member of the ISRIE project team working at York, exposure was gained of how wavelets decompose the time–frequency plane. Figure 5.1 shows the dyadic representation of the time–frequency plane which results from wavelet decomposition.

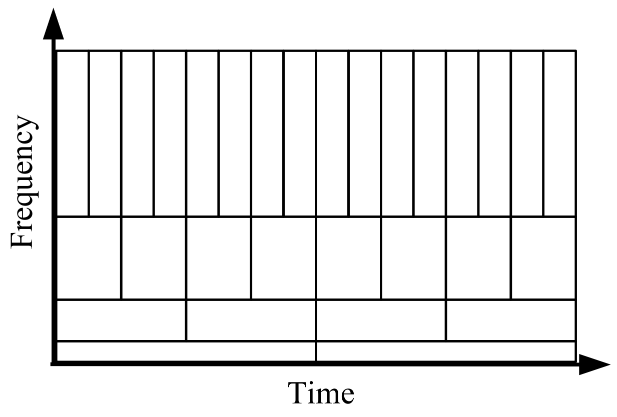


Figure 5.1: *Dyadic representation of an idealized time–frequency plane. Adapted from Hubbard (1998).*

The dyadic representation reveals one of the key features of wavelets: low-frequencies are examined with windows which are precise in frequency but vague about time, and high-frequencies are examined using windows which are vague about frequency but more precise about time. It was decided to mimic this behaviour of wavelets using TDSC data. For each TDSC frame, epochs become grouped together according to their duration.

There is a similarity between wavelets and TDSC in that both analyse signals in a frame–wise manner. The nature of the dyadic wavelet decomposition of a signal shown in Figure 5.1 is such that a frame must have 2^n elements (usually the elements are samples of audio). It was discussed above that no such restriction had yet been applied in TDSC and the framelength had been selected by the user and given in seconds. It was decided to apply the 2^n elements per frame restriction to TDSC analysis, using epochs as the elements, resulting in Multiscale TDSC.

5.2 Multiscale Time–Domain Signal Coding

5.2.1 Dyadic Representation of TDSC Features

Figure 5.2 shows the dyadic representation of one frame of Multiscale TDSC (MTDSC) features. Time is represented on the X–axis and the frame shown comprises 2^n epochs. The Y–axis represents a set of duration levels (discussed further below) with duration reducing as the Y–axis increases. A single frame of MTDSC features was originally termed an *epochlet* when MTDSC analysis was first published (Stammers, 2009). However, a more descriptive term is now used for a frame of MTDSC features - an MTDSC *packet*.

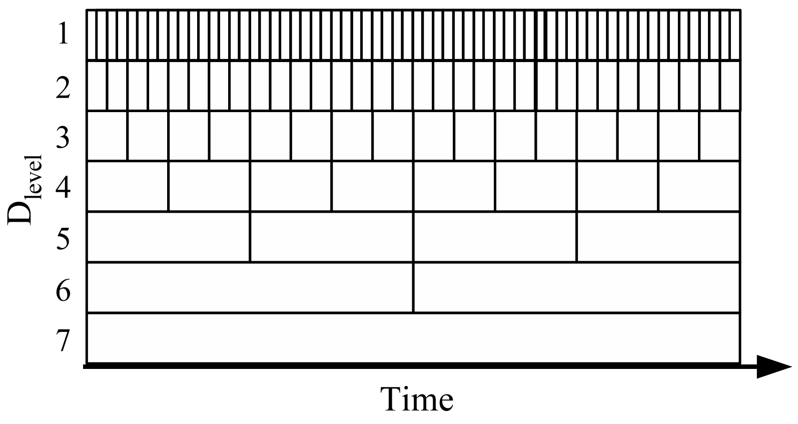


Figure 5.2: *A single frame of dyadic Multiscale TDSC features. Each frame comprises 2^n epochs.*

Each window of the MTDSC packet (the individual boxes in Figure 5.2) contains shape data for at least one epoch. The level of each window on the Y–axis shows which duration level the window contributes towards. The smallest of the boxes represent the shortest duration epochs and will contain data for the fewest number of epochs (2^{n-6}). The largest of the boxes represent epochs with the longest durations and will contain data for 2^n epochs. Therefore, precision in time decreases with duration. This is similar to the wavelet representation of a signal where the precision in time increases as frequency increases.

5.2.2 Constructing an MTDSC Packet

Figure 5.3 show a flow diagram describing the process of constructing an MTDSC packet. The first two stages of the process, finding the epochs and calculating D and S, are identical to the methods used in standard TDSC analysis. However, the order in which the signal is divided into frames and analysed differs in MTDSC, as does the way in which data is represented. Stages 3 to 6 of Figure 5.3 are discussed in further detail below.

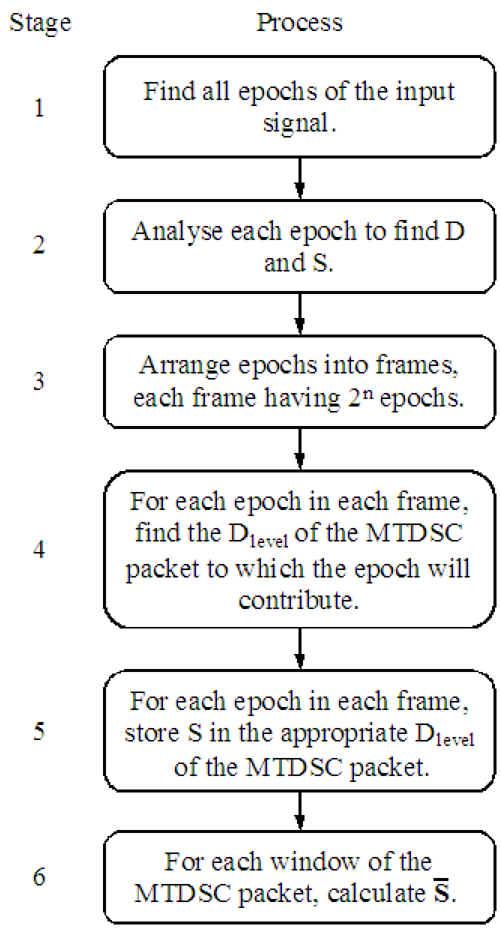

Stage
Process
1
Find all epochs of the input signal.
2
Analyse each epoch to find D and S.
3
Arrange epochs into frames, each frame having 2^n epochs.
4
For each epoch in each frame, find the D_{level} of the MTDSC packet to which the epoch will contribute.
5
For each epoch in each frame, store S in the appropriate D_{level} of the MTDSC packet.
6
For each window of the MTDSC packet, calculate $\bar{S}$.

Figure 5.3: *Flow diagram to show the stages of constructing an MTDSC packet.*

Stage 3: Arrange the epochs into frames

In standard TDSC the input audio signal is divided into frames, each having a length given in seconds. The MTDSC process differs in that the segmenting of the signal occurs after the epochs have been found and the length of each frame is 2^n epochs.

A problem with this process of segmenting the signal is that many signals will not have a number of epochs that is exactly divisible by 2^n. In solving this issue, the following approaches were considered:

1. remove any epochs at the end of the signal that go beyond $X \times 2^n$, where X is a whole number and represents the number of frames in the signal;
2. increase the number of epochs at the end of the signal until the total number of epochs is a multiple of 2^n and set the additional epochs equal to zero; or
3. increase the number of epochs at the end of the signal until the total number of epochs is a multiple of 2^n and set the additional epochs equal to other epochs from the same signal.

Approach *1* was discounted because there is the possibility of losing a large proportion of the signal being analysed. For example, if the number of epochs per frame was set to 512 and a signal had 1023 epochs, using approach *1* would dispose of 511 epochs – nearly half of the signal.

Approach *2* was also discounted as a viable option because there is the possibility that a large proportion of the signal represented by MTDSC packets will consist of zero values. For example, if the number of epochs per frame was again set to 512 and a signal had 513 epochs, the signal would be increased to 1024 epochs using zero values. This would result in nearly half of the signal being equal to zero and would not provide a fair representation of the actual signal.

Therefore, it was decided to use approach *3*. The epochs used to enlarge the signal are taken from the very beginning of the signal. The epochs are essentially wrapped around back to the beginning of the signal. Figure 5.4 illustrates this process.

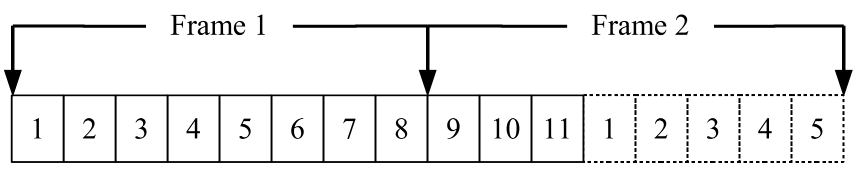


Figure 5.4: *How MTDSC handles signals that do not contain $m(2^n)$ epochs. In this example $m = 2$ and $n = 3$, meaning the signal is required to have 16 epochs. As the signal only contains 11 epochs, epochs from the beginning of the signal are used as filler epochs at the end of the signal.*

Stage 4: Find the D_{level} to which each epoch will contribute

The next stage in the process of constructing an MTDSC packet requires the D_{level} of each epoch to be found. D_{level}s are illustrated in Figure 5.2. As the D_{level} increases, the number of epochs represented by each window reduces. The D_{level} which an epoch contributes towards is determined using Equation 5.1:

$$D_{group} = \lfloor log_2(D) \rfloor \tag{5.1}$$

where $\lfloor\ \rfloor$ denotes an arithmetic floor operation (the mapping of a real number to the largest previous whole number). Using a base–2 logarithm allowed for seven duration levels to be defined – recall that $D_{max} = 150$, giving $\lfloor log_2(150) \rfloor = 7$. Table 5.1 presents the levels and their associated duration ranges.

Table 5.1: *Duration levels and associated duration ranges using base–2 logarithms.*

D_{level}	**Duration range (samples)**	**Number of epochs per window**	**Number of windows per packet**
1	2 – 3	2^{n-6}	64
2	4 – 7	2^{n-5}	32
3	8 – 15	2^{n-4}	16
4	16 – 31	2^{n-3}	8
5	32 – 63	2^{n-2}	4
6	64 – 127	2^{n-1}	2
7	128 –255	2^{n}	1

Table 5.1 also shows the number of epochs which contribute to each window on a given D_{level}. The smallest windows are associated with D_{level} 1 where the shortest duration epochs will contribute to the data. The lowest number of epochs that can be present in a window on D_{level} 1 is one (2^0). Therefore, the lowest number of epochs that can be used to construct a whole MTDSC packet is 64 (2^6). The final column of Table 5.1 shows how many windows will be present in an MTDSC packet on each D_{level}.

Stage 5: Store S in the appropriate D_{level}

After the D_{level} has been found for a given D–S pair, using Equation 5.1, the shape data for that pair can be stored in the appropriate location of the MTDSC packet. This location will be on the calculated D_{level} in the window which encapsulates the position of the epoch under consideration.

Stage 6: For each window, calculate $\bar{S}$

When the data for all of the epochs in a given MTDSC packet has been collated into the packet, the final stage of processing is to find the mean of S for each window, $\bar{S}_{window}$, given by Equation 5.2:

$$\bar{S}_{window} = \frac{\sum^{p} S}{p} \tag{5.2}$$

where p is the number of epochs in the window.

5.3 Implementing MTDSC in MATLAB®

In Chapter 3, standard TDSC was explained and an overview of the functions written in MATLAB® to perform TDSC was given. Some aspects of the analysis were the same for MTDSC, as mentioned above. However, a different set of functions were required to perform the overall MTDSC analysis outlined in Figure 5.3. The functions invoked to construct MTDSC packets for a given signal are shown in Figure 5.5. Full listings for the functions are given in Appendix C. The function `mPacketStackReturn.m` takes the path of a WAV file as its input and returns an MTDSC packet (shortened to *MPacket* in the code listings) which has been stacked to be presented to a network for classifying.

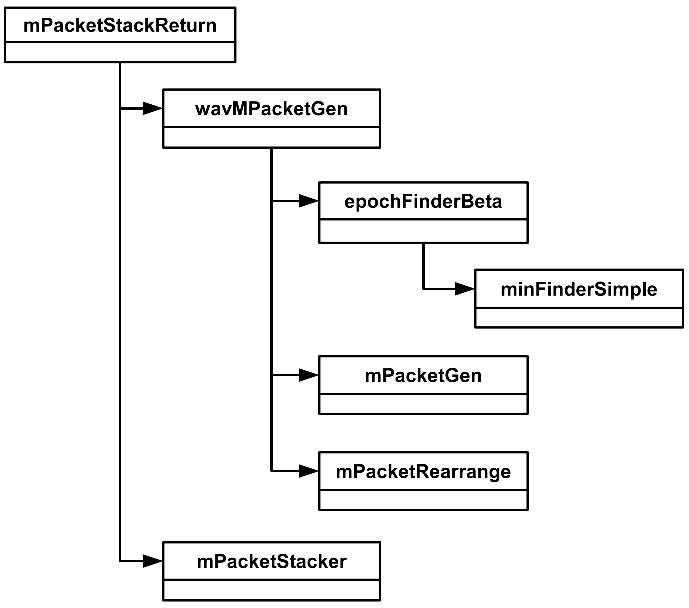


Figure 5.5: *Function dependencies for MTDSC analysis.*

5.3.1 Stacking MTDSC Packets

The input to the classifiers that were used throughout the presented research (MLP and LVQ networks) require an N-by-1 input. Each frame of MTDSC data (an MTDSC packet) is a 7-by-64 matrix and is incorrectly sized for input to the networks being considered. It was therefore necessary to stack the MTDSC data for each frame to give a 127-by-1 matrix. The process of stacking is shown in Figure 5.6.

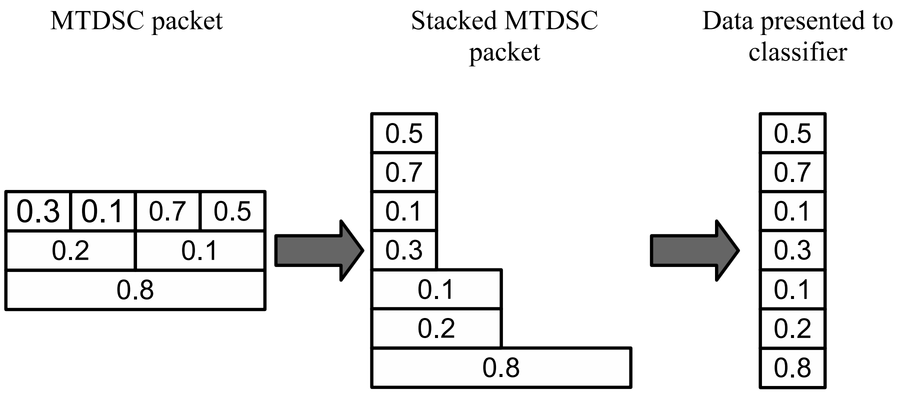


Figure 5.6: *The process of stacking an MTDSC packet. The diagram shows a much simplified MTDSC packet with only 3 D_{levels} to illustrate how stacking is performed.*

5.3.2 MTDSC Examples

To illustrate the capabilities of MTDSC in terms of representing input data, experimental work was carried out using swept–frequency cosine signals. The data was generated using the MATLAB® function `chirp.m` and written to a WAV file ready for input to `mPacketStackReturn.m`.

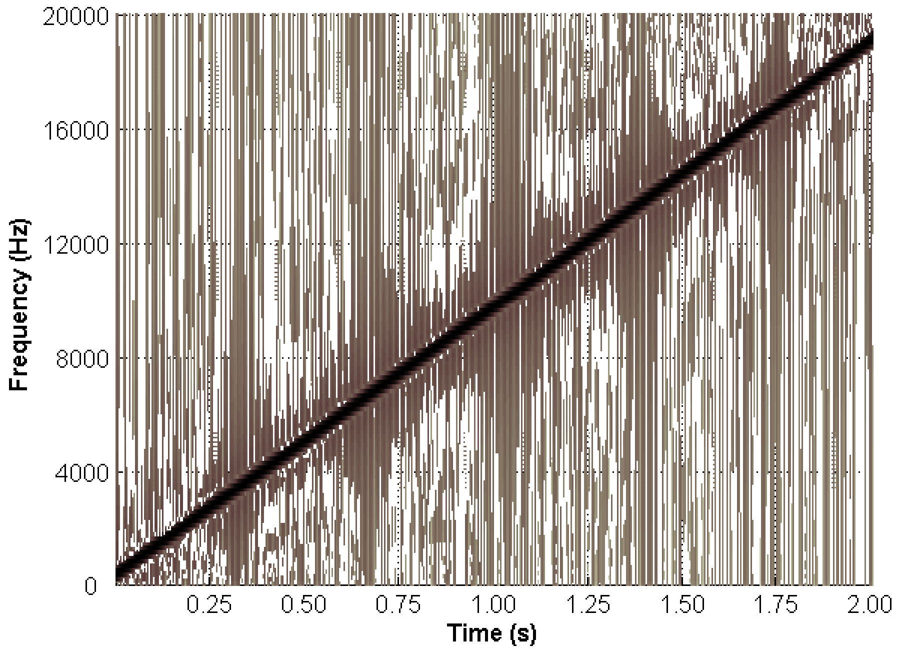
The first cosine generated was a linear sweep from 100Hz to 20kHz. Figure 5.7(a) shows a spectrogram of the signal. Figure 5.7(b) shows the MTDSC packets for each frame of the data. A framelength of 256 epochs was used.

A second cosine was generated to produce a logarithmic sweep from 100Hz to 20kHz. The logarithmic chirp oscillates between the upper and lower frequencies towards the end of the chirp. This can be seen quite clearly in the spectrogram of the sweep in Figure 5.8(a). Figure 5.8(b) shows the MTDSC packets for each frame of the logarithmic chirp. Again, a framelength of 256 epochs was used.

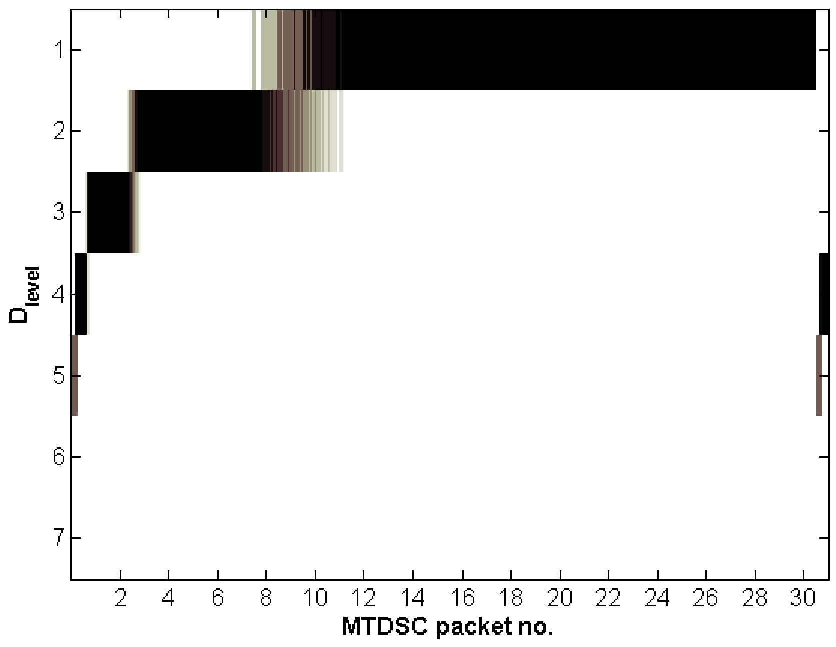
The plots of MTDSC data presented in Figures 5.7 and 5.8 show some of the features discussed above as well as others:

- On the right–most side of the MTDSC plots for both sweeps the repetition of the start epochs can be seen.
- As the frequency of the sweep increases, data is only present in the MTDSC plots at the numerically lower D_{levels}.
- The MTDSC packets can track rapid changes in a signal quite accurately. The fluctuations in the logarithmic sweep are also clearly present in the MTDSC plot.

After seeing the above results, it was decided to test the performance of classification systems using MTDSC features. The process and results of the tests are discussed below.

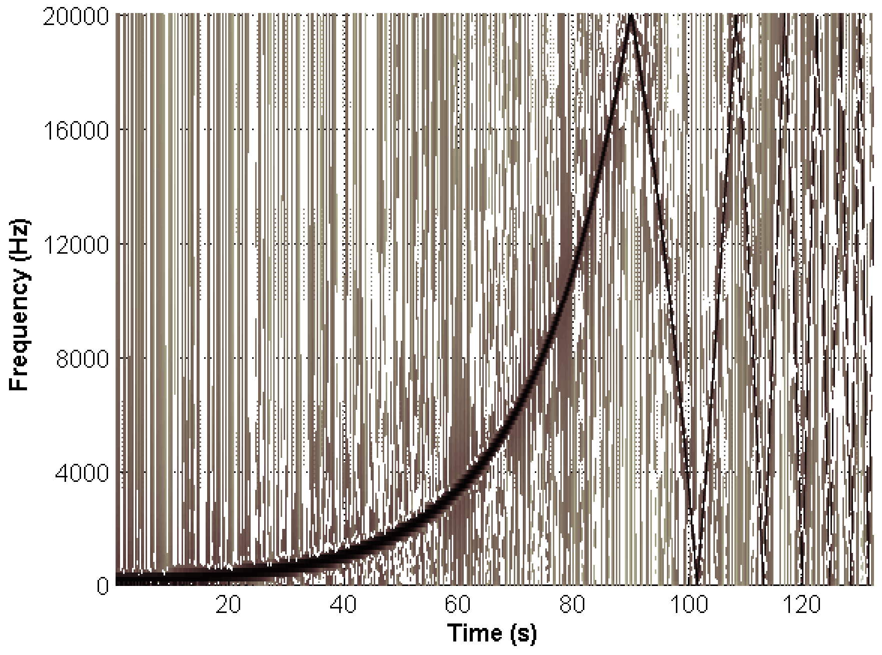


(a) Spectrogram of a linear cosine sweep from 100Hz to 20kHz.

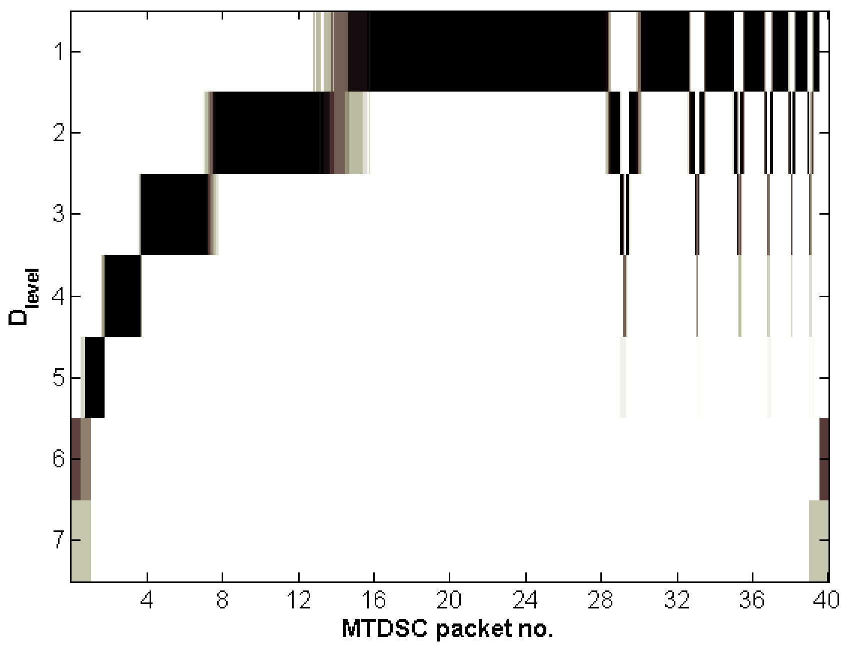


(b) Plot of the MTDSC packets for the linear sweep.

Figure 5.7: *MTDSC analysis of a linear swept–frequency cosine. The shade of the data points corresponds to the value of $\bar{S}$ for each window of the MTDSC packets.*



(a) Spectrogram of a logarithmic cosine sweep from 100Hz to 20kHz.



(b) Plot of the MTDSC packets for the logarithmic sweep.

Figure 5.8: *MTDSC analysis of a linear swept–frequency cosine. The shade of the data points corresponds to the value of $\bar{S}$ for each window of the MTDSC packets.*

5.4 Urban Audio Event Classification with MTDSC

In Chapter 4 results were presented of experimental work using the 262 codebook to generate the TDSC S–matrices. The series of experiments trialled two different classifier structures using two different classifiers (MLP and LVQ networks).

Presented in this section are results from similar experimental work using MTDSC as the feature extractor. For each classifier structure and type, preliminary tests were carried out to determine the optimum classifier topology (i.e. the number of hidden units and, in the case of an MLP network, the training and transfer functions). Summaries of the preliminary results are included here and full sets of results are presented in Appendix F.

The data used for the experimental work is the same data used to generate the results discussed in Chapter 4. Details of the audio data can be found in Appendix A. For the preliminary tests, a reduced data set was used, comprising only the audio from the *air conditioning/ventilation*, *bird* and *road* categories.

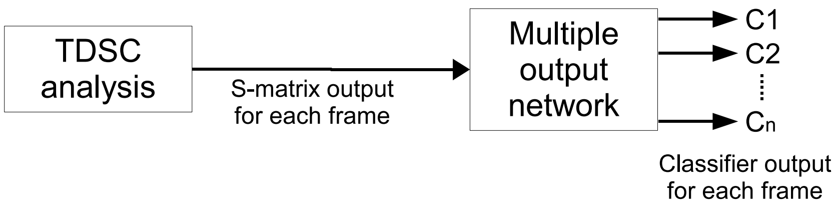
Each network was trained and tested with MTDSC features extracted from the audio data set (either the reduced set or full set). The MATLAB® function `postreg.m` was used to compare the actual simulation results of the trained networks with the target results (where possible[1]). The M–value output of the `postreg` function provides a value indicating how close the simulation results are to the target results.

The two classifier structures that were tested are multiple–output single networks and groups of single–output networks. The design of these two structures can be seen in Figure 5.9 (this is a copy of Figure 4.1 presented again for ease of reference).

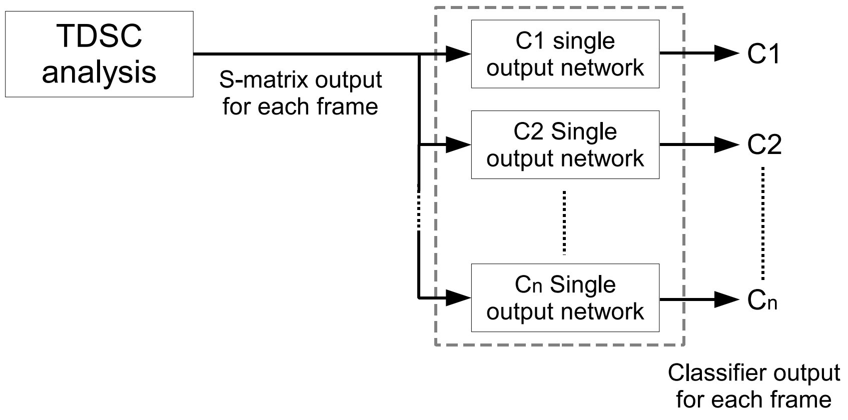
5.4.1 Number of Epochs per MTDSC Packet

Prior to commencing any testing with MTDSC features, it was necessary to determine the ideal number of epochs used to create each MTDSC packet for any given signal. Tests were carried out using five different numbers of epochs: 64 (the lowest number possible), 128, 256, 512 and 1024.

[1]It was found during the experimental work presented in Chapter 4 that the output of the `postreg` function was not always a reliable indicator of network accuracy. This was discussed further in Section 4.7.


TDSC analysis
S-matrix output for each frame
Multiple output network
C1
C2
Cn
Classifier output for each frame

(a) Using a single multiple–output classifier.



(b) Using multiple single–output expert classifiers.

Figure 5.9: *The two classifier structures used during experimental work.*

A 100–3 MLP network was used as the classifier for the tests. The network training function used was `trainscg` and the hidden–layer transfer function was `tansig`. Furthermore, the number of hidden units was 100. This particular combination was chosen because it was the highest performing multiple–output MLP network from the 262 codebook experiments. The data used for training and testing was the reduced set of audio files. Table 5.2 presents the results of the test.

Table 5.2: *Test results to determine the optimum number of epochs per MTDSC packet. The results shown in a bold typeface indicate the highest performance.*

Epochs per M–packet	Time to train (s)	Training epochs completed	M–value mean
64	351.50	556	0.684
128	146.54	445	0.751
256	28.06	119	0.765
512	**10.13**	**99**	**0.819**
1024	4.80	71	0.798

The results in Table 5.2 show that using 512 epochs per MTDSC packet produced the highest M–value mean. Therefore, all of the experimental work discussed below used 512 epochs for each MTDSC packet.

5.4.2 Experiment 1: Multiple–output MLP Network

Using the classifier structure shown in Figure 5.9(a) preliminary tests were carried out using the reduced data set to find the optimal multiple–output MLP network topology. A summary of these results is presented in Table 5.3, with full results available in Appendix F.1.

Table 5.3: *Summary of results for preliminary testing with multiple–output MLP networks and MTDSC data. The entries with a bold typeface indicate the networks with the highest mean M–value.*

Topology	**Training function**	**Hidden–layer transfer function**	**Time to train (s)**	**M–value mean**
10-3	`traingdx`	`tansig`	4.55	0.854
		`logsig`	6.81	0.851
	`trainrp`	`tansig`	2.53	0.831
		`logsig`	1.64	0.815
	`trainscg`	`tansig`	1.60	0.797
		`logsig`	**3.40**	**0.859**
50-3	`traingdx`	`tansig`	8.80	0.811
		`logsig`	16.01	0.825
	`trainrp`	`tansig`	2.28	0.786
		`logsig`	2.32	0.819
	`trainscg`	`tansig`	3.01	0.837
		`logsig`	**6.64**	**0.846**
100-3	`traingdx`	`tansig`	20.50	0.799
		`logsig`	38.36	0.800
	`trainrp`	`tansig`	3.47	0.839
		`logsig`	3.70	0.782
	`trainscg`	`tansig`	8.96	0.500
		`logsig`	**23.04**	**0.846**
300-3	`traingdx`	`tansig`	104.42	0.803
		`logsig`	315.07	0.798
	`trainrp`	`tansig`	19.87	0.802
		`logsig`	11.01	0.830
	`trainscg`	`tansig`	30.65	0.827
		`logsig`	**42.78**	**0.839**

The results in Table 5.3 show that a network having 10 hidden units trained with the `trainscg` algorithm produces the highest M–value mean (0.859). To ensure that there was consistency across each network topology, a mean of the mean M–values was calculated for each topology[2]. This calculation showed that the MLP network having 10 hidden units consistently outperformed the other network topologies.

Based on the results from the preliminary tests, training and testing with the full set of audio data was performed using an MLP network having 10 hidden units. As with the experiments presented in Chapter 4, given the very short time required to train the MLP networks, all six combinations of training function and hidden layer transfer function were tested.

Training and testing was carried out five times in total with new network initialisations for each session. Repeating the training and testing allows a mean accuracy to be calculated. The networks are re-initialised for each session to give different initial weight vectors (these are randomly assigned when a network is initialised). Repeated training and testing sessions is recommended by Demuth et al. (2008).

Mean results across all five session are presented in Table 5.4. The results show that the networks trained with the conjugate gradient descent function `trainscg` with a `logsig` hidden–layer transfer function had the highest overall M–value mean. The result shown in Table 5.4 for this network has two values shown. The value in parentheses (0.557) includes the M–value mean from the first training session, whereas the value shown in bold typeface (0.683) does not.

The M–value mean for the `trainscg--logsig` network in the first training session was 0.053, significantly lower than the M–value means for all of the other training sessions. The result for the first training session was therefore treated as anomalous and not used to calculate the final M–value mean. Full results for all five training and testing sessions can be found in Appendix F.2.

[2]This step was performed after all preliminary tests to determine which network topology gave a consistently high M–value mean. For example, a `traingdx-logsig` 50–3 network might have an M–value mean of 0.9 but the rest of the 50–3 network may have M–value means of 0.7. This would indicate that the 50–3 topology does not consistently perform well.

Table 5.4: *Summary of performance results for 10–6 MLP networks classifying MTDSC features of urban audio data. The training times and M–value means are the averages for five training and testing sessions.*

Training function	**Hidden–layer transfer function**	**Training time (seconds)**	**M–value mean**
`traingdx`	`tansig`	34.55	0.651
	`logsig`	42.35	0.666
`trainrp`	`tansig`	8.05	0.589
	`logsig`	6.96	0.618
`trainscg`	`tansig`	17.19	0.666
	`logsig`	**20.16**	**0.683** (0.557)

To further validate the results, the 10–6 `trainscg--logsig` network trained during session five was used to classify MTDSC data for individual audio files (the network from session five had the highest M–value mean of all the trained networks). The data used for this testing comprised longer audio files than those used for training and testing. The individual audio files did not contain any of the same data that had been used to train the networks. Table 5.5 presents the results of the tests.

Table 5.5: *Individual WAV file classification results for the multiple–output 10–6 MLP network classifying MTDSC features. The fourth column shows how many frames of the audio data were correctly classified as a percentage.*

Sound Category	**File**	**Desired output**	**% correct desired output**
Air transport	`AirTrans002`	1	67
Air conditioning	`AirCon001`	2	95
Bird	`Soundrec036_Bird`	3	63
Building works	`digger_drill`	4	69
Rail transport	`train_idle`	5	73
Road transport	`Soundrec024_Road`	6	55
Mean			70

The results presented in Table 5.5 show that the combination of MTDSC features and an MLP classifier correctly classifies 70% of all data presented to it. Overall, the results for each category are satisfactory with most achieving over 60%. The only category falling below this threshold is *road transport*.

5.4.3 Experiment 2: Multiple Single-output MLP Networks

The results presented in this section relate to using a classifier structure such as that shown in Figure 5.9(b). For this structure, each network is trained as an expert in determining if the presented data belongs to its category or not. The results of all of the single–output networks are compared to determine the winning category.

The results from the preliminary tests are presented in Appendix F.3. The tests showed that the highest performing single–output network across all three preliminary test categories had 10 hidden units. Furthermore, the networks having 10 hidden units performed over all training and transfer functions. Therefore, training and testing of the full audio data set was performed using six single–output MLP networks each having 50 hidden units.

Five training and testing sessions were performed with new network initialisations for each session. All training functions were tested across all sessions because of the very short time required to train the single–output networks. Full results of the sessions can be found in Appendix F.4. A summary of the results is presented in Table 5.6.

Table 5.6: *Summary of results for six 50–1 MLP networks classifying MTDSC features for urban audio data. The M–values given are the mean values across five training and testing sessions.*

Category	Time to train (s)	M–value
Air transport	0.95	0.883
Air conditioning	4.25	0.709
Bird	6.39	0.816
Building works	5.05	0.478
Rail transport	6.97	0.317
Road transport	8.57	0.594

The results show that both the *building works* and *rail transport* categories caused problems for the networks in terms of accuracy. The *air transport*, *air conditioning* and *bird* categories all had consistently high M–values.

Further validation of the trained networks was carried out using the individual WAV files seen in Section 5.4.2. The function `singleOpNetTest.m` (see Appendix C) was used to analyse the outputs of all six networks and determine the winning category.

The results show that overall 65% of the MTDSC frames extracted from the individual WAV files were correctly classified. The network trained to classify audio from the *building works* category had the poorest accuracy with only 44% of MTDSC frames being correctly classified. This correlates with the M–value mean achieved by the network which was also very low. Conversely, the *rail transport* network managed to correctly classify 73% of the MTDSC frames whilst having the lowest M–value mean. This could have resulted from the network developing a bias to one particular type of sound from the *rail transport* category (in this case a train idling at a station).

Table 5.7: *Individual WAV file classification results for the six 50–1 MLP networks classifying MTDSC features. The fourth column shows how many frames of the audio data were correctly classified as a percentage.*

Sound Category	**File**	**Desired output**	**% correct desired output**
Air transport	`AirTrans002`	1	67
Air conditioning	`AirCon001`	2	90
Bird	`Soundrec036_Bird`	3	56
Building works	`digger_drill`	4	44
Rail transport	`train_idle`	5	73
Road transport	`Soundrec024_Road`	6	59
Mean			65

5.4.4 LVQ Network Experiments

Preliminary testing was carried out in the same way as discussed above using LVQ networks as the classifiers instead of MLP networks. The results for both the multiple–output network and the single–output networks are discussed together because the results were, overall, quite poor.

A summary of the preliminary test results is given in Table 5.8 for the multiple–output classifier structure. The networks were trained and tested using the reduced data set.

Table 5.8: *Preliminary results summary for multiple–output LVQ networks classifying MTDSC data.*

Topology	**Time to train (s)**	**M–value mean**
10–3	76.02	0.393
50–3	124.60	0.469
100–3	161.04	0.475
300–3	365.65	0.404

Recalling that an M–value of 1 indicates a perfect match between the simulated data and the target data, the M–value means for all of the network topologies are much lower than desired. Nevertheless, training and testing using the full data set was performed. The 100–3 network had the highest M–value mean during preliminary testing. Therefore, the networks used for classifying during full testing had 100 hidden units. The results of testing using the full data set are presented in Table 5.9.

Table 5.9: *Summary of results for five training and testing sessions using a 100–6 LVQ network to classify MTDSC data.*

Session	**Time to train (s)**	**M–value mean**
1	151.21	0.275
2	170.30	0.338
3	150.29	0.324
4	160.22	0.287
5	148.02	0.290
Mean		0.303

It was expected to see accuracies as low as those presented in Table 5.9 given the low M–value means from the preliminary testing. Outputs 1, 2 and 5 (categories *air transport*, *air conditioning* and *rail transport* respectively) had an M–value mean of zero for all five training and testing sessions.

Individual WAV files were then used to test the combined system using the network from the session with the highest M–value mean (session 2). The mean of the correctly classified frames of MTDSC data for all six of the individual WAV files (listed in Table 5.7) was 37%. This result confirmed the poor performance of the multiple–output LVQ network.

Preliminary testing using single–output LVQ networks classifying MTDSC data produced similar results to those achieved using the 262 codebook. All networks had an M–value mean of the order 1×10^{-15}. No further testing was carried out using single–output LVQ networks based on the performance during the preliminary tests.

5.4.5 MTDSC Urban Audio Event Classification – Results Summary

Experimental results using MTDSC features for urban audio events have been presented in this section. Preliminary testing provided an indication of the network topologies and training/transfer functions that would give the best performance with the full data set of audio files.

Using a multiple–output MLP network with 50 hidden units, a log–sigmoid transfer function and the conjugate gradient descent training algorithm gave the highest accuracy: 70% of correctly classified frames of MTDSC data. Switching to single–output MLP networks (one network per sound category) resulted in a lower overall accuracy of 65%. These are promising results for the classification of features extracted using a completely novel method.

The LVQ networks were less successful in their results. The multiple–output LVQ network (100–6) only managed to correctly classify 37% of the MTDSC frames presented to it. The single–output LVQ networks failed to produce adequate results during preliminary testing to warrant further experimentation.

5.5 Bioacoustic Signal Classification with MTDSC Data

Multiscale TDSC is a novel method of extracting and presenting the time–domain features of a signal, allowing satisfactory classification results with urban audio data. Experimental work continued with MTDSC in applying it to bioacoustic signals to determine if similar success could be had. This section discusses the work carried out. The objective of the bioacoustic testing was to be able to differentiate between the *Tibicen* cicada songs discussed in Chapter 4.

Prior to fully training and testing MLP and LVQ networks to classify the cicada songs, tests were performed to determine the number of epochs per MTDSC packet that would produce the best results. Four network topologies were tested with both MLP and LVQ classifiers (10, 50, 100 and 300 hidden units with three outputs) as well as five different numbers of epoch per MTDSC packet.

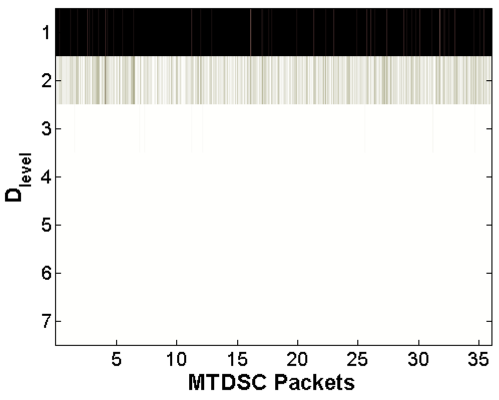
During this very early stage of testing, it became apparent that the classification results using MTDSC features would be very poor. The highest result achieved during the epoch number tests was 7% of MTDSC frames correctly classified (using 1024 epochs per MTDSC frame and a 10–3 `trainscg-tansig` MLP network). Results using single–output and LVQ networks were even lower ($\sim$3.5%). However, both MLP networks (single– and multiple–output) and the multiple–output LVQ network were capable of achieving 100% accuracy for each of the cicada training files.

To investigate why the classification accuracies were so poor, MTDSC packets were visualised for some of the cicada recordings. Figure 5.10 shows the visualised MTDSC packet data for the training files. Figure 5.11 shows visualised plots for four of the test files including one of the very low–quality recordings.

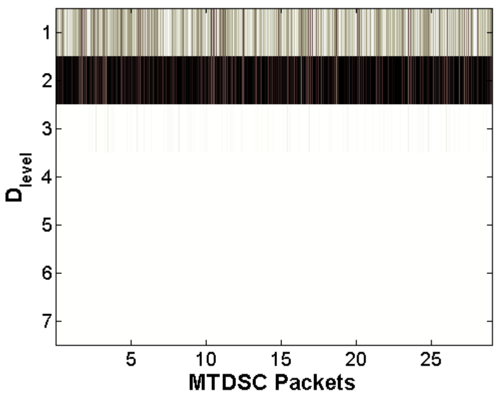
The plots of the training files in Figure 5.10 show clear differences between the signals from the three species of *Tibicen* cicada. The D–S data extracted from *T. bihamatus* is concentrated in the windows of $D_{level} = 1$. For *T. flammatus* $D_{level} = 2$ has the highest concentration of data, and for *T. japonicus* there is an even spread across $D_{level} = 1$ and $D_{level} = 2$. Visually the differences are obvious between the training files. It is therefore unsurprising that the classifiers were able to differentiate between the training files with 100% accuracy.

Conversely, three of the test file plots in Figure 5.11 do not display any obvious allegiance to one particular category. Test files 1, 2 and LQ1 (Figures 5.11(a), 5.11(b) and 5.11(d) respectively) all had classification accuracies of 0%. An accuracy of 25% was achieved with test file 10 which shows a concentration of data on $D_{level} = 1$ in Figure 5.11(c). The plot for test file LQ1 shows why this file was labelled as low–quality. The plot shows significant levels of data on almost all D_{level}s.

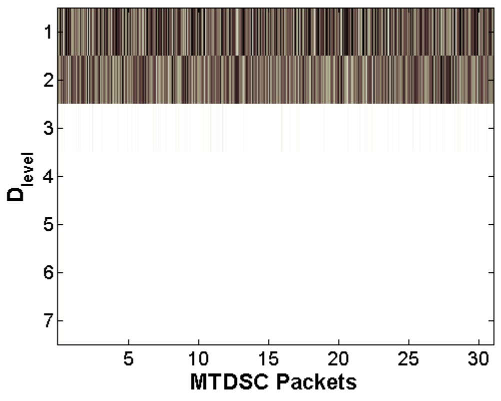
All of the test file plots show data present in the longer duration windows (higher D_{level}s). This suggests that the test files had low frequency artifacts in them. These artifacts had a significant impact on the MTDSC representation of the signals and consequently on the classification accuracies achieved.



(a) *T. bihamatus* training file.

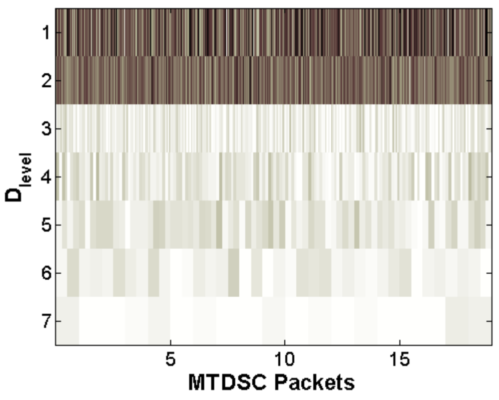


(b) *T. flammatus* training file.

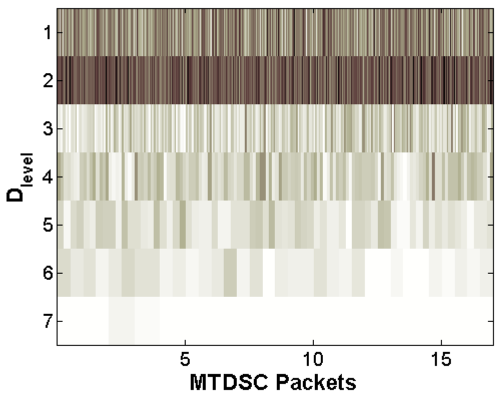


(c) *T. japonicus* training file.

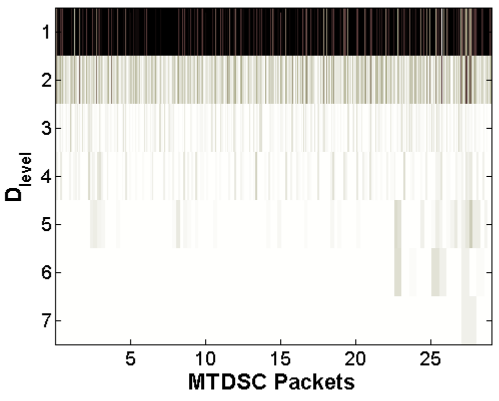
Figure 5.10: *Plots of the MTDSC packets for the* Tibicen *cicada training files.*



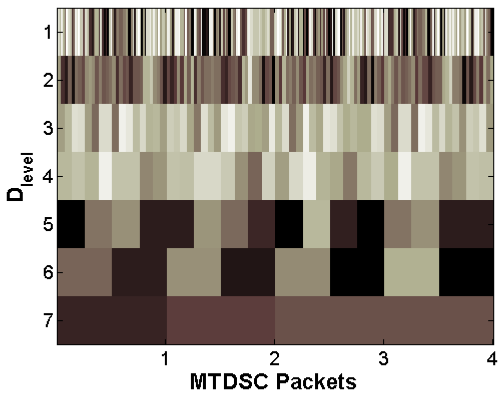
(a) Test file 1 – *T. japonicus*



(b) Test file 2 – *T. bihamatus*



(c) Test file 10 – *T. bihamatus*



(d) Test file LQ1 – *T. bihamatus*

Figure 5.11: *Plots of the MTDSC packets for four of the* Tibicen *cicada test files.*

5.6 Discussion of Results

Results have been presented in this chapter for classification experiments using MTDSC features extracted from both urban audio signals and bioacoustic signals. Table 5.10 presents the salient results of the chapter. Also presented in Table 5.10 is a summary of the results achieved using the 262 codebook to allow comparisons to be made.

Table 5.10: *Summary of experimental results using MTDSC features. Also shown are the results using the 262 codebook for the same classification problems.*

MTDSC features		
Classification problem	**Network architecture**	**Frames correctly classified (%)**
Urban audio events	MLP 50–6 `trainscg-logsig`	70
	MLP six 50–1 `trainscg-tansig`	65
	LVQ 100–6	37
	LVQ six single–output	no result
Tibicen songs	All	no result
262 codebook features		
Classification problem	**Network architecture**	**Frames correctly classified (%)**
Urban audio events	MLP 100–6 `trainscg-tansig`	81
	MLP six 100–1 `trainscg-logsig`	80
	LVQ 100–6	52
	LVQ six single–output	no result
Tibicen songs	MLP 10–3 `trainscg-tansig`	69
	LVQ 100–3	72
	MLP three single–output	no result
	LVQ three single–output	no result

For classification of MTDSC features of urban audio signals, the highest accuracy was achieved with a multiple–output MLP network having 50 hidden units, trained using the conjugate gradient descent training function (`trainscg`) and with a log–sigmoid hidden layer transfer function. This structure of classifier achieved an M–value mean across four of five training and testing sessions of 0.683 (results from one of the sessions were discounted as anomalous). In classifying previously–unseen data from individual WAV files, the network correctly classified 70% of all MTDSC frames extracted from the data.

Using a set of six single–output MLP networks (50–1 topology, `trainscg` training function, `tansig` hidden layer transfer function) achieved an M–value mean of 0.633 across all networks and training sessions. This translated to 65% of all MTDSC frames being correctly classified for the previously–unseen individual WAV files.

Classification with LVQ networks was poor. Using a multiple–output LVQ network (100–6 topology) achieved a mean M–value of 0.303 across five training and testing sessions. Only 37% of MTDSC frames were correctly classified for the individual WAV files. Once again, no results were gathered for single–output LVQ networks due to very poor preliminary results.

For the classification of bioacoustic signals, no significant results were achieved using MTDSC features. The cause of this became apparent when the MTDSC features were visualised in Figures 5.10 and 5.11.

In comparison to the results achieved using the 262 codebook features, the MTDSC results are lower for the urban audio signals and non–existent for the *Tibicen* songs. The results achieved with the 262 codebook were very promising but it was thought they could be improved on with a new approach to generating time–domain data. The results from this chapter have shown that MTDSC features were satisfactory for urban audio signals but not as good as the 262 codebook features.

It is worth noting that the time taken to extract features from a signal using MTDSC is significantly lower than the time taken to extract 262 codebook features. For example, to analyse all 62 of the urban audio training data (each three seconds long) using the 262 codebook takes ∼210 seconds. This reduces to only ∼20 seconds using MTDSC. Furthermore, network training times using MTDSC features are also lower. This benefit of MTDSC could be of relevance to a classification system that needs to be able to learn on–the–fly and cannot spend large amounts of time doing so.

Chapter Summary

In this chapter a new method of representing time–domain features was introduced. Multiscale TDSC is a completely novel feature extraction technique inspired by the dyadic nature of wavelet decomposition of signals. MTDSC allows frames of data to have a length that is dependent on the content of the signal rather than on a framelength pre–selected by the user. Hence it is *multiscale*.

Summaries of results were presented for the classification of urban audio data using MTDSC features for both of the classification structures shown in Figure 5.9. The highest accuracies were achieved using MLP networks and in particular the multiple–output MLP networks. Performance from LVQ networks was poor with the single–output LVQ networks failing to provide any significant results.

Classification of bioacoustic signals was attempted using recordings of *Tibicen* cicada songs. Classification accuracies using MTDSC features were disappointingly low (∼7%) but analysing plots of the MTDSC data revealed why the networks struggled to classify the data.

Comparisons of the MTDSC results and the 262 codebook results were given showing that the 262 codebook features allow for better classification accuracies. However, the feature extraction and training times using MTDSC are significantly lower.

The next chapter draws together the research of this thesis in the discussion and presents some of the possibilities for taking the research further.

Chapter 6

Discussion and Further Work

In this chapter the work that has been carried out is considered and reviewed. Results from the experimental work are discussed and compared with prior work. Suggestions are made for further work that could be conducted following on from the results presented here.

6.1 Signal Classification in Urban Soundscapes

Chapter 2 gave a comprehensive review of the literature related to the field of urban soundscape analysis as well as signal classification. The literature revealed that studying the audio content of the urban sonic environment is not a recent trend but dates back to the 1970s with the work of R. Murray Schafer (1977; 1978). Recent studies of urban soundscapes have focused more on the comfort of the users of the environment and studying the audio content from a psychology perspective (Ge and Hokao, 2005; Yang and Kang, 2005; Cain et al., 2011). It was noted that many of the more recent soundscape studies often used attended monitoring to further evaluate an environment. The proposed ISRIE system would negate the need for laborious analysis of recordings. The classification methods developed in this thesis could provide users with indicators of the audio content of a soundscape, particularly if combined in the proposed ISRIE system.

Prior applications of signal classification were discussed in detail. The focus of the discussion was primarily on recognition of general audio, and some key pieces of research were identified. Given the plethora of alternatives when it comes to feature extractors and classifiers, the work of Cowling and Sitte (2003) was particularly beneficial. The authors provide an excellent overview of many different methods of extracting salient features and how they can be classified. The application of their work was in recognising general sounds for a security system and their methods of training classifiers was very influential during the testing phase of the presented research.

The work of Defréville et al. (2006) was also highly relevant because the authors raise the critical issues of *completeness* and *consistency* that are faced when developing a classification system for such a broad range of sounds. In particular, the *consistency* issue was encountered in the present work when classifying MTDSC features of audio from the *rail transport* category (see Section 5.4.3).

There was little existing work in the field of general audio classification or soundscape analysis that had considered the use of a time–domain zeros–based method of feature extraction. One such technique, time–domain signal coding (TDSC), was selected for further investigation for its prior successes in bioacoustic signal identification (Chesmore, 2001; Farr, 2007). Furthermore, TDSC has very low computation costs, an important factor for the presented research as a contributor to the ISRIE project given the proposed stand–alone portable system. Ghiurcau et al. (2012) used TESPAR (a predecessor to TDSC) for the detection of intruders in wild areas. Ghiurcau et al. chose to test a time–domain method for a stand–alone acoustic sensor specifically because of the simplicity of the feature extraction and the consequently low computational costs.

6.2 Time–Domain Signal Coding

The original applications of TDSC (and time–encoded speech, from which TDSC was derived) only sought to extract features from band–limited signals and required only 29 codes (each code representing a D–S pair). In the case of bioacoustic identification (see Chesmore, 2001; Farr, 2007) this was not an issue as the sounds were limited in their bandwidth already. However, for TDSC to be applicable to more general audio signals it had to be adapted to allow for a wider bandwidth of signal.

The initial version of a *generic codebook* with 1716 codes was very sparse. Low preliminary classification accuracies, combined with network training taking too long, resulted in in–depth analyses of the duration and shape properties of the test data. The analyses revealed that the maximum duration and shape required for most signals was much lower than the 1716 codebook allowed for. The next iteration of the generic codebook had 343 codes. Again, there were redundant codes and the size was further reduced to give the 262 codebook. This reduction resulted in higher classification accuracies during preliminary tests.

The 262 codebook was tested rigorously using different classifiers and different types of audio. The results were very promising with overall accuracies as high as 81% for urban audio signals. The codebook was also used to extract features from bioacoustic signals (*Tibicen* cicada songs) resulting in network accuracies that outperformed previous studies using the same signals (Chesmore, 2004; Ohya, 2004).

To further improve the features extracted using a time–domain method, other ways of representing the data were investigated. Inspiration was gained from wavelets, a technique seen throughout the literature and also used by a colleague. The dyadic representation of wavelet data led to the development of Multiscale TDSC (MTDSC), a completely novel zeros–based feature extraction technique.

The standard TDSC method requires the user to select a length in seconds for each frame of TDSC data. Where the signals are similar, such as in bioacoustic applications, a preselected frame length is not an issue. However, where there are signals from various categories the preselected frame length may not be optimal. MTDSC uses the number of epochs to determine frame length, and the length of each epoch (in samples or seconds) will depend on the signal. Therefore, the frame length can adapt to the signal being analysed.

Classification results using MTDSC features were satisfactory (70%). However, the network accuracies achieved were not as high as the accuracies achieved using the 262 codebook. A probable reason for this is the grouping of durations into only seven D_{level}s. This number of levels was arrived at using Log_2 calculations with the durations and provided a rapid method of reducing the total number of codes. An improvement may be seen if the number of D_{level}s is increased.

Using MTDSC features to classify the *Tibicen* cicada songs exposed a limitation of the MTDSC method. The classifiers were able to distinguish between the training files with 100% accuracy because each category dominated a particular D_{level}. However, all of the test files had significant artifacts appearing in other D_{levels}s leading to classification accuracies of ∼7% (see Figure 5.11, page 126).

The results achieved using both the 262 codebook features (highest = 81%) and the MTDSC features (highest = 70%) are comparable with the results seen in the literature. Cowling and Sitte (2003) report accuracies of 70% using discrete wavelet transform features with a dynamic time warping classifier. In classifying acoustic environments, Dufournet et al. (1998) report an accuracy of 80%. The results reported by Defréville et al. (2006) and Ma et al. (2006) are higher than those of the presented research. However, the feature extraction and/or classification methods used by Defréville et al. and Ma et al. are arguably more complex, and hence more computationally expensive, than those used by the presented research.

It will not have gone unnoticed by the reader that a significant amount of this thesis is dedicated to the development of the feature extractor. This is a reflection on the causative potency that the quality of features can have on classification results. The same network designs were used to classify the features extracted using the 262 codebook and MTDSC, but the results vary significantly between the two. In particular, classification of bioacoustic signals was significantly improved using the 262 codebook features over the MTDSC features. The suggestions made later in this chapter for further work focus on additional development of the MTDSC technique and on experimenting with other classifiers.

6.3 Evaluation of Experimental Work

The focus of the presented research was to develop a classification system capable of determining which category an audio signal belonged to. Much of the development work was concentrated on the feature extraction methods, and it was important to validate how sufficiently each method represented a signal. The experimental work undertaken rigorously tested each of the feature extraction methods, combining them with different classifiers and network designs to determine the optimal system structure.

The methodology for testing each feature extractor and classifier (FEC) combination was to first establish the internal architecture of the classifier. It was found that there existed a paucity of guidance in the literature on methods for designing classifier networks. The focus tended to be on the internal workings and optimisation of the networks rather than where to begin with each classifier. This had not gone unnoticed by other authors, with Rafiq et al. (2001) commenting on the necessity for trial–and–error.

Therefore, preliminary testing was performed for each FEC combination. When using an LVQ network the tests merely established the network topology (i.e. how many hidden units were required). However, in the case of MLP networks, testing was required to determine the ideal training and transfer functions as well as network topology. The preliminary tests provided results allowing the full testing to concentrate on a small number of FEC combinations.

The full tests themselves were devised to provide reliable sets of results, with the training and testing of each network design being repeated five times. This allowed mean results to be calculated and any anomalies to be discarded.

Overall, the results achieved were very satisfactory and, as already mentioned, were comparable with the results presented in previous works by other authors. Using the 262 codebook features produced the highest classification accuracies when combined with a multiple–output MLP network (81%) for the urban audio signals. Using the MTDSC features did not produce as high a result but still achieved an accuracy of 70% when combined with a multiple–output MLP.

In general, the MLP networks outperformed the LVQ networks. There was a tendency for the LVQ networks to anchor onto signals from particular categories, achieving high accuracies for those categories (∼90–99%) but having very poor accuracies for all others. Nevertheless, the results achieved with both the MLP and LVQ networks are very satisfactory because the focus of the research leaned towards the development of novel feature extraction methods rather than classfier development. The classifiers used were tailored slightly to meet the needs of the present application but more robust experimentation and customisation of classifiers could lead to improvements in the results. This is considered further in Section 6.6.

As a system for classifying bioacoustic signals, the LVQ networks out–performed the MLP networks. The combination of the 262 codebook and a multiple–output LVQ network achieved an accuracy of 72% including some very low–quality recordings of the *Tibicen* songs. This result is an improvement over previous studies using the same data (see Chesmore, 2004; Ohya, 2004).

MTDSC features were not very well suited to classifying the `Tibicen` songs. Except for the training files where 100% accuracy was achieved, the highest overall accuracy using MTDSC features was 7%. Visual examination of the MTDSC packets revealed that artifacts in the test signals were very well represented where it would have been better for them to be ignored. This is both a strength and a limitation of MTDSC which is considered more fully in Section 6.5.

The final area of the experimental work that warrants discussion is the urban audio data used for training and testing. Features were extracted from 62 training WAV files and the networks were tested with 20 WAV files to give the network M–values. Further testing was performed with individual WAV files that were not pre–formatted for training or testing.

The accuracies varied across each category and there are a number of reasons for this. Firstly, there is the issue of consistency in the sounds for that category, as highlighted in the literature by Defréville et al. (2006). Secondly, some categories of audio had more training files than others, meaning the networks were exposed to more variations in the possible signals. Both of these points are very closely related and their relationship is discussed in more detail in Section 6.5. Here the issues are considered individually against the classification accuracies achieved with the individual WAV files. Table 6.1 provides a summary for the accuracies achieved with all of the FEC combinations for each category.

To discuss the consistency issue, the variation in the training files must be considered. Details of the recordings are given in Appendix A. The file descriptions show that there were considerable variations within each category thus exposing the classification system to a range of sound files. However, whilst this could provide the system with the ability to recognise more sounds, it also reduces the amount of training data for each particular type of sound in each category. For example, the *Bird* category contains 14 training files providing a mixture of songbirds and waterfowl in varying mixtures and different distances. A network exposed to features extracted from this data set would have a wide and varied experience of possible bird sounds but would have no in–depth experience of a chorus.

With regard to the number of training files for each category and the effect of this on accuracies, Table 6.1 includes a column to show how many training files were used for each category. There does not appear to be any direct correlation between the mean number of frames correctly classified for each category and the number of files used for training. Both the *bird* and *road transport* categories had 14 training files available but their mean accuracies are 60% and 82% respectively. *Air transport* had 6 fewer training files than the *bird* category yet overall had a higher mean accuracy. Of course, the consistency issue re–enters the discourse here because the *air transport* recordings were arguably more consistent than the *bird* recordings.

Table 6.1: *Individual WAV file classification accuracies for all feature extraction and classifier combinations. To denote whether the network had multiple–outputs or if a set of single–output networks were used, the abbreviations* mo *and* so *are used respectively.*

Category	**Number of files**	**Mean % correct frames**	**262 codebook**			**MTDSC**		
			MLP mo	**MLP so**	**LVQ mo**	**MLP mo**	**MLP so**	**LVQ mo**
Air transport	8	67	90	87	90	67	67	0
Air conditioning	10	70	73	73	87	95	90	0
Bird	14	60	81	85	38	63	56	38
Building works	9	67	100	97	0	69	44	91
Rail transport	7	41	50	48	0	73	73	0
Road transport	14	82	93	88	99	55	59	95

In summary, the experimental work was sufficient in testing the FEC combinations and determining which of these were suitable for the applications considered. It would be a jactitation to say that there are no avenues for improvement in the experimental work, and some of the possibilities have been discussed.

6.4 Satisfaction of Project Aims

In Chapter 1 the project aims were stated as follows:

1. Perform a comprehensive review of the literature to identify classification systems that already exist and where contributions can be made to the field of signal classification.
2. Develop and test a classification system suitable for identifying audio signals and determining the category to which the signals belong.

The first aim was satisfied by the literature review presented in Chapter 2. Many fields were examined to discover how signal classification had been implemented, and the techniques used therein were discussed. Inspiration was eventually drawn from bioacoustic signal analysis. TDSC was identified as a feature extraction technique that had not been used widely outside of bioacoustics but had the potential for application in the presented research.

The work detailed in Chapters 3 to 5 fulfilled the requirements of the second aim. TDSC was developed beyond its original realm and applied to the classification of urban audio as well as being tested against a bioacoustic problem. The basic zero–crossings principle of TDSC was further advanced resulting in the completely novel approach to signal representation that was Multiscale TDSC.

6.5 Strengths and Limitations of the Research

The most significant outcome of this research is the development of a system that is capable of distinguishing between sounds belonging to six different categories of urban sounds. The 81% accuracy achieved using the 262 codebook and an MLP network is a very satisfactory result.

The case study presented in Chapter 1 introduced the need for a robust system capable of categorising audio events for the purposes of town planning. The analysis methods developed and tested in the presented research could be applied in the processes outlined in the case study. A classification system capable of deciding if a sonic event was caused by transportation noise or from other sources would negate the need for manual analysis of recordings – so called *attended monitoring*. It is believed that the 262 codebook and an MLP network would contribute towards such a system.

A further important outcome of the presented research is the development of a completely novel zero–crossings based feature extractor, MTDSC. Using MTDSC features, 70% of frames were correctly classified for urban audio events, a result which is comparable with prior research in the field. The technique of constructing MTDSC features lends itself to further development. Shape information is currently the only descriptor used for generating the MTDSC packets. It would be possible to include other signal descriptors in the windows of the MTDSC packet, such as the signal energy of each epoch or the relative amplitude of the positive and negative minima which form the shape information.

When used to extract features from the *Tibicen* recordings, a limitation of the MTDSC method was observed. Anomalies in the signal, such as noise or overspill from other sources, become distinct features of the MTDSC packet. As a result it was only the MTDSC features from the clean *Tibicen* recordings that were correctly classified. The aim of the classification problem was to determine if the test signals belonged to one particular species and in this task the MTDSC features failed. However, if the classification task was to check that a signal was as clean as possible with no artifacts or overspill, MTDSC features could be an excellent choice. Fault diagnosis classification tasks, either in machines or in human health, is an example of an application for MTDSC. If all is well with a system, then the MTDSC features of the monitored signals should be identical to those with which the classifier has been trained. If a fault occurs resulting in an anomaly in the monitored signal, this would be detected in the MTDSC features.

Both of the techniques developed in the presented research, the 262 codebook and MTDSC, are computationally lightweight methods. There is no requirement to transform to another domain when the signals are presented to the system as time–domain waveforms. It was also noted that feature extraction using MTDSC is particularly fast – ∼22 seconds to analyse 62 three–second WAV files.

It was briefly mentioned in Section 6.3 that the presented research would have benefitted from having a larger library of audio data for training and testing purposes. The breadth of the audio data could also have been expanded to further encompass each of the categories of urban audio. Having more training files with which to train networks should result in the networks having a high accuracy for a broader range of audio events.

However, there is the possibility that having more training files encompassing a broader range of sounds could lead to a reduction in classification accuracy. The trained networks may become too generic and only able to classify a large number of sounds with a low accuracy as opposed to a lower number of sounds with a high accuracy. An intricate combination of classifiers could overcome this problem, with each being trained to classify one particular sound or group of sounds into their associated category.

The number of categories of urban sounds used in the presented research was adequate for the aims of the project. The categories arose from meetings with collaborators on the ISRIE project and fulfilled the requirements outlined by the project. It is not thought that the presented research would have benefitted from having many more categories of sound for the urban audio event classification tasks, with the exception of an *alarm/siren* class to encompass security alarms and sirens from emergency vehicles.

The presented research provided a proof of concept of applying time–domain based methods of feature extraction to classifying urban audio events. This was achieved using only two types of classifier: MLP networks and LVQ networks. Experimentation with other classifiers could have revealed a more robust feature extractor and classifier combination resulting in higher classification accuracies. Some of the possibilities for further research in this field are commented on in Section 6.6.

A review of the options available for training MLP networks using the MATLAB® Neural Network Toolbox (NNT) was undertaken and a summary is presented in Appendix D. The preliminary tests carried out with varying combinations of training function and hidden layer transfer function were a necessity of the experimental work in determining the design of the MLP networks. Although an unintended outcome of the research, it is hoped that the summary and preliminary test methodology will be of use to future researchers in deciding which MLP designs to investigate with the NNT.

6.6 Further Work and Future Applications

This chapter has discussed the presented research, highlighting the salient areas of the project. The strengths and limitations of the work have also been considered. This section will present suggestions for taking the project forward and provide some ideas for future applications of an identification tool as proposed by the ISRIE project.

6.6.1 On–Screen Analysis Tool

At present, analysis of an audio signal or batch of audio signals must be performed manually from the MATLAB® command line. Controlling functions are provided to minimise the amount of manual function calling, but it is not an ideal system for other users. Therefore, it is envisaged that the next stage for this research would be to develop a graphical user interface (GUI) which allows users for whom the MATLAB® environment is a foreign land to control the analysis of signals using the techniques developed in the presented research.

In the first instance, such a GUI could be produced using the MATLAB® GUI Development Environment tool (GUIDE). A user interface constructed with GUIDE would have direct access to the functions that have been developed in MATLAB® already. The GUI could have drop–down menus and selection boxes, which will be familiar to all PC users, for selecting the type of analysis to use and the network for classification. A library of pre–trained networks could be provided as well as the option of training (or re–training) a network with new data. The networks could be trained for specific classification tasks such as urban audio event identification or bioacoustic signal analysis.

Farr (2007) developed an on–screen analysis tool such as that described above. Referred to as the *Waveform Analyser and Sound Profiler*, or *WASP*, the tool allowed the uploading of signals for analysis and identification. *WASP* was developed using GUIDE, exploiting the option of keeping the analysis functionality in the MATLAB® environment.

Developing a screen–based tool would also move the project towards the ISRIE goal of a stand–alone intelligent noise meter, as the concepts applied in a software environment can often be adapted to work on a hardware platform.

6.6.2 Network Retraining

In any classification system there will be occasions where the classifier cannot make an accurate decision on the category of a sound (e.g. all six outputs of a 100–6 MLP have a value of 0.167). The reasons for this indecision are numerous: the signal could have a very low signal–to–noise ratio, there may be over–spill from other sources, or the sound may be completely new to the network.

When such a signal is encountered, rather than let the system misclassify or effectively ignore the signal, an option could be included for the system to make a recording of the signal, or flag the signal if it is part of a pre–recorded set, for human intervention. The signal can then be identified by a user and used to retrain the classifier. Consequently, if a similar signal is encountered again the classifier will be better equipped to accurately recognise it.

The proposal outlined above would be well suited to the on–screen analysis tool discussed previously. An option for retraining could easily be built into the GUI allowing the user to quickly retrain a network with any misclassified signals.

6.6.3 Classifier Design

Two types of classifier were investigated in the presented research project and these have been discussed in some detail already. It has also been mentioned that the research would benefit from experimental work using types of classifier beyond MLP and LVQ networks. There is a myriad of networks to choose from, some having only subtle distinctions between them with others employing completely different methods of analysis.

Another option for investigating classifier design is to consider a hierarchical approach. That is, the classification task is broken down into stages, with each stage having a classifier that is designed to choose between a small number of outputs. Defréville et al. (2006) discuss this concept in relation to the *completeness* issue (i.e. the difficulty of designing descriptors to differentiate between every sound category) and suggest that hierarchical classification is one possible solution. A full investigation into hierarchical classification of urban audio events would need to analyse the possibility of using different types of classifier (e.g. MLP, LVQ, etc.) for each stage. It may be the case that certain types of classifier are better suited to differentiating between particular categories.

In their discussion on network optimisation, Hansen and Salamon (1990) propose the use of network *ensembles*. The authors highlight that standard practice when performing a classification task is to find a suitable architecture and tuning of a network *"... and then trust all future classifications to the best network we find."* This was certainly the approach adopted in the presented research and the approach seen in much of the literature. Hansen and Salamon (1990) explain how the set of networks developed during the trial–and–error phase of development should be kept and all used to produce a classification for a given input. A consensus from all of the networks is then used to reach a final decision. This method of classification could be further adapted to again use a variety of network types (rather than just the trial–and–error rejects) and a consensus decision for each of the presented data.

6.6.4 Real–world Applications

All of the further work discussed so far has related to immediate advancements that could be made to the presented research. The suggestions detailed here look much further into the future and encompass the overall vision of the ISRIE project: to develop an intelligent noise monitoring system. The classification methods developed in this thesis would have their place in any such monitoring system because it has been shown that the methods are capable of accurately classifying a variety of signals.

The ISRIE project proposed to develop a system capable of informing the user of the content and nature of the sounds heard in a particular environment. The focus was on the urban environment and sounds relating to PPG 24 and BS 4142. The following proposal builds on this idea to include other potential areas where an identification tool could be used, and is illustrated by way of a fictional case study.

Airport expansion is a pertinent current affair at present. One of the arguments against expansion is the increase in traffic noise, both air traffic and road traffic. Tonshude airport received the backing of local residents by promising that there would be no increase in noise levels as a result of its expansion, neither during nor after the building process. To ensure accurate records were kept of the noise levels before, during and after the expansion process, a number of intelligent noise monitoring devices (INMDs) were installed in the area surrounding the airport.

The INMDs had the capability to identify the signals they detected and place the source of the signal into one of a number of categories. The devices could also estimate the direction of arrival of each signal source. The INMDs could communicate with each other wirelessly thus allowing triangulation of the direction of arrival data to provide source localisation of a signal. The wireless communication also allowed the classification and source localisation results to be collated on a central server. Classification accuracies were compared automatically on the central server to arrive at a consensus decision for the audio content of the soundscape. The data on the server was analysed periodically by a sound consultant whose role included manual classification of the few signals that could not be classified by the system.

After the airport expansion was completed the INMDs remained in place

> *to monitor the impact of the expansion on air and road transport noises. The data generated during the expansion was compiled into a report which presented the contributions to the overall noise level of the different sound categories.*
>
> *At a later date, the planes taking off from Tonshude airport started having problems with bird strikes, resulting in many emergency landings. To determine the best method of controlling the bird populations, the INMDs were reprogrammed, via the wireless network, to detect the species of bird present in the vicinity of the airport. The classifiers in the INMDs were replaced with networks that had been pre–trained with a selection of bird calls from species often associated with the land surrounding airports.*

This fictional case study illustrates how a system such as ISRIE could be used in the future. All of the elements described do not yet exist but the vision of ISRIE is captured in the example.

Chapter 7

Conclusions

The presented research was focused on developing a classification system capable of differentiating between urban audio events. The most significant developments were made in how features were extracted from a signal and presented to classifiers. Classification accuracies of 82% were achieved using the developed feature extraction methods. The research made valid contributions to the field of signal classification on its own but also collaborated with other researchers in a larger project in developing an intelligent noise monitoring system.

A review of the literature revealed the vast array of signal classification techniques that have been used in other works. The review was not confined to prior works classifying the types of signal present in an urban environment. Inspiration was sought from many other fields including bioacoustic species recognition. Time–domain signal coding (TDSC), a purely time–domain feature extraction method, was identified as a candidate for further investigation within the scope of the presented research. It was chosen because it had been used successfully in bioacoustics and was presented as a computationally light–weight method. There was no indication in the literature that a similar method of feature extraction had been applied in the realm of general audio classification. Therefore, by investigating and developing TDSC further, the presented research has made a contribution to the field of general audio classification.

Initially, the techniques involved in extracting TDSC features were adapted to allow analysis of more generic signals. Previously, TDSC had only been used with bandlimited signals and the codebooks used for that were not sufficient to represent the broadband signals present in an urban environment. After two initial iterations of generic codebooks, the 262 codebook was developed. The features produced using the 262 codebook comprised duration and shape information about the epochs of a signal. Despite its heavily reduced size (from 1716 codes initially), the 262 codebook encapsulated $\sim$93% of the total D–S pair combinations possible in the urban audio data analysed.

The 262 codebook underwent rigorous testing to determine how well it represented the input signals. In doing this, the 262 codebook feature extractor was combined with both a multilayer perceptron (MLP) network and a learning vector quantisation (LVQ) network. In the process of testing the 262 codebook, the intricacies of MLP design using the MATLAB® Neural Network Toolbox were established and a summary of the all possible training functions was produced.

Classification accuracies of urban audio events into six categories (*air transport*, *air conditioning*, *bird*, *building works*, *rail transport* and *road transport*) reached as high as 82% when the 262 codebook features were classified with a multiple–output MLP network. The generic 262 codebook was also used to extract features in a bioacoustic classification problem. This task aimed to identify recordings of *Tibicen* cicada songs recorded in environments with varying amounts of background noise. The system achieved an accuracy of 72% with an LVQ classifier. This result was a distinct improvement over previous studies using the same audio data.

The results from the testing showed that TDSC using the 262 codebook is a satisfatory feature extraction method for the classification of urban audio events. Despite the generic nature of the 262 codebook, it proved to also be an excellent feature extractor for bioacoustic signals.

Further investigation of the basic TDSC method was influenced by the dyadic wavelet decomposition of a signal. Multiscale TDSC was developed as a completely novel zero–crossings based method of signal analysis. Classification results for urban audio events were good with accuracies of 70% achieved in combination with an MLP classifier.

Bioacoustic signal classification using MTDSC features produced very low accuracies for the test recordings. This was due to there being significant artifacts present in the test recordings. This result suggests that MTDSC would not be a suitable classifier of signals that have a low signal–to–noise ratio. However, this discovery led to the conclusion that MTDSC would be suitable in fault diagnosis systems because it can accentuate artifacts in the signals that should not be there.

Of the feature extraction methods developed through the presented research, the 262 codebook showed the most promise as a feature extractor suitable for classifying urban sound events and bioacoustic signals.

Recommendations for further work were centred on expanding the training data available, to make the classifiers more accurate, and on experimentation with other classifier designs. The case study outlined in the further work, although fictional, provides a good indication of how signal classification equipment, such as ISRIE, could be used in the future.

It is considered unlikely that a computer–based system will be able to work completely independent of a human observer in the near future in identifying the entire gamut of sounds present in a soundscape. Nevertheless, the methods developed and tested in this thesis contribute towards the evolution of a system which lessens the need for laborious and time–consuming manual analysis of recorded signals.

Appendix A

Audio Data Summary

A.1 Urban Audio Training Data Summary

Table A.1: *Summary of the urban audio files used for training the classification system configurations.*

File	Description
3secAir001.wav	Light aircraft in distance, some birds
3secAir002.wav	Light aircraft passing overhead
3secAir003.wav	Helicopter in distance
3secAir004.wav	Helicopter in distance
3secAir005.wav	Light aircraft high revs
3secAir006.wav	Light aircraft with birds in foreground
3secAir007.wav	Light aircraft (Doppler pitch change)
3secAir008.wav	Distant heavy aircraft
3secAirCon001.wav	Close mic A/C unit
3secAirCon002.wav	Close mic A/C unit w/ bird tweet
3secAirCon003.wav	Close mic fan heater
3secAirCon004.wav	Close mic A/C unit
3secAirCon005.wav	Close mic A/C unit
3secAirCon006.wav	Close mic acoustic louvres
3secAirCon007.wav	Close mic acoustic louvres
3secAirCon008.wav	Distant A/C with some bird calls
3secAirCon009.wav	Distant A/C
3secAirCon0010.wav	Close mic clean room air supply
3secBird001.wav	Mixture of different birds, some background road noise
3secBird002.wav	Blackbird singing
3secBird003.wav	Mixture of birds singing with light aircraft in background

continued on next page...

Table A.1

. . . continued from previous page

File	Description
3secBird004.wav	Mixture of birds singing
3secBird005.wav	Blackbird singing
3secBird006.wav	Mixture of singing birds and Barnacle geese
3secBird007.wav	Greylag geese
3secBird008.wav	Barnacle geese close
3secBird009.wav	Canadian geese flying by
3secBird010.wav	Coot, distant with reverb
3secBird011.wav	Ducks quacking quietly
3secBird012.wav	Ducks quacking quietly with adolescent ducks
3secBird013.wav	Bird tweeting, close mic
3secBird014.wav	Chorus of birds
3secBuilding001.wav	Distant digger
3secBuilding002.wav	Close by digger with scoop movement
3secBuilding003.wav	Close by digger
3secBuilding004.wav	Close by pneumatic drill
3secBuilding005.wav	Close by angle grinder
3secBuilding006.wav	Close by angle grinder with other site noises
3secBuilding007.wav	Distant digger
3secBuilding008.wav	Digger pneumatic drill startup
3secBuilding009.wav	Digger pneumatic drill
3secRail001.wav	Fast train approaching
3secRail002.wav	Fast train passing
3secRail003.wav	Fast train engine passing
3secRail004.wav	Fast train passing
3secRail005.wav	Fast train passing distant
3secRail006.wav	Train idling at station
3secRail007.wav	Train revving at station
3secRoad001.wav	Car passing
3secRoad002.wav	General road noise
3secRoad003.wav	Motorbike revving
3secRoad004.wav	Tractor approaching
3secRoad005.wav	Truck pulling away
3secRoad006.wav	Car passing
3secRoad007.wav	Car passing (distant)
3secRoad008.wav	Low frequency vehicle passing
3secRoad009.wav	Bus passing
3secRoad010.wav	Dual carriageway/motorway
3secRoad011.wav	Busy dual carriageway/motorway

continued on next page. . .

Table A.1

. . . continued from previous page

File	Description
3secRoad012.wav	Car passing quite fast
3secRoad013.wav	Distant traffic noise
3secRoad014.wav	Distant traffic noise

A.2 Urban Audio Test Data Summary

Table A.2: *Summary of the urban audio files used for testing the classification system configurations.*

File	Description
3secTestAir001.wav	Light aircraft in distance
3secTestAir002.wav	Helicopter in distance
3secTestAir003.wav	Light aircraft with birds in foreground
3secTestAir004.wav	Distant heavy aircraft
3secTestAirCon001.wav	Close mic A/C unit
3secTestAirCon002.wav	Close mic fan heater
3secTestAirCon003.wav	Distant A/C unit
3secBird001.wav	Chorus of birds
3secBird002.wav	Lakeside recording of a mixture of water fowl and tweeting birds
3secBird003.wav	Blackbird singing
3secTestBuilding001.wav	Close by digger
3secTestBuilding002.wav	Angle grinder
3secTestBuilding003.wav	Digger using large pneumatic drill
3secTestRail001.wav	Fast train passing
3secTestRail002.wav	Fast train approach
3secTestRail003.wav	Train idling at station
3secTestRoad001.wav	Large vehicle passing, some low frequencies
3secTestRoad002.wav	Dual carriageway/motorway
3secTestRoad003.wav	Car passing quite fast
3secTestRoad004.wav	Distant traffic noise

A.3 *Tibicen* Cicada Audio Data Summary

Table A.3: *Summary of the* Tibicen *cicada audio files used for training and testing the classification system configurations.*

File	**Description**
Training files	
BH.wav	*T. bihamatus*
FL.wav	*T. flammatus*
JP.wav	*T. japonicus*
Test files	
1.wav	*T. japonicus*
2.wav	*T. bihamatus*
3.wav	*T. bihamatus*
4.wav	*T. bihamatus*
5.wav	*T. bihamatus*
6.wav	*T. bihamatus*
7.wav	*T. bihamatus*
8.wav	*T. bihamatus*
9.wav	*T. bihamatus*
10.wav	*T. bihamatus*
11.wav	*T. bihamatus*
12.wav	*T. bihamatus*
13.wav	*T. bihamatus*
14.wav	*T. bihamatus*
15.wav	*T. bihamatus*
16.wav	*T. bihamatus*
17.wav	*T. bihamatus*
18.wav	*T. bihamatus*
19.wav	*T. bihamatus*
20.wav	*T. bihamatus*
lq1.wav	*T. bihamatus* (low quality)
lq2.wav	*T. bihamatus* (low quality)
lq3.wav	*T. bihamatus* (low quality)
lq4.wav	*T. bihamatus* (low quality)
lq5.wav	*T. bihamatus* (low quality)

Appendix B

Duration Ranges Used in Initial TDSC Analysis

Table B.1: *Table showing the ranges of durations used during initial TDSC analysis with the 1716 codebook.*

Duration Range	Durations Included	Duration Range	Durations Included
1	2	21	31–40
2	3	22	41–50
3	4	23	51–60
4	5	24	61–70
5	6	25	71–80
6	7	26	81–90
7	8	27	91–100
8	9	28	101–150
9	10	29	151–200
10	11	30	201–250
11	12	31	251–300
12	13	32	301–350
13	14	33	351–400
14	15	34	401–450
15	16	35	451–500
16	17	36	501–600
17	18	37	601–700
18	19	38	701–800
19	20	39	801–900
20	21–30	40	901–1000

Appendix C

MATLAB® Code Listings

C.1 Standard TDSC Analysis Code

sMatReturn.m

```
function [datasmat] = sMatReturn(wavfile,frameLength)

% Created: 23rd April 2008
% This function will perform TDSC analysis, retrieve the ...
    s−matrix and
% return it.

% Tdsc analysis
Data = tdscAnEps(wavfile,frameLength,1,1);

% Extract s−matrix
for i=1:length(Data)
    datasmat(:,i) = Data(i).sMat(:,3);
end
```

tdscAnEps.m

```
function [Frames] = tdscAnEps(wavFile,L,type,dRange)

% Controlling function to perform TDSC analysis of a WAV file ...
    to be used
% for testing.
% WAV file is broken into L−second segments of data for ...
    analysis. Each
% L−second lump, or 'frame', is stored in the data structure ...
    Frames.
% dRange is used for codebook generation and determines the ...
```

```
    duration range
% for each code (1-5,6-10,11-15,etc.)

% Read in WAV file - separate variables for data, sample rate and
% bit-depth.
[input,inFs,inB] = wavread(wavFile);

% Find length of file in L-second segments. If file is less ...
    than 1 L-second
% lump, increase it's size to L*Fs. This should avoid any ...
    errors later
% involving non-whole numbers.
[inR,inC] = size(input);
if L*inFs ≤ inR
    numberL = ceil(inR/(L*inFs));
else
    input(inR:inFs,:) = 0;
    numberL = 1;
end

% Predefine data structure for epoch storage

Frames = struct('Data',{},'sMat',{});
% Perform epoch analysis on each L-second segment of the WAV data.
for i=1:numberL
    Epochs = struct();
    tempS = ceil((i*L*inFs-(L*inFs-1)));
    tempF = ceil(i*L*inFs);
    % If tempF is greater than the length of the input data ...
        increase the
    % size of the input data by filling it to tempF with zeros.
    if inR < tempF
        input(inR+1 : tempF,:) = 0;
    end

    tempDat = input(tempS:tempF,:);
    dat = tempDat';

    [Epochs,unused] = epochFinderBeta(dat,type);

    % Add current epoch data to Frames IF current epoch data ...
        contains
    % information.
    if size(Epochs,1) ≠ 0
        Frames(size(Frames,2)+1).Data = Epochs;
    end
end

% Employ sMatGen to generate output s-matrix data for each frame.
for i=1:length(Frames)
    Frames(i).sMat = sMatGen(Frames(i).Data);
end
```

epochFinderBeta.m

```matlab
function [Epochs,epochMat] = epochFinderBeta(sound,type)

% This function performs analysis on each epoch in the data ...
    input 'sound'.
% Depending on the value of 'type' different accompanying data ...
    to the epoch
% duration can be found.
%
% Notes: Only shape information extracted at present.

% Check if data is a wav file or an array and read data in ...
    accordingly.
if ischar(sound)
    wav = wavread(sound);
else
    [row,col] = size(sound);
    if row<col
        wav = sound;
    else
        wav = sound';
    end
end

% Use "greater−than" function to find positive and negative ...
    epochs of the
% data.
posNeg = wav>0;
% Create an indexer for going through the data
n = 2:(length(posNeg));
% Generate a matrix of 0s and 1s denoting where the ZCs are. ...
    Abs is
% used so all results are +ve.
zeroCross = abs(posNeg(n) − posNeg(n−1));
% Find all the non−zero locations using the find() function.
NZ = find(zeroCross);
% Start values
epochMat = NZ';

% Find end values
epochMat(1:length(epochMat)−1,2) = NZ(1,2:length(NZ));
% Add 1 to all epoch start locations except the first epoch.
epochMat(2:length(epochMat)−1,1) = ...
    epochMat(2:length(epochMat)−1,1)+1;
% Find durations
epochMat(:,3) = epochMat(:,2)−epochMat(:,1);

% Remove epochs that have a duration of less than 1.
lto = epochMat(:,3) ≤ 1;
m = find(lto);
epochMat(m,:) = [];

% Place all epochMat data into a Data Structure with the form
% Data.Epoch(n).start
%               .end
%               .dur
%               .(this will eventually be the result of the ...
    following
%                    algorithms)

Epochs = struct('start',mat2cell(epochMat(:,1), ...
     [ones(size(epochMat(:,1),1),1)],[1]), ...
         'end',mat2cell(epochMat(:,2),...
     [ones(size(epochMat(:,2),1),1)],[1]),'dur',mat2cell(epochMat(:,3),...
     [ones(size(epochMat(:,3),1),1)],[1]));


% The features extracted from the input data are dependant on ...
    the input
% 'type'.
% Find relevant features and set Epochs. Only 1 option at present.
if type==1;
    Epochs = minFinderSimple(Epochs,wav);
end
```

minFinderSimple.m

```
function Epochs = minFinderSimple(Epochs,wav)

% This function very simply finds the number of minima (+ve or ...
    −ve) in
% each epoch contained within the data structure input ...
    'Epochs'. It performs
% this using simple logical operations on the epoch data ...
    gathered from the
% input 'wav'. Shape information is added to the data structure ...
    Epochs.

% For each epoch...
for i = 1:length(Epochs);
    % Get data for the current epoch
    epoch = wav(Epochs(i).start : Epochs(i).end);
    % Top and tail the epoch with a zero
    epoch(2:length(epoch)+1) = epoch;
    epoch(1) = 0;
    epoch(length(epoch)+1) = 0;
    % abs this data − shape data is unsigned so this does not ...
        affect
    % anything else.
    epoch = abs(epoch);
    % Create some index variables.
    n = 1:length(epoch) − 1;
    m = 2:length(epoch);
    % Find where the data in the epoch is ascending
    ascEpoch = epoch(n) > epoch(n+1);
    % Make the length of this new data the same as that of the ...
```

```
        epoch.
    ascEpoch(length(epoch)) = 0;
    % The following operation makes transitions from descending to
    % ascending appear as a 1, and vice-versa as a -1.
    shArray = ascEpoch(m-1) - ascEpoch(m);
    % Sum all of the 1s to get the shape (i.e. ignore the -1s) and
    % substract 1 for end of epoch anomaly.
    shape = sum(shArray>0);
    if shape > 0
        Epochs(i).shape = sum(shArray>0) - 1;
    else
        Epochs(i).shape = sum(shArray>0);
    end
end
```

sMatGen.m

```
function sMat = sMatGen(Epochs)

% Takes in Epochs data structure and returns a "codebook"

% Get data out of Epochs into a matrix
for i = 1:length(Epochs);
    dsMatrix(i,1) = Epochs(i).dur;
    dsMatrix(i,2) = Epochs(i).shape;
end

% Sort data according to d
dsMatrix = sortrows(dsMatrix);

% Initialise d x s matrix
if max(dsMatrix(:,1))>150
    dSize = max(dsMatrix(:,1));
else
    dSize = 150;
end

if max(dsMatrix(:,2))>15
    sSize = max(dsMatrix(:,2)+1);
else
    sSize = 16;
end
freqMatrix = zeros(dSize,sSize);

% Generate a d x s matrix with frequency of pairs in each location
for i = 1:size(dsMatrix,1);
    freqMatrix(dsMatrix(i,1),dsMatrix(i,2) + 1) = ...
        freqMatrix(dsMatrix(i,1),dsMatrix(i,2) + 1) + 1;
end

% Sorting algorithm to create a n x 3 matrix containing ...
   duration, shape
% and frequency data to output.
```

```
[d,s] = size(freqMatrix);
count = 1;
for i = 1:16
    for j = i:150;
        if i < j
            dsfData(count,1) = j;
            dsfData(count,2) = i−1;
            dsfData(count,3) = freqMatrix(j,i);
            count = count + 1;
        end
    end
end

% Sort this data out using sMatSorter1716, sMatSorter343 or ...
    sMatSort262
% Comment out as necessary.
%sMat = sMatSorter1716(dsfData); % 1716 s−matrix
%sMat = sMatSorter343(dsfData); % 343 s−matrix
sMat = sMatSorter262(dsfData); % 262 s−matrix

% Normalise sMat − unless max is zero.
if max(sMat(:,3)) ≠ 0
    sMat(:,3) = sMat(:,3) / max(sMat(:,3));
end
```

sMatSorter1716.m

```
function [sMat] = sMatSorter1716(dsfData)

% Initialise s matrix
sMat = zeros(1793,3);
% Initialise a counter
count = 1;
% For each row in dsMatrix
for i = 1:length(dsfData)
    % Comparing value
    c = dsfData(i,1);
    % Switch/case statements...switch is logical value true
    switch logical(true)
        case {c>1 && c≤20}
            dRange = c−1;
            indexD = (((c−1)^2 + (c−1))/2) + dsfData(i,2);
        case {c>20 && c≤30}
            dRange = 20;
            indexD = 210 + dsfData(i,2);
        case {c>30 && c≤40}
            dRange = 21;
            indexD = 249 + dsfData(i,2);
        case {c>40 && c≤50}
            dRange = 22;
            indexD = 298 + dsfData(i,2);
        case {c>50 && c≤60}
            dRange = 23;
```

```
                indexD = 357 + dsfData(i,2);
            case {c>60 && c≤70}
                dRange = 24;
                indexD = 426 + dsfData(i,2);
            case {c>70 && c≤80}
                dRange = 25;
                indexD = 502 + dsfData(i,2);
            case {c>80 && c≤90}
                dRange = 26;
                indexD = 578 + dsfData(i,2);
            case {c>90 && c≤100}
                dRange = 27;
                indexD = 654 + dsfData(i,2);
            case {c>100 && c≤150}
                dRange = 28;
                indexD = 730 + dsfData(i,2);
            case {c>150 && c≤200}
                dRange = 29;
                indexD = 806 + dsfData(i,2);
            case {c>200 && c≤250}
                dRange = 30;
                indexD = 882 + dsfData(i,2);
            case {c>250 && c≤300}
                dRange = 31;
                indexD = 958 + dsfData(i,2);
            case {c>300 && c≤350}
                dRange = 32;
                indexD = 1034 + dsfData(i,2);
            case {c>350 && c≤400}
                dRange = 33;
                indexD = 1110 + dsfData(i,2);
            case {c>400 && c≤450}
                dRange = 34;
                indexD = 1186 + dsfData(i,2);
            case {c>450 && c≤500}
                dRange = 35;
                indexD = 1262 + dsfData(i,2);
            case {c>500 && c≤600}
                dRange = 36;
                indexD = 1338 + dsfData(i,2);
            case {c>600 && c≤700}
                dRange = 37;
                indexD = 1414 + dsfData(i,2);
            case {c>700 && c≤800}
                dRange = 38;
                indexD = 1490 + dsfData(i,2);
            case {c>800 && c≤900}
                dRange = 39;
                indexD = 1566 + dsfData(i,2);
            case {c>900 && c≤1000}
                dRange = 40;
                indexD = 1642 + dsfData(i,2);
            otherwise
                dRange = 41;
                indexD = 1718 + dsfData(i,2);
        end
```

```
    % Store relevant data in sMat
    sMat(indexD,3) = dsfData(i,3);
    % For dev purposes, let's see what the D−S data is...
    sMat(indexD,1) = dRange;
    sMat(indexD,2) = dsfData(i,2);
    % Remove any rows after row 1717 as we're not too concerned ...
        with them.
    sMat(1717:length(sMat),:) = [];
end
```

sMatSorter343.m

```
function [sMat] = sMatSorter343(dsfData)
% Adapted sMatSorter to use a smaller number of ranges by ...
    limiting D to 150
% and S to 15.
% Initialise s matrix
sMat = zeros(344,3);
% Initialise a counter
count = 1;
% For each row in dsMatrix
for i = 1:length(dsfData)
        % Comparing value, "Duration"
        c = dsfData(i,1);
        % Switch/case statements...switch is logical value true
        switch logical(true)
            case {c>1 && c≤20}
                dRange = c−1;
                % This is a triangle number calculation
                indexD = (((c−1)ˆ2 + (c−1))/2) + dsfData(i,2);
            case {c>20 && c≤30}
                dRange = 20;
                indexD = 200 + dsfData(i,2);
            case {c>30 && c≤40}
                dRange = 21;
                indexD = 216 + dsfData(i,2);
            case {c>40 && c≤50}
                dRange = 22;
                indexD = 232 + dsfData(i,2);
            case {c>50 && c≤60}
                dRange = 23;
                indexD = 248 + dsfData(i,2);
            case {c>60 && c≤70}
                dRange = 24;
                indexD = 264 + dsfData(i,2);
            case {c>70 && c≤80}
                dRange = 25;
                indexD = 280 + dsfData(i,2);
            case {c>80 && c≤90}
                dRange = 26;
                indexD = 296 + dsfData(i,2);
            case {c>90 && c≤100}
                dRange = 27;
                indexD = 312 + dsfData(i,2);
            case {c>100 && c≤150}
                dRange = 28;
                indexD = 328 + dsfData(i,2);
            otherwise
                dRange = 29;
                indexD = 344 + dsfData(i,2);

    end
    % Store relevant data in sMat
    sMat(indexD,3) = sMat(indexD,3) + dsfData(i,3);
    % For dev purposes, let's see what the D−S data is...
    sMat(indexD,1) = dRange;
    sMat(indexD,2) = dsfData(i,2);
    % Remove any rows after row 343 as we're not too concerned ...
       with them.
    sMat(344:length(sMat),:) = [];
end
```

sMatSorter262.m

```
function [sMat] = sMatSorter262(dsfData)
% Adapted sMatSorterSmall to use a smaller number of codes by ...
    eliminating
% those codes that do not get used. MaxS has been seen to never ...
    exceed D/2.
%
% In the switch case statements below, it is the case for c>1, ...
    c≤20 that
% needed changing. Obviously the indexD for all further codes ...
    also needs
% adapting.

% Initialise s matrix
sMat = zeros(263,3);

% For each row in dsMatrix
for i = 1:length(dsfData)
        % Comparing value, "Duration"
        c = dsfData(i,1);
        % Switch/case statements...switch is logical value true
        switch logical(true)
            case {c==2}
                indexD = 1;
            case {c==3}
                indexD = 2 + dsfData(i,2);
            case {c==4}
                indexD = 4 + dsfData(i,2);
            case {c==5}
                indexD = 7 + dsfData(i,2);
            case {c==6}
                indexD = 10 + dsfData(i,2);
            case {c==7}
```

```
                indexD = 14 + dsfData(i,2);
            case {c==8}
                indexD = 18 + dsfData(i,2);
            case {c==9}
                indexD = 23 + dsfData(i,2);
            case {c==10}
                indexD = 28 + dsfData(i,2);
            case {c==11}
                indexD = 34 + dsfData(i,2);
            case {c==12}
                indexD = 40 + dsfData(i,2);
            case {c==13}
                indexD = 47 + dsfData(i,2);
            case {c==14}
                indexD = 54 + dsfData(i,2);
            case {c==15}
                indexD = 62 + dsfData(i,2);
            case {c==16}
                indexD = 70 + dsfData(i,2);
            case {c==17}
                indexD = 79 + dsfData(i,2);
            case {c==18}
                indexD = 88 + dsfData(i,2);
            case {c==19}
                indexD = 98 + dsfData(i,2);
            case {c==20}
                indexD = 108 + dsfData(i,2);
            case {c>20 && c≤30}
                indexD = 119 + dsfData(i,2);
            case {c>30 && c≤40}
                indexD = 135 + dsfData(i,2);
            case {c>40 && c≤50}
                indexD = 151 + dsfData(i,2);
            case {c>50 && c≤60}
                indexD = 167 + dsfData(i,2);
            case {c>60 && c≤70}
                indexD = 183 + dsfData(i,2);
            case {c>70 && c≤80}
                indexD = 199 + dsfData(i,2);
            case {c>80 && c≤90}
                indexD = 215 + dsfData(i,2);
            case {c>90 && c≤100}
                indexD = 231 + dsfData(i,2);
            case {c>100 && c≤150}
                indexD = 247 + dsfData(i,2);
            otherwise
                indexD = 263 + dsfData(i,2);
        end
    % Store relevant data in sMat
    sMat(indexD,3) = sMat(indexD,3) + dsfData(i,3);
    % For dev purposes, let's see what the D-S data is...
    sMat(indexD,1) = 0;
    sMat(indexD,2) = dsfData(i,2);
    % Remove any rows after row 263 as they are irrelevant.
    sMat(263:length(sMat),:) = [];
end
```

returnSMatrices.m

```
function [retSMat,targetVec] = returnSMatrices(path, ...
    framelength, type)
% Returns a matrix of S−matrices for a given folder of WAV files.

% Declare our overall matrix.
resMatrix = [];

% First some file handling is required to get a list of files ...
    contained
% in the directory
filesTemp = dir(path);

% The first 2 of these are "." and ".." respectively. I.e. not ...
    wav files.
% Get rid.
files(1:length(filesTemp)−2,1) = filesTemp(3:length(filesTemp),1);

% Analysis and construction of output matrix.
for i = 1:length(files)
    % Check that the file is a file and not a directory
    if files(i,1).isdir == 0
        % Generate string for wav file to analyse
        filename = strcat(path,files(i,1).name);
        % Check that this file is in fact a wav file
        [p,n,e,v]=fileparts(filename);
        if strcmp(e,'.wav')
            % Perform TDSC analysis using sMatReturn
            datasmat = sMatReturn(filename,framelength);
            % Check what type of audio is being classified and ...
                generate a
            % target for the WAV file.
            % Cicadas
            if strcmp(type,'cicadas')
                if ¬isempty(findstr('BH',n))
                    datasmat(size(datasmat,1)+1,:) = 1;
                elseif ¬isempty(findstr('FL',n))
                    datasmat(size(datasmat,1)+1,:) = 2;
                elseif ¬isempty(findstr('JP',n))
                    datasmat(size(datasmat,1)+1,:) = 3;
                end
            end
            % Single output networks − Cicadas
            if strcmp(type,'s_BH')
                if ¬isempty(findstr('BH',n))
                    datasmat(size(datasmat,1)+1,:) = 1;
                else datasmat(size(datasmat,1)+1,:) = 0;
                end
            elseif strcmp(type,'s_FL')
                if ¬isempty(findstr('FL',n))
                    datasmat(size(datasmat,1)+1,:) = 1;
                else datasmat(size(datasmat,1)+1,:) = 0;
                end
        elseif strcmp(type,'s_JP')
            if ¬isempty(findstr('JP',n))
                datasmat(size(datasmat,1)+1,:) = 1;
            else datasmat(size(datasmat,1)+1,:) = 0;
            end
        end
        % Sine Mixtures — for testing purposes
        if strcmp(type,'sinemix');
            if ¬isempty(findstr('SineM4.wav',n))
                datasmat(size(datasmat,1)+1,:) = 1;
            elseif ¬isempty(findstr('SineM7.wav',n))
                datasmat(size(datasmat,1)+1,:) = 2;
            end
        end
        % General Audio
        if strcmp(type,'gen')
            if ¬isempty(findstr('Air0',n))
                datasmat(size(datasmat,1)+1,:) = 1;
            elseif ¬isempty(findstr('AirCon0',n))
                datasmat(size(datasmat,1)+1,:) = 2;
            elseif ¬isempty(findstr('Bird0',n))
                datasmat(size(datasmat,1)+1,:) = 3;
            elseif ¬isempty(findstr('Building0',n))
                datasmat(size(datasmat,1)+1,:) = 4;
            elseif ¬isempty(findstr('Rail0',n))
                datasmat(size(datasmat,1)+1,:) = 5;
            elseif ¬isempty(findstr('Road0',n))
                datasmat(size(datasmat,1)+1,:) = 6;
            end
        end
        % General Audio inc Cicada
        if strcmp(type,'geninc')
            if ¬isempty(findstr('Air0',n))
                datasmat(size(datasmat,1)+1,:) = 1;
            elseif ¬isempty(findstr('AirCon0',n))
                datasmat(size(datasmat,1)+1,:) = 2;
            elseif ¬isempty(findstr('Bird0',n))
                datasmat(size(datasmat,1)+1,:) = 3;
            elseif ¬isempty(findstr('Building0',n))
                datasmat(size(datasmat,1)+1,:) = 4;
            elseif ¬isempty(findstr('Rail0',n))
                datasmat(size(datasmat,1)+1,:) = 5;
            elseif ¬isempty(findstr('Road0',n))
                datasmat(size(datasmat,1)+1,:) = 6;
            elseif ¬isempty(findstr('Cic0',n))
                datasmat(size(datasmat,1)+1,:) = 7;
            end
        end
        % Reduced data test set
        if strcmp(type,'test')
            if ¬isempty(findstr('AirCon0',n))
                datasmat(size(datasmat,1)+1,:) = 1;
            elseif ¬isempty(findstr('Bird0',n))
                datasmat(size(datasmat,1)+1,:) = 2;
            elseif ¬isempty(findstr('Road0',n))
                datasmat(size(datasmat,1)+1,:) = 3;
            end
        end
        % Single output networks — full general
        if strcmp(type,'s_air')
            if ¬isempty(findstr('Air0',n))
                datasmat(size(datasmat,1)+1,:) = 1;
            else datasmat(size(datasmat,1)+1,:) = 0;
            end
        elseif strcmp(type,'s_ac')
            if ¬isempty(findstr('AirCon0',n))
                datasmat(size(datasmat,1)+1,:) = 1;
            else datasmat(size(datasmat,1)+1,:) = 0;
            end
        elseif strcmp(type,'s_bird')
            if ¬isempty(findstr('Bird0',n))
                datasmat(size(datasmat,1)+1,:) = 1;
            else datasmat(size(datasmat,1)+1,:) = 0;
            end
        elseif strcmp(type,'s_build')
            if ¬isempty(findstr('Building0',n))
                datasmat(size(datasmat,1)+1,:) = 1;
            else datasmat(size(datasmat,1)+1,:) = 0;
            end
        elseif strcmp(type,'s_rail')
            if ¬isempty(findstr('Rail0',n))
                datasmat(size(datasmat,1)+1,:) = 1;
            else datasmat(size(datasmat,1)+1,:) = 0;
            end
        elseif strcmp(type,'s_road')
            if ¬isempty(findstr('Road0',n))
                datasmat(size(datasmat,1)+1,:) = 1;
            else datasmat(size(datasmat,1)+1,:) = 0;
            end
        end
         % Single output networks — network testing (half ...
             files)
        if strcmp(type,'st_ac')
            if ¬isempty(findstr('AirCon0',n))
                datasmat(size(datasmat,1)+1,:) = 1;
            else datasmat(size(datasmat,1)+1,:) = 0;
            end
        elseif strcmp(type,'st_bird')
            if ¬isempty(findstr('Bird0',n))
                datasmat(size(datasmat,1)+1,:) = 1;
            else datasmat(size(datasmat,1)+1,:) = 0;
            end
        elseif strcmp(type,'st_road')
            if ¬isempty(findstr('Road0',n))
                datasmat(size(datasmat,1)+1,:) = 1;
            else datasmat(size(datasmat,1)+1,:) = 0;
            end
        end
        % Single output *LVQ* networks — network testing ...
            (half files)
        if strcmp(type,'dt_ac')
            if ¬isempty(findstr('AirCon0',n))
```

```
                    datasmat(size(datasmat,1)+1,:) = 1;
                else datasmat(size(datasmat,1)+1,:) = 2;
                end
            elseif strcmp(type,'dt_bird')
                if ¬isempty(findstr('Bird0',n))
                    datasmat(size(datasmat,1)+1,:) = 1;
                else datasmat(size(datasmat,1)+1,:) = 2;
                end
            elseif strcmp(type,'dt_road')
                if ¬isempty(findstr('Road0',n))
                    datasmat(size(datasmat,1)+1,:) = 1;
                else datasmat(size(datasmat,1)+1,:) = 2;
                end
            end
            % Add this S−matrix with its target to the overall ...
                matrix.
            resMatrix(: , (size(resMatrix,2)+1) : ...
                (size(resMatrix,2)...
                + (size(datasmat,2)))) = datasmat;
        end
    end
end


% Extract the target vector (i.e. the last row of the matrix).
targetVec = resMatrix(size(resMatrix,1),:);
resMatrix(size(resMatrix,1),:) = [];

% Return the results
retSMat = resMatrix;
```

multiSMatReturn.m

```
function SmatData = multiSMatReturn(path,framelength)

% This function returns a Data structure of S−matrices for all ...
    of the WAV
% files located in "path" using a framelength of "framelength" ...
    seconds.

% Create the output Struct with 2 fields; "filelist" − a list ...
    of the
% filenames, and "sMat" − the S−matrix.
SmatData = struct('filename',[],'sMat',[]);

% File handling
filesTemp = dir(path);

% The first 2 of these are "." and ".." respectively. I.e. not ...
    wav files.
% Remove.
files(1:length(filesTemp)−2,1) = filesTemp(3:length(filesTemp),1);

```

```
% Matrix to store filenames
filenames = [];

% Analyse.
for i = 1:length(files)
    % Check that the file is a file and not a directory
    if files(i,1).isdir == 0
        % Generate string for wav file to analyse
        filename = strcat(path,files(i,1).name);
        % Check that this file is in fact a wav file
        [p,n,e,v]=fileparts(filename);
        if strcmp(e,'.wav')
            % Perform TDSC analysis using sMatReturn
            datasmat = sMatReturn(filename,framelength);
            % Add the S-matrix to the output data structure
            SmatData(i).sMat = datasmat;
            % Add the filename to the output data structure
            SmatData(i).filename = n;
            % Add the filename to a matrix of filenames
            filenames(length(filenames)+1:length(filenames)+length(n)) ...
                = n;
        end
    end
end
```

C.2 MTDSC Code

mPacketStackReturn.m

```
function mPacketStack = mPacketStackReturn(wavFile, mean, ...
    noOfEpochs)
% Given an input wavFile, this function returns a stacked ...
    mPacket ready
% for input to a neural network. If "mean" is true then the ...
    mean of the
% mPackets is found and returned.

% Get the mPackets for the wavFile
mPacket = wavMPacketGen(wavFile,noOfEpochs);

% Stack the mPackets.
mPacketStack = mPacketStacker(mPacket,mean);
```

wavMPacketGen.m

```
function wavMPacket = wavMPacketGen(wav, noOfEpochs)

% Generates a stitched together matrix of M-Packet data for a ...
```

```
    whole wav
% file. The number of epochs used is trimmed down to be a ...
    multiple of
% noOfEpochs.

% Create minNoOfEpochs, the number of epochs used in the ...
    smallest segments
% of the M-packet. This is always noOfEpochs / 64. There are 7 ...
    levels to
% each M-packet. The 64 comes from 2^(7-1). Catch errors with ...
    noOfEpochs
% that are not a multiple of 64.
minNoOfEpochs = noOfEpochs / 64;
if rem(noOfEpochs,64) ≠ 0
    error('Number of epochs must be a multiple of 64');
end

% Read in the WAV file.
wavData = wavread(wav);

% Find all of the Epochs
TempEpochs = epochFinderBeta(wavData, 1);

% If the number of epochs in TempEpochs is not an exact ...
    multiple of
% noOfEpochs then TempEpochs will be enlarged so its size is an ...
    exact
% multiple of noOfEpochs. The additional data will be identical ...
    to the
% first X epochs contained in TempEpochs (where X is the number ...
    of epochs
% that TempEpochs is short of being an exact multiple).
if rem(length(TempEpochs),noOfEpochs) ≠ 0
    % If length(TempEpochs) is less than noOfEpochs/2, repeat ...
        TempEpochs
    % within itself before finding FillerEpochs
    while length(TempEpochs)/2 < noOfEpochs
        TempEpochs(length(TempEpochs)+1 : length(TempEpochs) + ...
            length(TempEpochs)) = TempEpochs;
    end
    FillerEpochs = TempEpochs(1:(noOfEpochs - ...
        rem(length(TempEpochs),noOfEpochs)));
    Epochs = TempEpochs;
    Epochs(length(Epochs)+1 : length(Epochs) + ...
        length(FillerEpochs)) = ...
        FillerEpochs;
else
    Epochs = TempEpochs;
end

% Initialise output matrix
wavMPacket = zeros(7, 64 * (length(Epochs) / noOfEpochs));

% Generate all the data and fill up the output matrix
for i = 1:length(Epochs)/noOfEpochs
    % Generate mPacket data for each noOfEpochs epochs
    mPacket_pre = mPacketGen(Epochs((i*noOfEpochs)-...
        (noOfEpochs-1):(i*noOfEpochs)),minNoOfEpochs);
    mPacket = mPacketRearrange(mPacket_pre);
    % Add mPacket to the output matrix
    wavMPacket(:,(i*64)-63 : (i*64)) = mPacket;
end

% Normalise mPacket to itself
wavMPacket = wavMPacket / max(max(wavMPacket));
```

mPacketGen.m

```
function mPacket_pre = mPacketGen(Epochs,div)

% This function will take in a number (power of 2) of epochs ...
    stored in a
% data structure. The output will be a data matrix, an mPacket.

% Create the empty mPacket.
mPacket_pre = zeros(7, (length(Epochs) / div));

% Create an empty temporary storage matrix.
storage = zeros(7, length(Epochs));

% Using a for loop, insert data into the storage matrix. Find
% floor(log2(Epochs(i).dur)) to calculate which row the data ...
    should go
% into.
for i = 1:length(Epochs);
    storage(floor(log2(Epochs(i).dur)), i) = Epochs(i).shape + 1;
end

% For every "div" epochs, find the mean of the shape and store ...
    this in
% mPacket_pre.
for i = 1:length(Epochs)/div
    for j = 1:7
        mPacket_pre(j,i) = sum(storage(j, (div*i)-(div-1) : ...
            (div*i)),2);
    end
end
```

mPacketRearrange.m

```
function mPacket_out =  mPacketRearrange(mPacket)

% Re-arranges mPackets into the dyadic form.
% Very simply, matrix locations are set to the same value for a ...
    given span
% of 2, 4 , 8, etc. values.
```

```matlab

% Find noOfEpochs based on length of input
noOfEpochs = length(mPacket);

% Initialise mPacket_out
mPacket_out = zeros(7,64);

% Mean of 1*noOfEpochs epochs
mPacket_out(7,:) = mean(mPacket(7,:));

% Mean of 2*(noOfEpochs/2) epochs
for i = 1:2
mPacket_out(6,(32*i)-31:(32*i)) = ...
    mean(mPacket(6,(32*i)-31:(32*i)));
end

% Mean of 4*(noOfEpochs/4) epochs
for i = 1:4
mPacket_out(5,(16*i)-15:(16*i)) = ...
    mean(mPacket(5,(16*i)-15:(16*i)));
end

% Mean of 8*(noOfEpochs/8) epochs
for i = 1:8
mPacket_out(4,(8*i)-7:(8*i)) = mean(mPacket(4,(8*i)-7:(8*i)));
end

% Mean of 16*(noOfEpochs/16) epochs
for i = 1:16
mPacket_out(3,(4*i)-3:(4*i)) = mean(mPacket(3,(4*i)-3:(4*i)));
end

% Mean of 32*(noOfEpochs/32) epochs
for i = 1:32
mPacket_out(2,(2*i)-1:(2*i)) = mean(mPacket(2,(2*i)-1:(2*i)));
end

% Mean of 64*(noOfEpochs/64) epochs - data already exists
mPacket_out(1,:) = mPacket(1,:);
```

mPacketStacker.m

```matlab
function mp_stack = mPacketStacker(mPackets,mean)

% Returns a stacked mPacket ready for classification. If "mean" ...
    is "true"
% then the mean of the mPackets is used as the stack and ...
    "mp_stack" is a
% 1 x 127 matrix. If "mean" is "false" then "mp_stack" will be a
% NoOfMPackets x 127 matrix.

% If "mean" is true, find the mean mPacket
if mean
```

```
    mPackets_mean = zeros(7,64);
    for i = 1:length(mPackets)/64
        mPackets_mean = mPackets_mean + ...
            mPackets(:, (i*64) - 63 : (i*64) );
    end

    mPackets = mPackets_mean / (length(mPackets)/64);
end

% Now stack the mPacket(s). If "mean" is true then "mPackets" ...
    should now
% only have a length of 64 so there will only be 1 'stack' in ...
    the output
% variable "mp_stack".
% Number of mPackets
noOfMPackets = length(mPackets)/64;

% Create the output stack
mp_stack = zeros(127,noOfMPackets);

% Go through all of the mPackets
for mp = 1 : noOfMPackets
    % Create a temporary copy of the current mPacket
    curr_mPacket = mPackets(:,((mp*64) - 63):mp*64);
    % Create the stack for this mPacket
    curr_stack = mPackets(1, (mp*64) - 63 : (mp*64) );
    % Perform the stacking for each mPacket
    for i = 2:7 % Rows to be stacked
        for j = 1:(64 / 2^(i-1)); % Columns, effectively
            curr_stack(length(curr_stack) + 1) = ...
                curr_mPacket(i, (j * 2^(i-1)));
        end
    end
    % Transform the stacked mPacket and add it to the output ...
        "mp_stack"
    mp_stack(:, mp) = curr_stack';
end
```

C.3 Utility Code

multiLVQTrainer.m

```
function multiLVQTrainer(dataMat,targets,PR,PC,savename)

% Function for training a number of networks with different ...
    numbers of
% hidden layer units. Each network is saved after training.

% First declare all of the networks and their values before ...
    training (this
% is where most typo errors can be caught!).

% PR & PC — same for all networks.
% Change these if using different training data sets.
% PC is the typical class percentages (i.e. percentage of how ...
    many of the
% training data files belong to each class).
% PR is the min and max values for the input elements. As the ...
    inputs are
% normalised these will always be between 0 and 1. The size of ...
    PR will vary
% depending on the number of frames/WAV files being analysed.

% net300 — 300 units
net300 = newlvq(PR,300,PC);
net300.trainParam.epochs = 50;
net300.trainParam.show = 10;

% net100 — 100 units
net100 = newlvq(PR,100,PC);
net100.trainParam.epochs = 50;
net100.trainParam.show = 10;

% net50 — 50 units
net50 = newlvq(PR,50,PC);
net50.trainParam.epochs = 50;
net50.trainParam.show = 10;

% net10 — 10 units
net10 = newlvq(PR,10,PC);
net10.trainParam.epochs = 50;
net10.trainParam.show = 10;

% Create a directory and file names to save data to. If month ...
    is less than
% 10 add a '0' in before it.
% Clock returns a date vector. c(1:3) are year, month, day
c = clock;
if c(2) < 10
    savepath = strcat('C:\Phd\Matlab\Results\',...
        strcat(num2str(c(1)),'0',num2str(c(2)),num2str(c(3))),'_',savename);
else
    savepath = strcat('C:\Phd\Matlab\Results\',...
        strcat(num2str(c(1)),num2str(c(2)),num2str(c(3))),'_',savename);
end
mkdir(savepath);
savefile10 = (strcat(savepath,'\',savename,'_LVQ_10Hidden'));
savefile50 = (strcat(savepath,'\',savename,'_LVQ_50Hidden'));
savefile100 = (strcat(savepath,'\',savename,'_LVQ_100Hidden'));
savefile300 = (strcat(savepath,'\',savename,'_LVQ_300Hidden'));

% Training
% Each network will be saved after training.
tic
[net10,tr10] = train(net10,dataMat,targets);
save(savefile10,'net10','tr10','dataMat','targets','PR','PC');
'10 Hidden Units'
```

```
toc

tic
[net50,tr50] = train(net50,dataMat,targets);
save(savefile50,'net50','tr50','dataMat','targets','PR','PC');
'50 Hidden Units'
toc

tic
[net100,tr100] = train(net100,dataMat,targets);
save(savefile100,'net100','tr100','dataMat','targets','PR','PC');
'100 Hidden Units'
toc

tic
[net300,tr300] = train(net300,dataMat,targets);
save(savefile300,'net300','tr300','dataMat','targets','PR','PC');
'300 Hidden Units'
toc
```

singleOpNetTest.m

```
function [simRes,res,category] = singleOpNetTest(net_air, ...
    net_ac, ...
                                        net_bird, net_build, net_rail, ...
                                        net_road, testData, framelength)

% Function for testing a group of single-output MLP networks.
%
% Takes in 6 trained networks, the path of the test data and a ...
    framelength.
% TDSC features are extracted for the file and each network is ...
    simulated
% with the results. The network with the highest simulation ...
    output value
% is considered the winner.

% Perform TDSC analysis on test data
sMat = sMatReturn(testData,framelength);

% Simulate each network with sMat
simRes(1,:) = sim(net_air,sMat);
simRes(2,:) = sim(net_ac,sMat);
simRes(3,:) = sim(net_bird,sMat);
simRes(4,:) = sim(net_build,sMat);
simRes(5,:) = sim(net_rail,sMat);
simRes(6,:) = sim(net_road,sMat);

% Find mean of simulation results for each network
simResMean = mean(simRes,2);

% Find max value of mean simulation results
[v,category] = max(simResMean);
```

```matlab

% Find max results for each S—matrix and create a vector of ...
    which output
% had the highest output value for each frame.
for i=1:length(simRes)
    [v,ind] = max(simRes(:,i));
    res(i) = ind;
end
```

Appendix D

MATLAB® Neural Network Toolbox Training Functions and Testing Results

D.1 Training Function Summary

Table D.1: *Table summarising the training functions provided for MLP networks by the MATLAB® Neural Network Toolbox. Taken and adapted from the information provided in Chapter 5 of 'The Neural Network Toolbox User's Guide'(Demuth et al., 2008)*

Training function	MATLAB® code	Description
Batch Training	`train`	Weights and biases are updated only after the entire training set has been applied to the network.
Batch Gradient Descent	`traingd`	Steepest descent training function. Weights and biases are updated in the direction of the negative gradient of the performance function.
Batch Gradient Descent with Momentum	`traingdm`	Often gives faster convergence than the previous functions. Momentum allows the network to respond to both the local gradient and to recent trends in the error surface. Momentum acts like a low pass filter, allowing the network to ignore minor features in the error surface. This can help prevent the network getting stuck in a local minimum.

continued on next page...

Table D.1

... continued from previous page

Training function	**MATLAB® code**	**Description**
If the learning rate is too high, a network can oscillate and become unstable. If the learning rate is too low, the algorithm takes too long to converge. The optimal learning rate can vary during training so it is not practical to determine the optimal learning rate prior to training.		
Variable Learning Rate Backpropagation	`traingda`	Solves the issue of finding the ideal learning rate by changing the learning rate as training is carried out. After a training epoch, if the new network error exceeds the old error the learning rate is decreased. If the new error is less than the old error, the learning rate is increased. The increases and decreases to the learning rate are determined by multiplying the old learning rate by a constant (either greater or less than 1 depending on the error difference).
Variable Learning Rate Backpropagation with Momentum	`traingdx`	Similar to `traingda` but uses momentum to prevent the network getting stuck in a local minimum on the error surface and ignore small errors.
Resilient Backpropagation	`trainrp`	Eliminates the problem of only small changes being made to weights and biases even though the weights and biases may be far from optimal. This problem can be caused by using sigmoid transfer functions. Generally converges faster than previous algorithms.
Conjugate Gradient Algorithms: Basic backpropagation adjusts weights in the direction of the steepest descent (negative of the gradient), the direction in which the performance function is decreasing most rapidly. This does not always produce the fastest convergence. Conjugate gradient algorithms perform searches along conjugate directions producing generally faster convergence.		
Fletcher-Reeves Update	`traincgf`	Each of these algorithms take a slightly different approach to calculating the direction of the conjugates in which to search. Line searching is used by several of these algorithms, as well as some of the quasi-Newton algorithms.
Polak-Ribiere Update	`traincgp`	
Powell-Beale Restarts	`traincgb`	
Scaled Conjugate Gradient	`trainscg`	

continued on next page...

Table D.1

. . . continued from previous page

Training function	**MATLAB® code**	**Description**
Quasi-Newton Algorithms: Newtonian algorithms make use of the Hessian matrix (second derivative) of the performance index at the current values of the weights and biases. These algorithms often converge faster than conjugate methods but the Hessian matrices are complex and expensive to compute. Quasi-Newton methods use an approximate Hessian matrix that is updated. This is less complex and expensive. Quasi-Newton methods are more computationally complex and expensive than Conjugate methods.		
BFGS Algorithm	`trainbfg`	Uses the Broyden, Fletcher, Goldfarb, and Shanno (BFGS) update.
One Step Secant Algorithm	`trainoss`	Bridges the gap between the conjugate and BFGS algorithms by reducing the storage and computation requirements of a quasi-Newton algorithm.
Levenberg-Marquardt	`trainlm`	Similar to a quasi-Newton method, this algorithm was designed to approach the faster training speed of Newtonian methods but without having to compute the Hessian matrix. The Levenberg-Marquardt algorithm uses a Jacobian matrix as an approximation to the Hessian matrix.

D.2 Preliminary Testing Results

Table D.2 presents the results from testing 24 neural networks, each using a different combination of topologies, training functions and hidden layer transfer functions.

The four right-most columns show the slope from a regression analysis of the network for each output and a mean across all outputs. The regression analysis is performed between the network response to a test dataset and the corresponding targets for that data. The slope, represented by m, informs of how well the network response matches the targets: a 1 would signify a perfect match.

Table D.2: *Results of preliminary testing with various MLP network designs.*

Topology	Training function	Hidden layer transfer function	Time to train (seconds)	Training epochs completed	Performance at end of training	m(1)	m(2)	m(3)	Mean m(1–3)
10-3	`traingdx`	`tansig`	35.9	2687	0.0010	0.669	0.806	0.778	0.751
		`logsig`	49.7	4001	0.0024	0.663	0.801	0.778	0.748
	`trainrp`	`tansig`	2.1	117	0.0012	0.756	0.789	0.782	0.776
		`logsig`	1.0	60	0.0010	0.643	0.806	0.796	0.748
	`trainscg`	`tansig`	8.9	344	0.1269	0.698	-0.001	0.807	0.501
		`logsig`	**6.2**	**238**	**0.0041**	**0.714**	**0.839**	**0.826**	**0.793**
50-3	`traingdx`	`tansig`	133.2	4001	0.0023	0.683	0.820	0.782	0.762
		`logsig`	133.6	4001	0.0030	0.612	0.790	0.774	0.725
	`trainrp`	`tansig`	2.1	54	0.0008	0.628	0.831	0.802	0.754
		`logsig`	**2.2**	**56**	**0.0010**	**0.691**	**0.755**	**0.848**	**0.764**
	`trainscg`	`tansig`	21.2	304	0.0018	0.709	0.802	0.773	0.761
		`logsig`	23.7	339	0.1246	0.707	0.847	0.000	0.518
100-3	`traingdx`	`tansig`	243.1	4001	0.0012	0.695	0.795	0.819	0.769
		`logsig`	250.1	4001	0.0046	0.645	0.782	0.778	0.735
	`trainrp`	`tansig`	**5.9**	**84**	**0.0008**	**0.722**	**0.793**	**0.857**	**0.790**
		`logsig`	4.4	62	0.0008	0.657	0.829	0.831	0.773
	`trainscg`	`tansig`	46.8	352	0.0012	0.665	0.794	0.801	0.753
		`logsig`	43.4	321	0.0029	0.680	0.771	0.796	0.749
300-3	`traingdx`	`tansig`	750.1	4001	0.0043	0.622	0.815	0.765	0.734
		`logsig`	749.5	4001	0.0160	0.612	0.742	0.774	0.709
	`trainrp`	`tansig`	16.2	78	0.0018	0.657	0.836	0.879	0.791

continued on next page...

Table D.2

... continued from previous page

Topology	**Training function**	**Hidden layer transfer function**	**Time to train (s)**	**Training epochs completed**	**Performance at end of training**	**m(1)**	**m(2)**	**m(3)**	**Mean m(1–3)**
		`logsig`	13.6	65	0.0010	0.652	0.821	0.820	0.764
	`trainscg`	`tansig`	**164.4**	**417**	**0.0053**	**0.717**	**0.827**	**0.815**	**0.786**
		`logsig`	167.4	424	0.0035	0.669	0.774	0.791	0.745

Appendix E

Experimental Results Using the 262 Codebook

E.1 Multiple–output MLP Network – Urban Audio Data

Table E.1: *Full performance results for 100–6 MLP networks classifying urban audio data. Results for all five training and testing sessions are shown. Results in bold face denote the highest performing network in each training and testing session.*

Session	Training function	Hidden layer transfer function	Training time (seconds)	M–value mean
1	traingdx	tansig	430.73	0.680
		logsig	430.70	0.665
	trainrp	tansig	14.09	0.651
		logsig	10.45	0.582
	trainscg	tansig	**172.15**	**0.719**
		logsig	202.92	0.690
2	traingdx	tansig	424.56	0.686
		logsig	420.52	0.674
	trainrp	tansig	13.77	0.602
		logsig	13.57	0.616
	trainscg	tansig	**215.65**	**0.732**
		logsig	245.39	0.690
3	traingdx	tansig	439.98	0.686
		logsig	440.23	0.681
	trainrp	tansig	12.54	0.585
		logsig	12.78	0.608
	trainscg	tansig	**332.96**	**0.691**
		logsig	262.25	0.582
4	traingdx	tansig	428.92	0.595
		logsig	424.65	0.657
	trainrp	tansig	9.89	0.636

continued on next page…

Table E.1

. . . continued from previous page

Session	**Training function**	**Hidden layer transfer function**	**Training time (seconds)**	**M–value mean**
		logsig	12.55	0.623
	trainscg	tansig	181.95	0.691
		logsig	**235.33**	**0.718**
5	traingdx	tansig	431.30	0.695
		logsig	423.21	0.666
	trainrp	tansig	14.75	0.644
		logsig	10.99	0.607
	trainscg	tansig	**246.45**	**0.704**
		logsig	233.72	0.674

E.2 Multiple Single–output MLP Networks – Preliminary Testing

Table E.2 presents a summary of results for the initial testing of the single–output MLP networks. In total 72 single–output MLP networks were trained; three categories, six combinations of training function and transfer function, four different topologies.

Table E.2: *Mean performance results for single–output MLP networks arranged by topology and training and transfer functions.*

Topology	Training function	Transfer function	M–value mean
10-1	`traingdx`	`tansig`	0.7400
		`logsig`	0.7494
	`trainrp`	`tansig`	0.7520
		`logsig`	0.7405
	`trainscg`	`tansig`	0.5074
		`logsig`	**0.7634**
50-1	`traingdx`	`tansig`	0.7288
		`logsig`	0.7408
	`trainrp`	`tansig`	0.7471
		`logsig`	**0.7573**
	`trainscg`	`tansig`	0.7448
		`logsig`	0.5044
100-1	`traingdx`	`tansig`	0.7543
		`logsig`	0.7597
	`trainrp`	`tansig`	0.7595
		`logsig`	0.7597
	`trainscg`	`tansig`	0.7591
		`logsig`	**0.7817**
300-1	`traingdx`	`tansig`	0.7623
		`logsig`	0.7689
	`trainrp`	`tansig`	**0.7755**
		`logsig`	0.7070
	`trainscg`	`tansig`	0.7685
		`logsig`	0.5054

E.3 Six Single–output MLP Networks – Urban Audio Data

Table E.3: *Full performance results for six 100–1 MLP networks classifying urban audio data. Results for all five training and testing sessions are shown.*

Session	Category	Training function	Hidden layer transfer function	Training time (seconds)	M–value mean
1	Air	trainscg	logsig	7.57	0.756
	Air con			19.22	0.594
	Bird			8.62	0.656
	Build			15.62	0.734
	Rail			8.26	0.544
	Road			49.53	0.680
2	Air	trainscg	logsig	11.61	0.826
	Air con			22.77	0.657
	Bird			9.48	0.676
	Build			17.15	0.675
	Rail			9.98	0.604
	Road			30.26	0.685
3	Air	trainscg	logsig	6.88	0.848
	Air con			15.61	0.509
	Bird			12.81	0.738
	Build			20.34	0.666
	Rail			11.95	0.516
	Road			26.73	0.728
4	Air	trainscg	logsig	0.37	0.000
	Air con			0.35	0.000
	Bird			12.97	0.700

continued on next page…

Table E.3

. . . continued from previous page

Session	**Category**	**Training function**	**Hidden layer transfer function**	**Training time (seconds)**	**M–value mean**
	Build			0.36	0.000
	Rail			0.36	0.000
	Road			30.14	0.785
5	Air	trainscg	logsig	2.87	0.868
	Air con			14.31	0.597
	Bird			12.13	0.735
	Build			16.71	0.734
	Rail			12.32	0.540
	Road			16.63	0.785

E.4 Multiple–output LVQ Network – Urban Audio Data

Table E.4: *Full performance data for the five training and testing sessions for the 100 hidden unit, six output LVQ network.*

Session	**Topology**	**Time to train**	**Output 1 M–value**	**Output 2 M–value**	**Output 3 M–value**	**Output 4 M–value**	**Output 5 M–value**	**Output 6 M–value**
1	100-6	501.4776	0.704	0.094	0.707	0.000	0.000	0.729
2		511.0309	0.825	0.047	0.311	0.000	0.000	0.392
3		489.029	0.825	0.000	0.557	0.000	0.000	0.500
4		484.9373	0.629	0.222	0.715	0.000	0.000	0.729
5		496.7992	0.450	0.515	0.706	0.000	0.000	0.808

E.5 Multiple–output MLP Network – Preliminary Testing with *Tibicen* data

Table E.5: *Full performance data for preliminary testing of MLP networks with* Tibicen *audio data.*

Topology	**Training function**	**Hidden layer transfer function**	**Time to train (s)**	**Frames correctly classified (%)**
10-3	`traingdx`	`tansig`	5.10	80.77
	`traingdx`	`logsig`	1.18	72.56
	`trainrp`	`tansig`	0.39	13.85
	`trainrp`	`logsig`	0.39	4.36
	`trainscg`	`tansig`	**0.61**	**86.41**
	`trainscg`	`logsig`	0.47	71.03
50-3	`traingdx`	`tansig`	1.28	67.44
	`traingdx`	`logsig`	1.67	70.00
	`trainrp`	`tansig`	0.35	47.18
	`trainrp`	`logsig`	0.48	20.51
	`trainscg`	`tansig`	0.65	52.31
	`trainscg`	`logsig`	**0.84**	**68.97**
100-3	`traingdx`	`tansig`	1.99	70.77
	`traingdx`	`logsig`	**3.68**	**71.54**
	`trainrp`	`tansig`	0.53	4.87
	`trainrp`	`logsig`	0.60	42.31
	`trainscg`	`tansig`	1.01	65.13
	`trainscg`	`logsig`	0.92	68.21
300-3	`traingdx`	`tansig`	9.38	68.21
	`traingdx`	`logsig`	35.57	68.21
	`trainrp`	`tansig`	1.72	34.62
	`trainrp`	`logsig`	1.43	35.13
	`trainscg`	`tansig`	**4.30**	**71.79**
	`trainscg`	`logsig`	4.14	69.79

E.6 Multiple–output LVQ Network – Preliminary Testing with *Tibicen* data

Table E.6: *Full performance data for preliminary testing of LVQ networks with* Tibicen *audio data.*

Topology	**Time to train (s)**	**Correctly classified frames (%)**
10–3	10.02	74.10
50–3	17.89	50.51
100–3	27.73	86.15
300–3	72.39	0.00

Appendix F

Experimental Results Using MTDSC

F.1 Multiple–output MLP Network – Preliminary Testing

Table F.1: *Full performance results for preliminary testing with a three–output MLP network.*

Topology	Training function	Hidden layer transfer function	Time to train (s)	Training epochs completed	Output 1 M–value	Output 2 M–value	Output 3 M–value	M–value mean
10-3	`traingdx`	`tansig`	4.55	557	0.790	0.965	0.809	0.854
		`logsig`	6.81	840	0.740	0.964	0.849	0.851
	`trainrp`	`tansig`	2.53	291	0.735	0.992	0.765	0.831
		`logsig`	1.64	174	0.679	0.945	0.821	0.815
	`trainscg`	`tansig`	1.60	90	0.720	0.909	0.761	0.797
		`logsig`	**3.40**	**210**	**0.767**	**0.966**	**0.843**	**0.859**
50-3	`traingdx`	`tansig`	8.80	461	0.700	0.945	0.787	0.811
		`logsig`	16.01	827	0.723	0.961	0.792	0.825
	`trainrp`	`tansig`	2.28	103	0.739	0.941	0.677	0.786
		`logsig`	2.32	108	0.747	0.937	0.772	0.819
	`trainscg`	`tansig`	3.01	74	0.728	0.944	0.841	0.837
		`logsig`	**6.64**	**167**	**0.758**	**0.948**	**0.832**	**0.846**
100-3	`traingdx`	`tansig`	20.50	648	0.700	0.943	0.753	0.799
		`logsig`	38.36	1184	0.718	0.962	0.720	0.800
	`trainrp`	`tansig`	3.47	95	0.740	0.985	0.791	0.839
		`logsig`	3.70	103	0.754	0.904	0.688	0.782
	`trainscg`	`tansig`	8.96	139	0.720	0.000	0.780	0.500
		`logsig`	**23.04**	**349**	**0.717**	**0.979**	**0.840**	**0.846**
300-3	`traingdx`	`tansig`	104.42	1118	0.720	0.953	0.737	0.803
		`logsig`	315.07	3221	0.740	0.942	0.712	0.798

continued on next page...

Table F.1

. . . continued from previous page

Topology	**Training function**	**Hidden layer transfer function**	**Time to train (s)**	**Training epochs completed**	**Output 1 M–value**	**Output 2 M–value**	**Output 3 M–value**	**M–value mean**
	`trainrp`	`tansig`	19.87	177	0.729	0.978	0.699	0.802
		`logsig`	11.01	105	0.704	0.987	0.800	0.830
	`trainscg`	`tansig`	30.65	156	0.713	0.975	0.792	0.827
		`logsig`	**42.78**	**215**	**0.797**	**0.972**	**0.747**	**0.839**

F.2 Multiple–output MLP Network – Urban Audio Data

Table F.2: *Full performance results for 50–6 MLP networks classifying MTDSC features of urban audio data. Results for all five training and testing sessions are shown. Results in bold face denote the highest performing network in each training and testing session.*

Session	Training function	Hidden layer transfer function	Training time (seconds)	Mean M–value
1	`traingdx`	`tansig`	25.29	0.654
		`logsig`	34.80	0.655
	`trainrp`	`tansig`	6.95	0.566
		`logsig`	5.64	0.594
	`trainscg`	`tansig`	**14.48**	**0.667**
		`logsig`	16.13	0.053
2	`traingdx`	`tansig`	**41.55**	**0.679**
		`logsig`	32.25	0.668
	`trainrp`	`tansig`	10.26	0.622
		`logsig`	8.22	0.671
	`trainscg`	`tansig`	25.34	0.671
		`logsig`	18.97	0.673
3	`traingdx`	`tansig`	40.84	0.661
		`logsig`	47.33	0.663
	`trainrp`	`tansig`	8.11	0.552
		`logsig`	4.96	0.601
	`trainscg`	`tansig`	18.02	0.648
		`logsig`	**21.19**	**0.690**
4	`traingdx`	`tansig`	22.58	0.618
		`logsig`	48.43	0.666

continued on next page…

Table F.2

. . . continued from previous page

Session	**Training function**	**Hidden layer transfer function**	**Training time (seconds)**	**Mean M–value**
	`trainrp`	`tansig`	6.63	0.615
		`logsig`	5.09	0.610
	`trainscg`	`tansig`	11.87	0.654
		`logsig`	**18.17**	**0.673**
5	`traingdx`	`tansig`	42.47	0.643
		`logsig`	48.94	0.679
	`trainrp`	`tansig`	8.31	0.591
		`logsig`	10.92	0.615
	`trainscg`	`tansig`	16.23	0.689
		logsig	**26.34**	**0.697**

F.3 Multiple Single–output MLP Networks – Preliminary Testing

Table F.3 presents the results of the preliminary testing for single–output MLP networks classifying MTDSC data. In total, 72 different MLP networks were initialised and tested to cover the different categories, topologies, training functions, and transfer functions. The M–values shown in Table F.3 are the mean of the M–values from all three categories.

Table F.3: *Mean performance results for single–output MLP networks classifying MTDSC data arranged by topology and training and transfer functions.*

Topology	Training function	Transfer function	M–value mean
10-1	`traingdx`	`tansig`	0.808
		`logsig`	0.782
	`trainrp`	`tansig`	0.788
		`logsig`	0.794
	`trainscg`	`tansig`	**0.849**
		`logsig`	0.808
50-1	`traingdx`	`tansig`	0.790
		`logsig`	**0.840**
	`trainrp`	`tansig`	0.819
		`logsig`	0.792
	`trainscg`	`tansig`	0.835
		`logsig`	0.813
100-1	`traingdx`	`tansig`	0.821
		`logsig`	0.765
	`trainrp`	`tansig`	0.835
		`logsig`	0.794
	`trainscg`	`tansig`	**0.836**
		`logsig`	0.793
300-1	`traingdx`	`tansig`	0.785
		`logsig`	0.770
	`trainrp`	`tansig`	**0.799**
		`logsig`	0.708
	`trainscg`	`tansig`	0.776
		`logsig`	0.783

F.4 Multiple Single–output MLP Networks – Urban Audio Data

Table F.4: *Full performance results for six 100–1 MLP networks classifying MTDSC features of urban audio data. Results for all 5 training and testing sessions are shown.*

Session	Category	Training function	Hidden layer transfer function	Time to train (s)	M–value
1	Air	`trainscg`	`tansig`	0.96	**0.863**
	Air con			3.49	0.722
	Bird			5.88	0.833
	Build			5.52	0.516
	Rail			7.76	0.257
	Road			7.38	0.565
2	Air	`trainscg`	`tansig`	0.85	**0.872**
	Air con			3.67	0.694
	Bird			7.16	0.817
	Build			4.64	0.417
	Rail			7.23	0.355
	Road			9.90	0.601
3	Air	`trainscg`	`tansig`	1.14	**0.899**
	Air con			5.48	0.705
	Bird			5.77	0.830
	Build			4.49	0.503
	Rail			6.40	0.417
	Road			6.57	0.607
4	Air	`trainscg`	`tansig`	1.00	**0.893**
	Air con			3.26	0.687
	Bird			6.84	0.846
	Build			4.95	0.457

continued on next page...

Table F.4

. . . continued from previous page

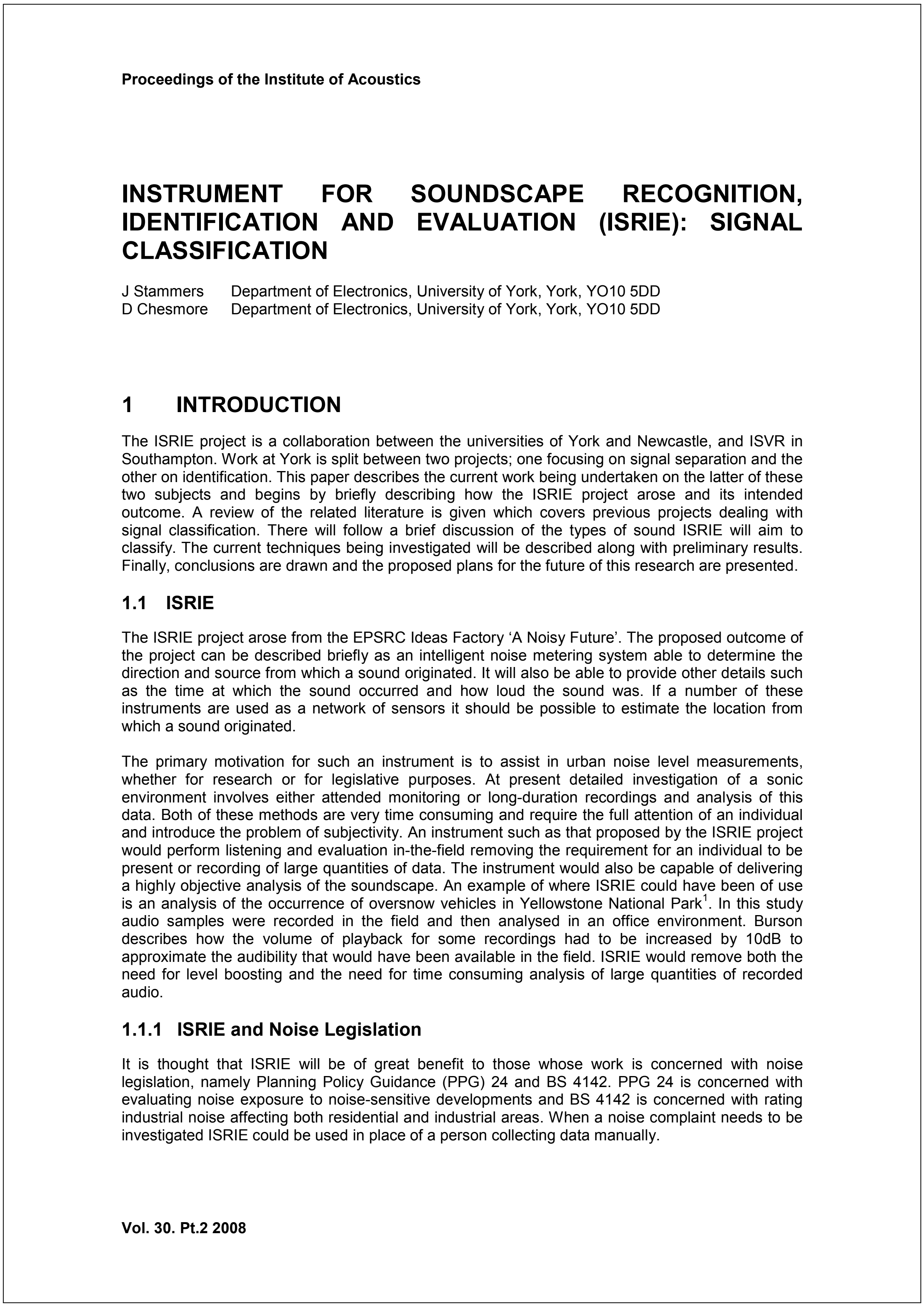
Session	**Category**	**Training function**	**Hidden layer transfer function**	**Time to train (s)**	**M–value**
	Rail			8.26	0.286
	Road			6.61	0.649
5	Air	`trainscg`	`tansig`	0.81	**0.886**
	Air con			5.33	0.739
	Bird			6.32	0.753
	Build			5.64	0.497
	Rail			5.20	0.272
	Road			12.42	0.548

Appendix G

Publications

The following publications are included here for reference:

- Stammers, J. and Chesmore, D. Instrument for soundscape recognition, identification and evaluation (ISRIE): signal classification. In *Proceedings of the Institute of Acoustics Spring Conference: Widening Horizons in Acoustics*, Vol 30(2), Reading, UK, 2008
- Stammers, J. and Chesmore, D. Instrument for soundscape recognition, identification and evaluation (ISRIE): signal classification. In *Proceedings of the 2nd ASA-EAA Joint Conference, Acoustics '08*, Paris, France, 2008
- Bunting, O., Stammers, J., Chesmore, D., Bouzid, O., Yun Tian, G., Karatsovis, C., and Dyne, S. Instrument for soundscape recognition, identification and evaluation (ISRIE): technology and practical uses. In *Proceedings of Euronoise 2009*, Edinburgh, Scotland, 2009



Proceedings of the Institute of Acoustics

INSTRUMENT FOR SOUNDSCAPE RECOGNITION, IDENTIFICATION AND EVALUATION (ISRIE): SIGNAL CLASSIFICATION

J Stammers Department of Electronics, University of York, York, YO10 5DD
D Chesmore Department of Electronics, University of York, York, YO10 5DD

1 INTRODUCTION

The ISRIE project is a collaboration between the universities of York and Newcastle, and ISVR in Southampton. Work at York is split between two projects; one focusing on signal separation and the other on identification. This paper describes the current work being undertaken on the latter of these two subjects and begins by briefly describing how the ISRIE project arose and its intended outcome. A review of the related literature is given which covers previous projects dealing with signal classification. There will follow a brief discussion of the types of sound ISRIE will aim to classify. The current techniques being investigated will be described along with preliminary results. Finally, conclusions are drawn and the proposed plans for the future of this research are presented.

1.1 ISRIE

The ISRIE project arose from the EPSRC Ideas Factory 'A Noisy Future'. The proposed outcome of the project can be described briefly as an intelligent noise metering system able to determine the direction and source from which a sound originated. It will also be able to provide other details such as the time at which the sound occurred and how loud the sound was. If a number of these instruments are used as a network of sensors it should be possible to estimate the location from which a sound originated.

The primary motivation for such an instrument is to assist in urban noise level measurements, whether for research or for legislative purposes. At present detailed investigation of a sonic environment involves either attended monitoring or long-duration recordings and analysis of this data. Both of these methods are very time consuming and require the full attention of an individual and introduce the problem of subjectivity. An instrument such as that proposed by the ISRIE project would perform listening and evaluation in-the-field removing the requirement for an individual to be present or recording of large quantities of data. The instrument would also be capable of delivering a highly objective analysis of the soundscape. An example of where ISRIE could have been of use is an analysis of the occurrence of oversnow vehicles in Yellowstone National Park[1]. In this study audio samples were recorded in the field and then analysed in an office environment. Burson describes how the volume of playback for some recordings had to be increased by 10dB to approximate the audibility that would have been available in the field. ISRIE would remove both the need for level boosting and the need for time consuming analysis of large quantities of recorded audio.

1.1.1 ISRIE and Noise Legislation

It is thought that ISRIE will be of great benefit to those whose work is concerned with noise legislation, namely Planning Policy Guidance (PPG) 24 and BS 4142. PPG 24 is concerned with evaluating noise exposure to noise-sensitive developments and BS 4142 is concerned with rating industrial noise affecting both residential and industrial areas. When a noise complaint needs to be investigated ISRIE could be used in place of a person collecting data manually.

Vol. 30. Pt.2 2008

PPG 24 uses four noise exposure category (NEC) bands to describe a sound. Where a measurement is placed within these bands is dependant on the contribution of each of the four noise categories; road traffic, rail traffic, air traffic and mixed noise sources. Currently the contribution of each of these categories to determine where a measurement is placed is performed by a person and could therefore be quite subjective. ISRIE would be able to provide a purely objective NEC band recommendation. Analysis for BS 4142 would also benefit from ISRIE as it would remove the excessive labour in performing the numerous measurements required.

1.2 Sound Categories

As part of the signal classification research it was deemed necessary to develop an acoustic taxonomy into which sounds would be categorised. Initially it was thought that the quantity of categories would be very high (given how many different sounds can be observed in an urban environment). However, as a result of discussing analysis of PPG 24 and BS 4142 with the project partners from ISVR (Southampton) the categories given in Figure 1 have been decided upon as the most important to be identified.

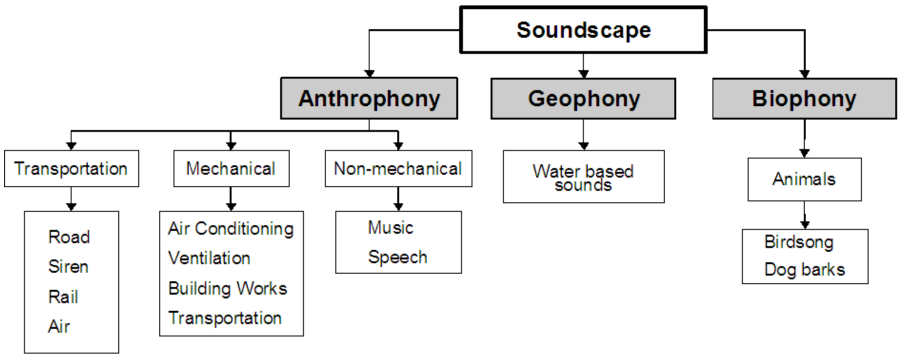


Figure 1: Acoustic taxonomy. The categories were derived from the sounds seen as having most influence on PPG 24 and BS 4142 measurements.

These sounds have been grouped into the main categories *anthrophony* (sounds related to human activities), *geophony* (sounds caused by nature), and *biophony* (sounds caused my animals). There are some sounds that may be seen as missing from this diagram. Under the Geophony category other natural based sounds, most of which are caused by the wind, could also be included. However, it was pointed out that when wind speeds are in excess of 5 m/s an acoustic consultant would often not perform a measurement as the noise induced on the sensor by the wind is too great. This situation will also apply to ISRIE so wind-induced sounds are not included. Other sounds considered missing from this diagram are not included because they are simply not loud enough to have an impact on a soundscape. The incidental sounds made by humans and animals are a good example of this. There is little in the literature regarding categorisation of urban audio signals. The reason for this may be that many sound classification projects focus on a small set of sounds or a particular species of animal. There are 2 good examples of categorisation in the literature. Raimbault and Dubois[2] use a tree-like structure which broadly splits the categories into *transportation/works* and *people presence*. These are then further subdivided to include some of the typical sounds found in an urban soudscape. It is somewhat strange to find the subcategories of *running water* and *birds singing* under *people presence* as these are not strictly due to the presence of humans. The other categorisation found in the literature is very similar to the idea generated at York[3] (Figure 1). The approach that Gage et al. use assigns specific frequency bands to each of anthrophony, geophony and biophony. This may cause a mis-categorisation because there are biophonic sounds that will fall below the 2.5 kHz low-end frequency of the biophony category.

2 CLASSIFICATION SYSTEMS

A classification system is typically a 2-stage process and consists of a feature extractor and a classifier[4]. The flow diagram in Figure 2 describes the basic structure of a classification system. Some systems also use post-processing after the classifier to correct for any errors in classification.

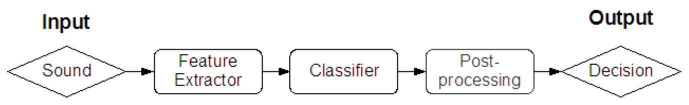


Figure 2: Flow diagram showing the basic structure of a classification system. Adapted from [5].

There are many feature extraction techniques and classifiers to choose from and the selection of each of these is usually based on the intended application and any prior knowledge. Table 1 gives examples of both feature extractors and classifiers found in the literature surrounding the topic of audio signal classification.

Feature Extractor	**Classifier**
Fourier transform (Fast/Short)	Multilayer Perceptron
Wavelet transform	Self-Organising Map
Wigner-Ville distribution	Learning Vector Quantisation
Time-Domain Signal Coding	Time Warping

Table 1: Some examples of feature extractors and classifiers often seen in the literature

2.1 Previous Studies of Sound Classification

There are many studies to be found in the literature which aim to perform classification of signals, whether they are audio signals or some other wave-based signals, such as an ECG signals. There are numerous studies which look at identifying different species of animal based on the sounds they emit, both incidental and deliberate sounds are considered. Some examples of animals identified are wood-boring insects[6], crickets[7], frogs[8], and birds[9]. Each of these studies show that it is possible to discriminate between very similar vocalisations which gives promise in the context of ISRIE as there will undoubtedly be similar sounds occurring in an urban environment. Cowling and Sitte[10] provide an excellent comparative examination of various feature extractors and classifiers for the recognition of environmental sounds. The aim of the study was to find which feature extractor-classifier pair provided the best classification results for a sound surveillance system. They found that using a continuous wavelet transform for feature extraction with a dynamic time warping classifier provided the best results (70% accuracy). In a novel approach to environmental sound classification[11] the goal is similar to that of ISRIE; to develop a system capable of giving an efficient representation of an acoustic environment. This is study is novel in that it uses a system based on genetic algorithms (GAs) to determine which features to extract from a sound. Using this method in combination with 2 different classifiers (a Gaussian mixture model and a k-Nearest Neighbour algorithm), classification results between 90% and 95% are achieved. No specific data is given but it is suspected that an approach using GAs will be computationally expensive and therefore not suited to ISRIE. A study looking at *sound textures*[12] uses a Self-Organising Map fed directly by the audio data without any feature extractor. The data is fed in 2^n ($1 < n < 4$) samples at a time and the SOM produces a histogram output. The theory behind this approach is that for a new input signal the trained SOM will produce a similar distribution to the distribution of the signal for which it was trained. This approach works very reliably to determine if a test signal is the same or different to the training signal but it cannot classify the test signal into a category.

All of the studies discussed above show that it is possible to perform accurate classification with a variety of audio signals. These can either be animal vocalisations or general audio signals such as those which may be found in an urban environment. In the following section the current approach to the classification problem being studied at York is described.

3 CURRENT APPROACH

The current approach being adopted at York is to use Time-Domain Signal Coding (TDSC) feature extraction and a Self-Organising Map (SOM) classifier. TDSC has been chosen because it is very computationally inexpensive and has had excellent success rates when used for classification of species[6,13]. A SOM has been chosen for use as an initial classifier partly due to its ease of use but also because it is easily expandable. If the SOM receives a signal it has not seen before it could potentially create a new output unit (and hence a new output class) for that input so that if a similar signal is presented it will be classified accordingly. TDSC is discussed in more detail below.

3.1 Time Domain Signal Coding

Time Domain Signal Coding is a purely time-domain technique based on a speech compression method known as Time Encoded Speech[13]. In TDSC analysis signals are segmented using zero crossings of the time-domain waveform. The data between successive zero crossings is termed an *epoch*. Each epoch can then be described by its shape (S - the number of minima) and its duration in samples (D) and a whole signal can then be described by its D-S pair characteristics. Figure 3 shows a simple example of D-S characteristics.

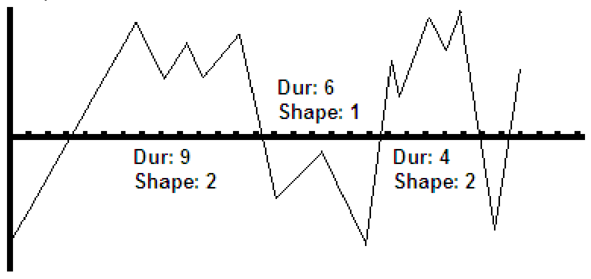


Figure 3: Simple example of D-S characteristics extracted using TDSC. The X-axis shows the sample intervals and the Y-axis represents amplitude.

In this example the first epoch has duration of 9 samples and a shape of 2. A signal analysed using TDSC will often have a large number of D-S pairs. These can be mapped onto a smaller symbol set (termed the *codebook*) to make processing easier. The codebook can then be used to generate either a 1- or 2-dimensional histogram describing the occurrence of D-S pairs. The 1-dimensional variant is called the *S-matrix* where the X-axis represents the code and the Y-axis the frequency of occurrence for each code.

3.1.1 Duration-Shape Distribution

In previous studies using TDSC the codebook has been generated manually to suit the application containing some 30 codes[6,13]. However, this is not possible for the application to the general sounds found in an urban environment (shown in Figure 1) due to the large variation in D-S distributions. To identify how much these distributions varied by, plots were generated of D versus S as shown in Figure 4. These plots were resized to show the areas containing the most information as the D-S distributions form very sparse matrices. The maximum D-S pairings found for each of these sounds were: A/C unit D=1468 S=165, Blackbird D=218 S=56, Digger D=434 S=68, and Human speech

D=1086 S=161. Using the speech recording as an example, there was 1 occurrence of the maximum D-S pair whereas the most frequently occurring D-S pair (D=2 S=0) was detected 9813 times. This illustrates that some sounds have anomalies present in the recordings which may not be characteristic of that sound.

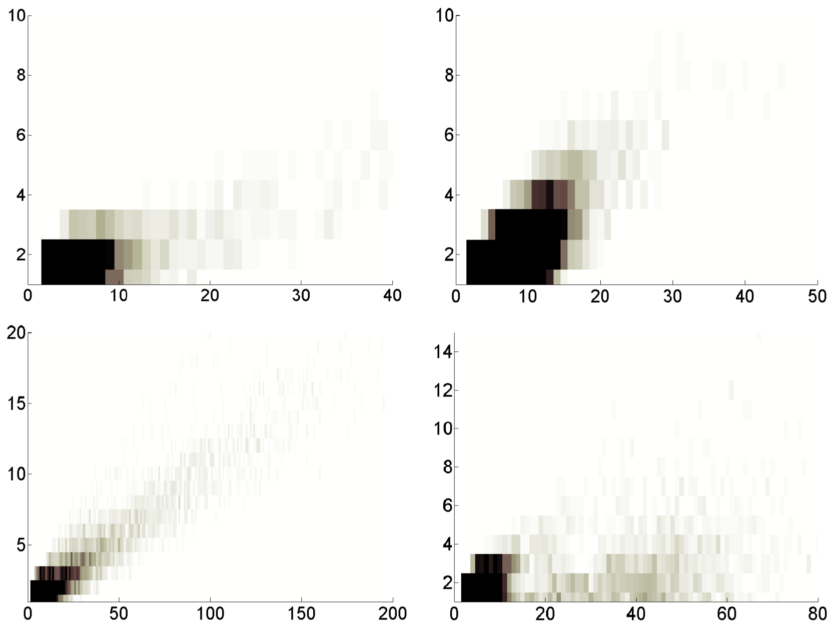


Figure 4: Distributions showing D (X-axis) versus S+1 (Y-axis). Clockwise from the top-left, the plots represent the following sounds: an air conditioning unit, a blackbird singing, human speech and a digger. The darker areas represent higher frequencies of occurrence for that D-S pair. *Recordings of the A/C unit and speech courtesy of ISVR, Southampton. All recordings sampled at 44.1 kHz.*

3.1.2 Generation of a general codebook

Based on the findings of plotting D-S distributions for a number of sounds it was decided that the maximum duration that would be useful to detect would be 1000 (using a sample rate of 44.1 kHz) and the maximum shape would be 75. This gave a very large total number of D-S pair combinations (>50,000). To reduce this number 40 duration ranges were devised based on the findings from the D-S distributions. The first 19 of these ranges had a range of 1 as most D-S data occurs within D<20. The subsequent ranges spanned durations of 10, 50 and 100 samples. After calculating these ranges the total number of D-S pair combinations was significantly reduced to 1700.

3.2 Classification of the Audio Data

Initial research has focussed on classifying audio recordings of the sounds given in Figure 1 using TDSC for feature extraction and a SOM to classify the resulting time domain data. Each audio sample was typically of 2 seconds in duration and analysed in 0.2 second frames. Each frame then had an S-matrix associated to it and these were used as the input to the SOM. Figure 5 illustrates this process.

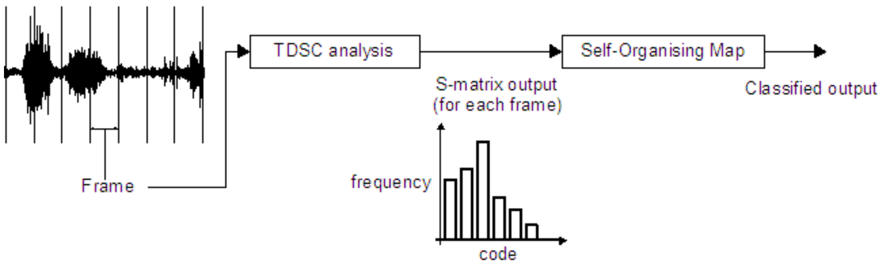


Figure 5: The classification process

During training of the SOM it was noticed that the winning unit was consistently either the highest or lowest numbered unit in the network. This result was not dependant on the input TDSC data; each input gave the same result. The SOM was tested with a smaller, controlled data set consisting of only one value and it performed as expected; the weights for one particular unit in the network approached the value of the input data after a number of training epochs. This showed that the SOM network was functioning as expected. Attention was then turned onto the S-matrices being produced by the TDSC algorithm. Figure 6 shows a plot of how the data is spread out through the codebook. It is quite clear to see that the arrays containing the code frequencies are very sparse. Analysis of the actual data contained in the arrays has shown that the number of cells with a value of zero averages 95%. This figure is similar for a variety of audio data typical to an urban environment. It is believed that this sparseness is the reason for the SOM continually training its weights toward zero and producing the same winning unit even for different initial audio data.

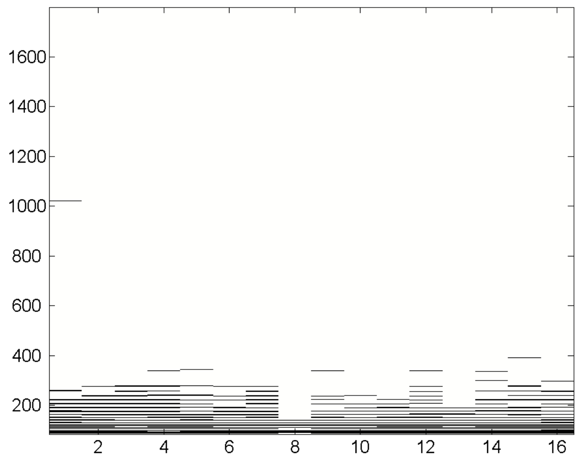


Figure 6: A plot showing the sparseness of the output for TDSC analysis for a recording of a blackbird. The X-axis shows the code number, the Y-axis shows the time frame number and the dark areas show which codes occur in each frame.

4 CONCLUSIONS AND FURTHER WORK

This paper has presented the current work on signal classification as part of the overall ISRIE project. What the ISRIE project is and what it is aiming to achieve has been described. Via discussions with project partners and analysis of PPG 24 and BS 4142 the set of sound categories presented in this paper has been defined to classify audio data into. Prior to discussing the current approach to signal classification a short description of how a classification system is typically constructed was given along with a brief review of previous research in the related areas.

The approach currently being used for signal classification has been described. This consisted of a time-domain signal coding (TDSC) feature extractor producing S-matrix data as an input for a self-organising map (SOM). It was clear to see from the distributions given in Figure 4 how D-S data varies from one sound to another showing that TDSC is a useful feature extractor. It is also clear to see from these distributions that the majority of D-S data is contained in the region D<25 S<10. It was shown that the general codebook generates very sparse arrays of data causing erroneous classification by the SOM. It is therefore necessary to review the arrangement of this general codebook to better describe the salient features of the D-S distributions in the region given above. Other methods of compressing the D-S information into a codebook to reduce the array sparseness will be investigated.

4.1 Further Work

Research will continue using the TDSC and SOM classification system described above. Other feature extraction methods are also due to be studied and tested, Wavelet transforms are of particular interest. A comparison of the classification accuracies produced will be made to see if any improvement can be had by using a different feature extractor.

It is intended that the classification process shown in Figure 5 will be expanded by using the output of the SOM as an input to a syntactic pattern recognition system (SPR - see [14] for an introduction to syntactic methods). The combination of a TDSC feature extractor and SOM classifier can produce an output class variation with repeating patterns. This is illustrated in Figure 7 which used a recording of a cricket as the input to the TDSC-SOM classifier. The change from class 1 to class 4 occurs in a repeating pattern throughout in line with the song of the cricket and the silence between chirps. It has been proposed that this sort of data could be used as the grammar in a syntactic pattern recognition system. Further analysis of the SOM output for various audio recordings will demonstrate if a SPR approach is a feasible solution in the context of the ISRIE project.

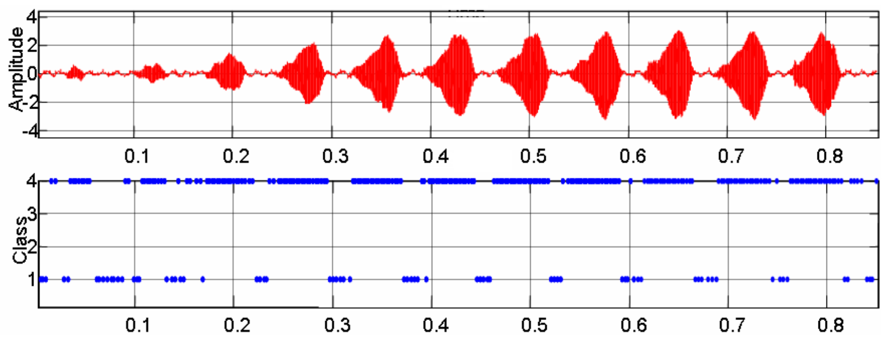
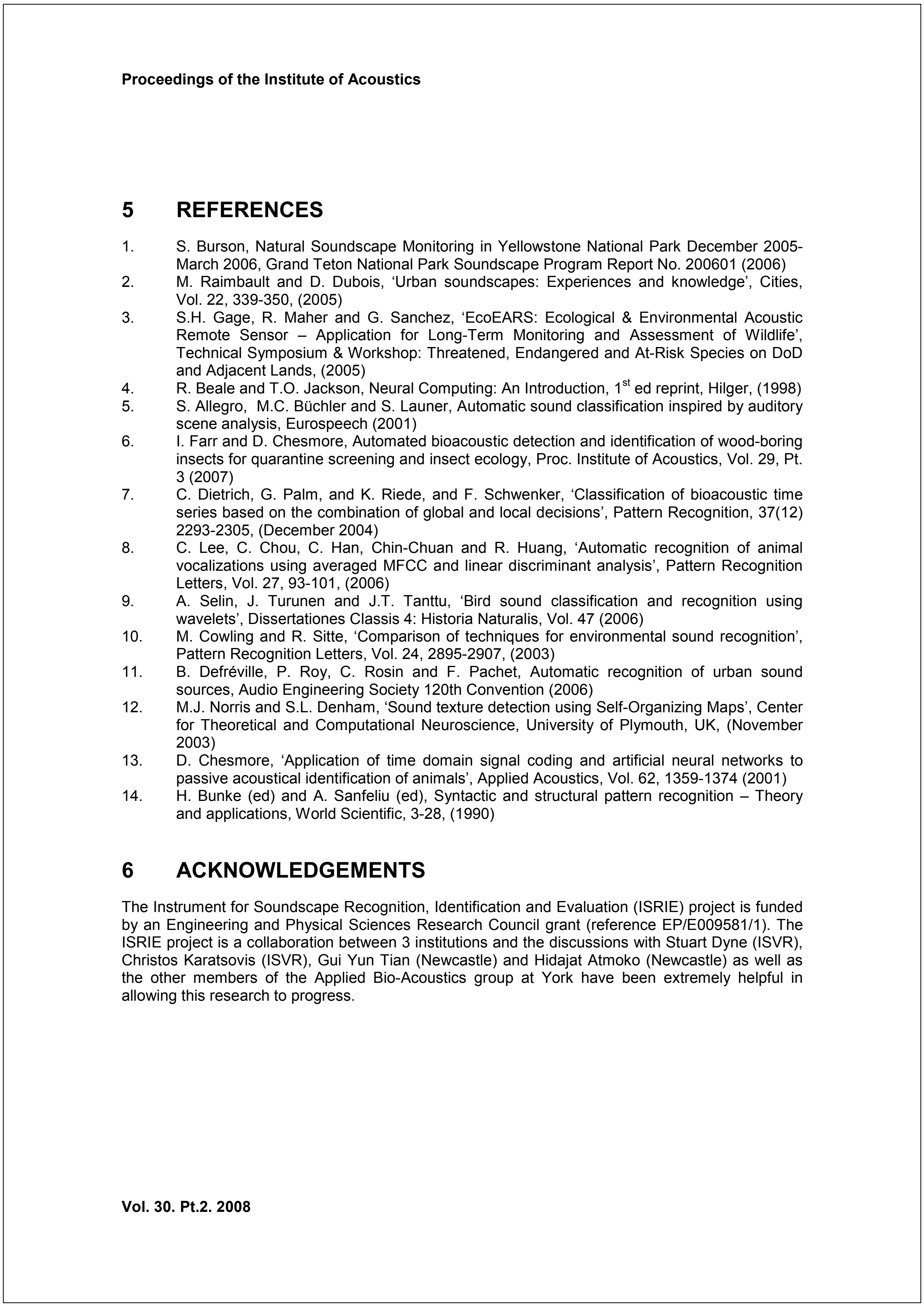


Figure 7: The top diagram shows the time-domain waveform of the cricket song. The lower diagram shows the pattern of the class output as the signal changes.



5 REFERENCES

1. S. Burson, Natural Soundscape Monitoring in Yellowstone National Park December 2005-March 2006, Grand Teton National Park Soundscape Program Report No. 200601 (2006)
2. M. Raimbault and D. Dubois, ‘Urban soundscapes: Experiences and knowledge’, Cities, Vol. 22, 339-350, (2005)
3. S.H. Gage, R. Maher and G. Sanchez, ‘EcoEARS: Ecological & Environmental Acoustic Remote Sensor – Application for Long-Term Monitoring and Assessment of Wildlife’, Technical Symposium & Workshop: Threatened, Endangered and At-Risk Species on DoD and Adjacent Lands, (2005)
4. R. Beale and T.O. Jackson, Neural Computing: An Introduction, 1st ed reprint, Hilger, (1998)
5. S. Allegro, M.C. Büchler and S. Launer, Automatic sound classification inspired by auditory scene analysis, Eurospeech (2001)
6. I. Farr and D. Chesmore, Automated bioacoustic detection and identification of wood-boring insects for quarantine screening and insect ecology, Proc. Institute of Acoustics, Vol. 29, Pt. 3 (2007)
7. C. Dietrich, G. Palm, and K. Riede, and F. Schwenker, ‘Classification of bioacoustic time series based on the combination of global and local decisions’, Pattern Recognition, 37(12) 2293-2305, (December 2004)
8. C. Lee, C. Chou, C. Han, Chin-Chuan and R. Huang, ‘Automatic recognition of animal vocalizations using averaged MFCC and linear discriminant analysis’, Pattern Recognition Letters, Vol. 27, 93-101, (2006)
9. A. Selin, J. Turunen and J.T. Tanttu, ‘Bird sound classification and recognition using wavelets’, Dissertationes Classis 4: Historia Naturalis, Vol. 47 (2006)
10. M. Cowling and R. Sitte, ‘Comparison of techniques for environmental sound recognition’, Pattern Recognition Letters, Vol. 24, 2895-2907, (2003)
11. B. Defréville, P. Roy, C. Rosin and F. Pachet, Automatic recognition of urban sound sources, Audio Engineering Society 120th Convention (2006)
12. M.J. Norris and S.L. Denham, ‘Sound texture detection using Self-Organizing Maps’, Center for Theoretical and Computational Neuroscience, University of Plymouth, UK, (November 2003)
13. D. Chesmore, ‘Application of time domain signal coding and artificial neural networks to passive acoustical identification of animals’, Applied Acoustics, Vol. 62, 1359-1374 (2001)
14. H. Bunke (ed) and A. Sanfeliu (ed), Syntactic and structural pattern recognition – Theory and applications, World Scientific, 3-28, (1990)

6 ACKNOWLEDGEMENTS

The Instrument for Soundscape Recognition, Identification and Evaluation (ISRIE) project is funded by an Engineering and Physical Sciences Research Council grant (reference EP/E009581/1). The ISRIE project is a collaboration between 3 institutions and the discussions with Stuart Dyne (ISVR), Christos Karatsovis (ISVR), Gui Yun Tian (Newcastle) and Hidajat Atmoko (Newcastle) as well as the other members of the Applied Bio-Acoustics group at York have been extremely helpful in allowing this research to progress.

Acoustics'08 Paris
June 29-July 4, 2008
www.acoustics08-paris.org

Instrument for Soundscape Recognition, Identification and Evaluation (ISRIE): Signal Classification

J. Stammers and D. Chesmore

University of York, Department of Electronics, Heslington, YO10 5DD York, UK
js185@york.ac.uk

ISRIE is a collaborative project between the universities of York and Newcastle and ISVR in Southampton. The work being undertaken at York is in its second year and focuses on signal separation and classification. Developing novel methods for classifying urban and other sounds into distinct categories (such as transportation, industrial, human, animal, etc.) is the focus of the work detailed in this paper. The classification system will initially consist of 2 main parts: a feature extractor and a classifier. Results from this basic system will be presented and a discussion given on how the system will be expanded. It is envisaged that eventually the system will use some form of syntactic pattern recognition to perform the identification of individual sounds.

1 Introduction

The ISRIE project arose from the EPSRC Ideas Factory 'A Noisy Future'. The proposed outcome of the project can be briefly described as an intelligent noise metering system able to determine the direction and source from which a sound originated. It will also be able to provide other details such as the time at which the sound occurred and how loud the sound was. If a number of these instruments are used in a sensor network it should be possible to estimate the location of the sound source. Such an instrument would be a useful tool in urban noise level measurements, either for research or for legislative purposes. This can be a very time consuming process when performed manually and can also be subjective if soundscape content is also being examined. More details on the ISRIE project and its application to legislative procedures can be found in [1].

1.1 Sound Categories

The signal classification part of the ISRIE project has identified the key sounds that are to be recognised within an urban soundscape. These are shown in Fig. 1. The decision to focus on these sounds for the final system was based on discussions with project partners and current noise legislation in the UK.

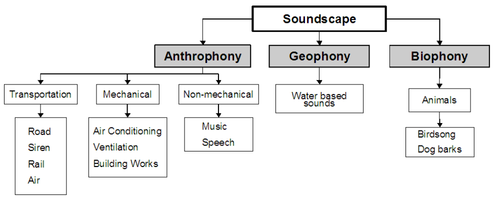


Fig. 1 The relationship between the key sounds to be identified.

The soundscape has been split into 3 main categories: *Anthrophony*, relating to sounds made by humans; *Geophony*, naturally occuring sounds; and *Biophony*, sounds made by animals. Only the most prevalent sounds that would be heard in an urban soundscape have been included. So under the category of *biophony* only birdsong and the bark of a dog have been included because other animal sounds are unlikely to exceed the background noise level of an urban environment. Other sounds that could be included under *geophony* are likely to be caused by the interaction between an object and the wind (a tree, for example). These sounds are not included in the diagram because it was pointed out that when wind speeds exceed 5 m/s an acoustic measurement is unlikely to be taken because of the noise induced on the sensor by the wind. A similar method of breaking down the soundscape has been seen in [2]. The approach taken by Gage et al. applied frequency divisions to separate the 3 main categories. This could lead to mis-classification as not all sounds found in each of the categories will necessarily adhere to the frequency bands.

2 Classification Systems

Classification systems typically consist of 2 main components – a feature extractor and a classifier [3]. The role of the feature extractor is to reduce the complexity of the data being input to the classifier to optimise the classifying process [4]. There are many examples in the literature of classification systems for the analysis and classification of both audio and other wave-based signals. The range of techniques used and applications vary considerably from wavelet feature extraction and multi-layer perceptron network classifiers for human bowel-sound monitoring [5] to time-domain and Mel-frequency techniques for species identification [6,7,8]. The area of environmental sound analysis has also had a lot of development. Cowling and Sitte [9] provide an excellent overview of techniques as applied to a sonic security system. Their research found that a continuous wavelet transform feature extractor coupled with a dynamic time warping classifier gave the highest recognition accuracy (70%). A novel approach to environmental sound recognition is found in the work of Defréville et al. [10]. Their work focussed on using genetic algorithms to find problem-specific features for each individual signal. The results of this method are promising (~90% accuracy) but the signal processing techniques discovered are very complex.

To date, of the many feature extractor and classifier methods available Time-Domain Signal Coding for feature extraction and a Self-Organising Map have been implemented to make up the classification system.

Time-Domain Signal Coding (TDSC) is a feature extraction technique which focuses purely on the time-domain representation of an audio signal. The waveform is seperated into epochs (the signal data between two consecutive zero crossings) and each of these are analysed in terms of shape (S) and duration (D). The shape of an epoch is determined by how many positive or negative minima it contains and the duration is simply the length of the epoch in samples. Further details of how TDSC is performed can be found in [6,7]. TDSC has previously been used for monitoring of machinery and heart sound analysis [6]. In its application to species recognition TDSC has achieved 100% classification accuracy for 13 different Cricket species.

3 Application of TDSC

It was mentioned above that a TDSC feature extractor was coupled with a Self-Organising Map (SOM) classifier for the initial system development. Details of SOMs and their implementation can be found in [11]. The main focus of this work has so far been to produce an output from the TDSC algorithm which is suitable for classification by the SOM. The duration-shape (D-S) information gathered by TDSC is typically organised into a codebook representing a range D-S combinations. The S-matrix is an array of data which associates a frequncy of occurrence to each of these combinations. In previous studies using TDSC the codebook has been manually designed for the application, typically giving ~30 codes. To generate a suitable codebook for containing urban sound data distributions were produced of D-S combinations for various sounds. Fig. 2 shows an example of one such distribution.

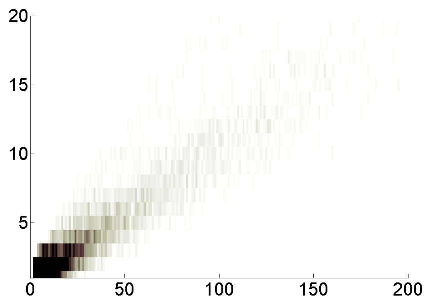


Fig. 2: D-S distribution for a recording of a building site digger. The x-axis represents D and the y-axis represents S.

The maximum D-S pairing found was for an air conditioning unit with D=1468 and S=165. Based on this and other results it was decided to limit D to 1000 and S to 75. Using all possible combinations of these D and S ranges would give a codebook with a very large order of magnitude (>50,000). This number was reduced to 1700 by deviding duration ranges to fit the data into.

Fig. 3 shows the initial classification system used. The audio signal was broken into frames and each of these were analysed seperately using the TDSC algorithms. The TDSC output data for each frame was then classified by the SOM to give a class output for each frame.

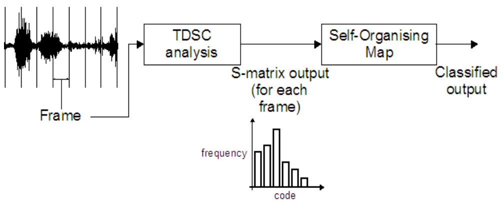


Fig. 3: The arrangement of the initial classification system.

Initial results using the codebook discussed above were disappointing. The SOM consistently gave the same winning units for all sounds. Upon analysis of the TDSC output it was found that the S-matrices had an average sparseness of 95%. For this reason the SOM was struggling to differentiate between the S-matrices generated for audibly different signals.

Further inspection of D-S distributions (Fig. 2) showed that further reduction of the maximum D and S values was possible without sacrificing significant amounts of data. A codebook with a size of 340 was achieved using D=150 and S=15. Results using this codebook are presented below using recordings of sounds found in an urban setting and recordings of some Cicadas.

3.1 Cicada Classification

High quality recordings of 3 different species of Cicada were made available to test the system. It was decided to experiment with these recordings for two reasons: a) the Cicadas are difficult to differentiate by ear so it would be a good test to see if the system could; and b) there is interest in developing a real-time system for the identification of different Cicada species. A total of 24 recordings were used – 3 in which the species were known (training set) and 21 unknown (for testing). The framelength used in the TDSC analysis was 0.2 seconds. It was decided to use a 10 unit SOM for classification.

Fig. 4 shows a plot of the class outputs for each frame of the known recordings. It is clear to see from this plot that a very simple decision rule (perhaps based on an LVQ method) would allow seperation of the 3 different species of Cicada. *Flamatus* appears only in classes 1-3, *japonicus* in classes 4-6 and *biahamatus* dominates classes 8-10. Fig. 4 also shows the class outputs for one of the unknown recordings. By visual inspection it is clear that this particular recording would be placed in the *bihamatus* category. Overall, the system comprising of a TDSC feature extractor and a SOM classifier achieved a classification accuracy of 95%.

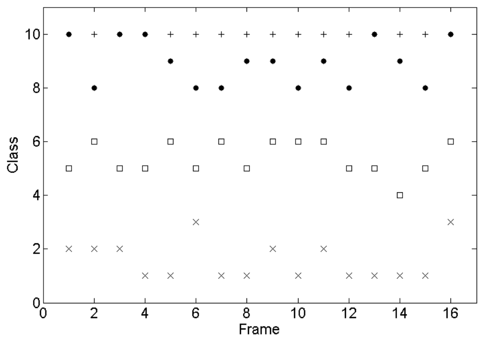


Fig. 4: Frame-wise representation of SOM class output. The different classes are *bihamatus* (●), *japonicus* (□), *flamatus* (x) and test data (+).

3.2 Urban sound classification

Using the same system as that described above, classification of urban sounds was experimented with. The recordings used were of some of the sounds given in Fig. 1

(air conditioning unit, single motor vehicle, birdsong and building works). There were not as many recordings of each of these sounds available as there were for the Cicadas but the resulting data discussed below is still useful.

The recordings of each type of sound were seperated into 6 second sections (of the available data this provided 2 or 3 different recordings for testing). Initially a framelength of 0.1 seconds was chosen. As little is known of the significance of framelength in this application, 0.1 seconds was chosen as a starting value. The same theory applies to the number of output classes chosen for the SOM . In this instance 40 classes were used.

A plot of class output for each frame of a building site recording is shown in Fig. 5. The prominent sound in this recording was a large caterpillar-track-driven digger interspersed with some road noise.

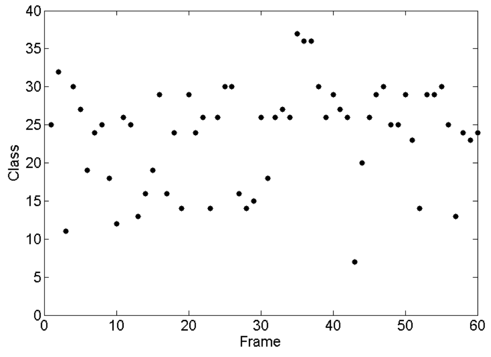


Fig. 5: Framewise SOM class output for a 6 second building site recording.

It is clear to see from this plot that there is no clear banding of class output as there was for the Cicada recordings. Plots for the other sounds listed above produced very similar results, i.e. no obvious class dominance. These disappointing results influenced the decision to start looking at how framelength affected the SOM class output plots. Reducing the framelength had the effect of increasing the apparent lack of structure seen in Fig. 5. Increasing the framelength to 0.5 seconds produced plots that were more promising. Fig. 6 shows the result of using the longer framelength with the same recording as used to produce Fig. 5 and another recording of a similar soundscape.

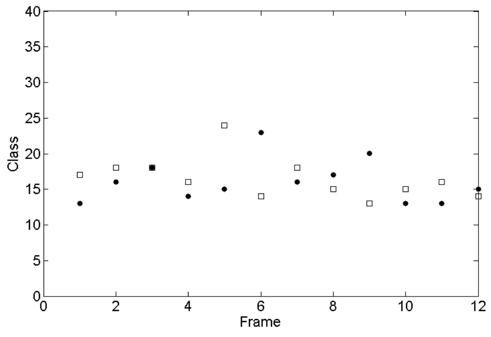


Fig. 6: SOM class output for 2 similar building site recordings; (●) uses the same audio data as that in Fig. 5 and (□) is shown for comparitive purposes.

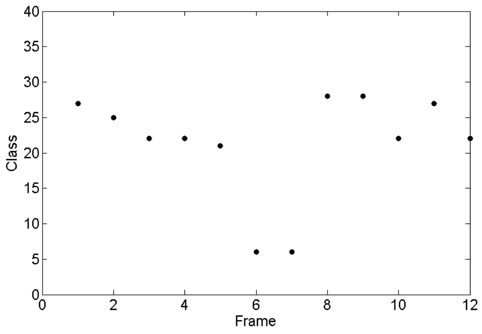


Fig. 7: SOM class output for a blackbird recording.

Using a longer framelength does seem to have a positive affect on the SOM output. Visual analysis of Fig. 6 shows that both of the building site recordings mostly produce outputs in the 12-20 class region. Figure 7 shows the class outputs for a blackbird recording and is included for comparison to the building site output. The class range for the blackbird recording is mostly 20-27, different ot that of the building site.

Converting the class output data for the building site into a histogram shows a distinct tendency for the class range stated (see Fig. 8). Similar SOM output histograms were achieved for the other urban sounds when analysed using a framelength of 0.5 seconds. The Cicada recordings produced results in line with those discussed in Section 3.1 showing that increasing the framelength to 0.5 seconds does not have an adverse affect on their classification.

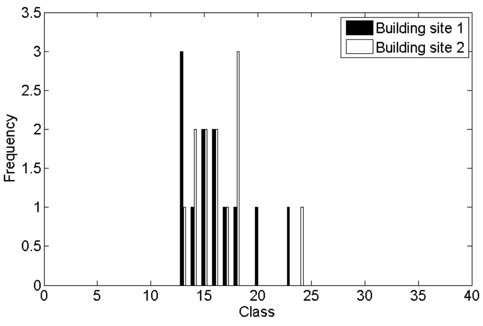


Fig. 8: Histogram of SOM class output for 2 building site recordings using a framelength of 0.5s for TDSC analysis.

3.3 Further Classification Work

The above findings are encouraging for the development of a system able to distinguish between various urban sounds (and species specific sounds). Further work in this area will initially focus on retrieval and analysis of more urban sound recordings. This will enable validation of the above results and to see if this approach shows promise for a broader range of audio data. SOM output class histograms for the new recordings will also be generated to discover any trends that may be present. It could also be possible to use these histograms as an input to another classifier to see if a clear distinction can be found.

One direction for development of the classification system which is of particular interest would be to implement syntactic methods. Syntactic pattern recognition (SPR) involves breaking data into its basic building blocks, known as pattern primitives, and devising a grammar for a data set [12]. In the case of speech analysis (an area in which SPR is often used) the pattern primitives and grammar are fairly obvious. To use syntactic techniques for urban sound classification the pattern primitives will have to be devised based on the data available. From the plots given in Figures 4-6 there seem to be two options for pattern primitives; either the actual class outputs and how these follow each other, or applying trends to how the classes change with time. Once some pattern primitives have been decided upon a suitable classifier is then required to analyse these. A hidden Markov model (HMM) classifier could be the solution. HMMs are based on a state machine structure where the transition from the current state to the next has a probability associated with it [13]. A HMM will need training like any other classifier to determine the state transition probabilities. There are other possibilities for SPR classifiers but HMMs will be considered in the first instance because previous studies have shown they can be used for classification of everyday sonic environments [14].

The further work described above expands on the system structure shown in Fig. 3 by making the TDSC feature extractor and SOM classifier combination a preprocessor for further classification.

4 Conclusions

This paper has discussed the current work on signal classification for the ISRIE project. A system has been described consisting of a Time-Domain Signal Coding feature extractor and a Self-Organising Map classifier. A suitable TDSC codebook has been developed for use with urban audio signals and the current version of the codebook has improved significantly on the original. The effect of framelength on the SOM output classes has been investigated. From the results given a framelength of 0.5 seconds has shown the best results so far. Increasing the framelength further may have the effect of averaging the results too much and there will be no discernible difference between sources.

Suggestions for further work have been made which investigate the potential for using syntactic methods in the classification process and how the current implementation can be improved upon. The current TDSC/SOM combination will become a preprocessing unit for any expanded system that is developed.

Acknowledgments

The Instrument for Soundscape Recognition, Identification and Evaluation (ISRIE) project is funded by the Engineering and Physical Sciences Research Council (EP/E009581/1). The authors would like to thank the ISRIE project collaborators (Stuart Dyne and Christos Karatsovis, ISVR Southampton and Gui Yun Tian and Hidajat Atmoko, University of Newcastle) and other members of the Applied Bioacoustics group at York for their discussions and input relating to this work.

References

[1] C.Karatsovis, S. Dyne, "Instrument for soundscape recognition, identification and evaluation: an overview and potential use in legislative applications", *Proceeding of the Institute of Acoustics*, 602-608 (2008)

[2] S.H. Gage, R. Maher, G. Snachez, "EcoEARS – Application for Long-Term Monitoring and Assesment of Wildlife", In *Technical Symposium & Workshop: Threatened, Endangered and At-Risk Species on DoD and Adjacent Lands* (2005)

[3] R. Beale, T.O. Jackson, "Neural Computing: An Introduction", 1st ed. Reprint, Hilger (1998)

[4] N. Beltran, M. Duarte-Mermoud, M. Bustos, S. Salah, E. Loyola, A. Peña-Neira, J. Jalocha, "Feature extraction and classification of chilean wines", *Journal of Food Engineering* 75, 1-10 (2005)

[5] C. Dimoulas, G. Kalliris, G. Papanikolaou, V. Petridis, A. Kalampakas, "Bowel-sound pattern analysis using wavelets and neural networks with application to long-term, unsupervised, gastrointestinal motility monitoring", *Expert Systems with Applications* 34, 26-31 (2008)

[6] D. Chesmore, "Application of time domain signal coding and artificial neural networks to passive acoustical identification of animals", *Applied Acoustics* 62, 1359-1374 (2001)

[7] I. Farr, D. Chesmore, "Automated bioacoustic detection and identification of wood-boring insects for quarantine screening and insect ecology", *Proceedings of the Institute of Acoustics* 29, Pt. 3 (2007)

[8] C.H. Lee, C.H Chou, C.C. Han, R.Z. Huang, "Automatic recognition of animal vocalizations using averaged MFCC and linear discriminant analysis", *Pattern Recognition Letters* 27, 93-101 (2006)

[9] M. Cowling and R. Sitte, "Comparison of techniques for environmental sound recognition", *Pattern Recognition Letters* 24, 2895-2907 (2003)

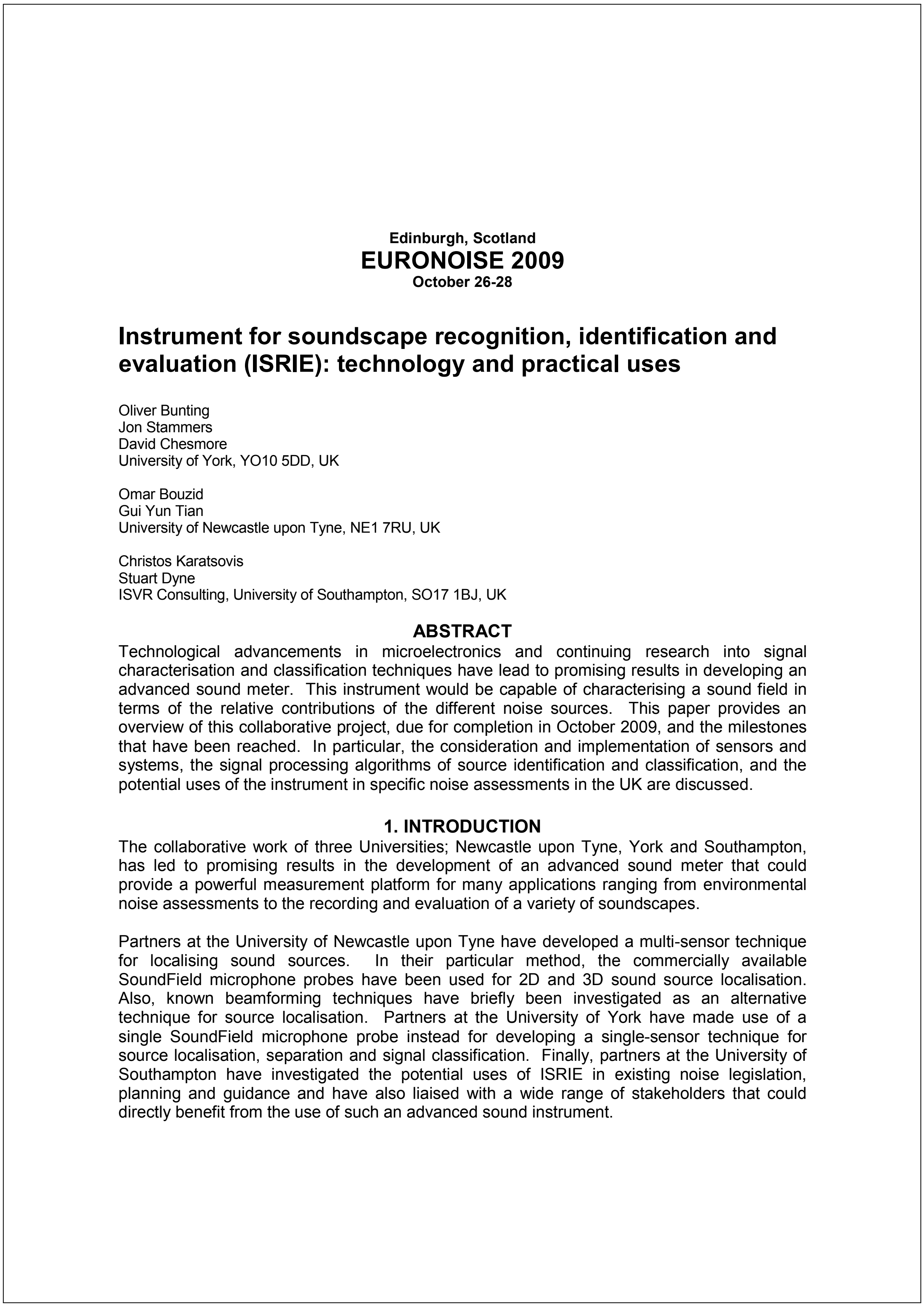
[10] B. Defréville, P. Roy, C, Rosin, F. Pachet, "Automatic recognition of urban sound sources", *Audio Engineering Society 120th Convention* (2006)

[11] F.M. Ham and I. Kostanic, "Principles of Neurocomputing for Science & Engineering", McGraw-Hill (2001)

[12] K.S. Fu, "Syntactic Methods for Pattern Recognition", Academic Press (1974)

[13] L.R. Rabiner, "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition", *Proceedings of the IEEE* 77, No. 2 (1989)

[14] L. Ma, B. Milner, D. Smith, "Acoustic Environment Classification", *ACM Transactions on Speech and Language Processing* 3, No. 2 (2006)



Edinburgh, Scotland

EURONOISE 2009

October 26-28

Instrument for soundscape recognition, identification and evaluation (ISRIE): technology and practical uses

Oliver Bunting
Jon Stammers
David Chesmore
University of York, YO10 5DD, UK

Omar Bouzid
Gui Yun Tian
University of Newcastle upon Tyne, NE1 7RU, UK

Christos Karatsovis
Stuart Dyne
ISVR Consulting, University of Southampton, SO17 1BJ, UK

ABSTRACT

Technological advancements in microelectronics and continuing research into signal characterisation and classification techniques have lead to promising results in developing an advanced sound meter. This instrument would be capable of characterising a sound field in terms of the relative contributions of the different noise sources. This paper provides an overview of this collaborative project, due for completion in October 2009, and the milestones that have been reached. In particular, the consideration and implementation of sensors and systems, the signal processing algorithms of source identification and classification, and the potential uses of the instrument in specific noise assessments in the UK are discussed.

1. INTRODUCTION

The collaborative work of three Universities; Newcastle upon Tyne, York and Southampton, has led to promising results in the development of an advanced sound meter that could provide a powerful measurement platform for many applications ranging from environmental noise assessments to the recording and evaluation of a variety of soundscapes.

Partners at the University of Newcastle upon Tyne have developed a multi-sensor technique for localising sound sources. In their particular method, the commercially available SoundField microphone probes have been used for 2D and 3D sound source localisation. Also, known beamforming techniques have briefly been investigated as an alternative technique for source localisation. Partners at the University of York have made use of a single SoundField microphone probe instead for developing a single-sensor technique for source localisation, separation and signal classification. Finally, partners at the University of Southampton have investigated the potential uses of ISRIE in existing noise legislation, planning and guidance and have also liaised with a wide range of stakeholders that could directly benefit from the use of such an advanced sound instrument.

2. ACOUSTIC SOURCE LOCALISATION

Over the course of the ISRIE project the co-authors at Newcastle University implemented an acoustic localisation system that is capable of locating a single sound source using at least three omni-directional microphones (i.e. 2D linear arrays) in a reverberant indoor environment with high accuracy for angle detection and small errors for distance estimation[1]. Sound source localisation in a 3D environment has been achieved by utilising the commercially available SoundField probes.

Figure 1 shows the use of three acoustic sensors in the context of a sound localisation system.

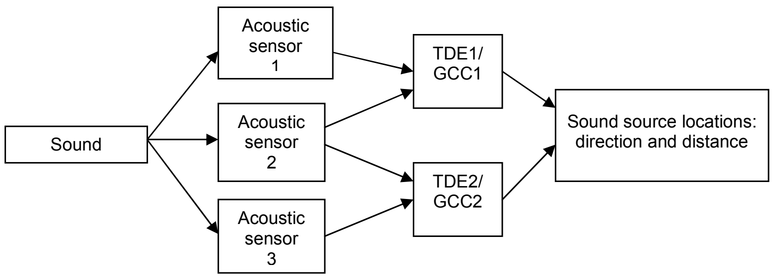


Figure 1: Three-microphone array system for acoustic monitoring[1].

The three acoustic sensors (omni-directional or 3D SoundField microphones) capture the sound simultaneously and the Time Delay Estimation (TDE) is extracted from any two sound signals from the three sensors using the Generalized Cross-Correlation (GCC). This method would ultimately derive sound source direction and distance through triangulation and geometric parameters. The three microphones are positioned in a straight line and the sides of the triangles formed by the source and each microphone represent the directional propagation paths from the source to each microphone. The direction of each propagation path is determined from the time differences between the signals arriving at the microphones. GCC is used to increase robustness to the adverse effects of early reflections and reverberation.

A. The 3 SoundField Microphone Method

Three SoundField SPS422B microphones were arranged in a straight line in order to achieve source localisation in a 3D environment[1]. Each microphone output is formed into a special signal format, the B-format, where four channels represent the velocity component in the three Cartesian directions; X (front-back), Y (left-right), Z (above-below) and one omni-directional signal, W, representing the pressure component. These signals are then fed into a PC for post-processing.

The Y and Z channel will generally be the same due the linear arrangement of the probes. The 2D configuration can be used for tilt and yaw estimation of sound direction in 3D. The X and W were therefore used for estimation in the experiment. With this arrangement, it has been possible to locate a single sound source in a reverberant indoor environment with an accuracy of 1° for angle detection and errors less than 4% for distance estimation. A rearrangement of the soundfield array in the Z Cartesian direction was tested in order to provide estimates of yaw instead of azimuth angles. The W and Z microphone outputs were used for the estimation and the results were similar. The SoundField probes could therefore potentially be used in a commercial source localisation system, where the sensitivity of these microphones to sounds arriving from different directions will be applied to source localisation in planes other than that defined by the line of the array.

B. Beamforming Techniques

In the literature, beamforming is another suggested technique that has extensively been used in developing instruments for soundscape recognition, identification and sound source localisation[2, 3]. The beamforming technique is a technique that searches for a peak (or peaks) by achieving a full directional scan in order to determine the source(s) direction(s) from this (or these) peak(s). This can be achieved by delaying and summing the acoustic emitted signals to minimise the noise effects and enhancing (or maximising) the amplitude of the point (or direction) that represents the location of the sound source[2, 3]. The sound source can be considered to be in the near-field if the wavefront is modelled as spherical, whereas it is considered to be in the far-field if it is assumed to be planar[3]. The consequences of these assumptions are that in the near-field both the range and Direction of Arrival (DOA) can be computed, whereas in the far-field, only the DOA can be estimated due to computational costs[3]. Li[3] designed a flexible broad-band beamformer using nested Concentric Ring Array (CRA) that can be divided into sub arrays, where each sub array can cover a specified operating range. In our study, the acoustic camera, which mainly includes a microphone array of Star 36 sensors[4], a data-reader device, a laptop computer and the "NoiseImage" software[4], has been used for the investigation on flexible beamforming techniques and instrument validation. The data from this study is currently under investigation.

3. SOURCE SEPARATION

The task of automated recognition of audio signals is made considerably more complex by multiple sources being present in the audio recording, with a consequent reduction in recognition accuracy rates. To provide enhanced recognition accuracy, ISRIE employs a source separation algorithm prior to the recognition stages. The separation method developed for ISRIE is based on the assumption of W-disjoint orthogonality. That is, audio sources are sparse in a time-frequency domain. The sensor used is a Soundfield ST350, a B-format coincident microphone array[5, 6] that offers a more portable microphone system over the SPS422B.

A. Model

Consider a 3-dimensional coincident array comprising of 3 orthogonal sets of figure-of-eight microphones and an omni-directional microphone at the centre of the array. Given the location of the sources, the B-format mixture of signals in the anechoic case can be expressed as:

$$\begin{vmatrix} w(t) \\ x(t) \\ y(t) \\ z(t) \end{vmatrix} = \begin{vmatrix} 1/\sqrt{2} & \ldots & 1/\sqrt{2} \\ \cos(\theta_1)\cos(\lambda_1) & \ldots & \cos(\theta_N)\cos(\lambda_N) \\ \sin(\theta_1)\cos(\lambda_1) & \ldots & \sin(\theta_N)\cos(\lambda_N) \\ \sin(\lambda_1) & \ldots & \sin(\lambda_N) \end{vmatrix} \begin{vmatrix} s_1(t) \\ . \\ . \\ . \\ s_N(t) \end{vmatrix} \qquad (1)$$

where x, y, z are the mixtures observed on the Cartesian axis, w is the mixture observed by the omni-directional sensor, and θ, λ are the azimuth and elevation for the direction of arrival of a particular source.

B. Assumptions
Separation of a given mixture is subject to two conditions on the source mixture being met. These are W-disjoint orthogonality[7] and radial sparsity. These are described formally below.

W-disjoint Orthogonality
Two sources s_i and s_j are W-disjoint orthogonal if the following condition is met.

$$S_i(\omega,\tau)S_j(\omega,\tau) = 0 \qquad \forall\ i \neq j, \omega, \tau \qquad (2)$$

where $S(\omega,\tau)$ represents the time-frequency domain transformation of $s(t)$.

Radial Sparsity
This a condition placed on the geographical location of the sources. Each source must have a unique direction of arrival at the sensor.

$$(\theta_i, \lambda_i) \neq (\theta_j, \lambda_j) \qquad \forall\ i \neq j \qquad (3)$$

C. Direction of Arrival (DOA) Calculation
Provided the above conditions have been met, the DOA of the B-format audio signal can be calculated in the time-frequency domain using a method from Directional Audio Coding Scheme (DirAC)[8, 9].

$$\vec{D}(\omega,\tau) = -\Re\left(W^*(\omega,\tau) * \left(\vec{e}_x X(\omega,\tau) + \vec{e}_y Y(\omega,\tau) + \vec{e}_z Z(\omega,\tau)\right)\right) \forall\ \omega,\tau \qquad (4)$$

where $\vec{e}_x$, $\vec{e}_y$ and $\vec{e}_z$ are unit vectors along the Cartesian axes.

D. Source Location Estimation
Using the calculated DOA vectors, it is possible to perform source localisation using a variety of techniques. Perhaps the simplest is to construct a histogram over an arbitrary time period, and look for peaks. This method, along with another clustering method based on self-learning neural networks, has been looked at to perform this task.

E. Demixing
For each source location, which is denoted E_i, M_i describes a bit mask in the time-frequency domain for each source.

$$M_i(\omega,\tau) = \begin{Bmatrix} 1 & \mid & \arccos\left(\frac{\vec{E}_i}{\left|\vec{E}_i\right|}\cdot\frac{\vec{D}}{\left|\vec{D}\right|}\right) \le \delta \\ 0 & \mid & otherwise \end{Bmatrix} \quad \forall i \tag{5}$$

where δ provides a user defined angular margin around the source location.

The sources can then be recovered by using the mask to filter W in the time-frequency domain.

$$\hat{S}_i = M_i(\omega,\tau) * W(\omega,\tau) \tag{6}$$

from which $\hat{s}_i$ can be gained by performing an inverse time frequency transformation.

F. Results
Table 1 shows the results from a signal separation experiment.

Table 1: Results from a signal separation experiment.

Speaker	Performance Measure				Location	
	Signal-to-Interference Ratio (SIR) in mixture	SIR after masking	SIR gain	Preserved Signal Ratio (PSR) after masking	azimuth	elevation
1	-0.17 dB	12.14 dB	12.32 dB	12.32 dB	120	0
2	-2.96 dB	12.30 dB	15.27 dB	15.27 dB	280	10
3	-6.81 dB	10.89 dB	17.70 dB	17.70 dB	340	20

The separation algorithm was tested on a mixture of three male speakers reading passages from a novel. Each speaker was recorded independently under anechoic conditions, and the mixture created by the summation of the three B-format recordings. The recordings were performed in this manner to allow analytical comparison of the separated speakers with the original recording. Speakers one and two show much higher Preserved Signal Ratio (PSR) results compared to speaker three. This is perhaps unsurprising, considering that speaker three has an initial Signal-to-Interference Ratio (SIR) of −6.81 dB. All the speakers are intelligible on listening, although there is an appreciable level of crackling on speaker three. The SIR gain for all speakers shows excellent results, showing high suppression of the interfering speakers, with an average improvement in SIR of 15 dB. These results compare well to those listed for mixtures of two speakers[10].

As far as the validity of the assumptions, W-disjoint orthogonality has been shown to be a valid assumption for speech signals. Acoustic niche theory also suggests an evolutionary pressure for this to be the case in the animal kingdom. However, the authors concede that in the general case, the assumptions are not guaranteed to hold true. Further investigations into the applicability of these assumptions to a range of situations need to be performed.

4. SIGNAL CLASSIFICATION

ISRIE will also perform the classification of the separated audio signals which are provided by the signal separation as discussed previously. The output of the classification algorithms will advise the user of ISRIE which category of sounds a particular signal belongs to. It is assumed that the input signal to the classification system contains only one sound source.

A. Sound Categories

A taxonomy of sound categories has been devised specifically for the purpose of ISRIE. Figure 2 illustrates these categories.

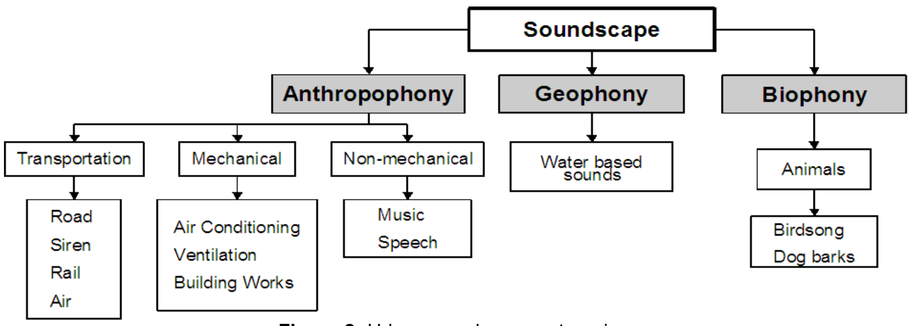


Figure 2: Urban soundscape categories.

Initially, the soundscape is split into three main categories. *Anthropophony* relates to sounds made or caused by human activity, *biophony* sounds are those made by animals, and *geophony* encompasses sounds not caused by either of the above.

B. Classification using Time-Domain Signal Coding

A typical classification system consists of two components: a feature extractor and a classifier[11]. There is sometimes a third component to provide some pre- or post-processing either at the input or output to the system. The data that is to be classified will be passed into the feature extractor whose role it is to reduce the complexity of the data before it reaches the classifier[11] thus optimising the classification process. A good overview of a selection of these techniques can be found in the comparison made by Cowling and Sitte[12].

The feature extraction method that has been used for data reduction in ISRIE is known as Time-Domain Signal Coding (TDSC). This is a purely time-domain analysis method which has previously shown to be successful in the identification of wood-boring insects[13] and in the classification of different Orthoptera[14]. The data produced by the TDSC algorithm describes a waveform by the number of samples (duration - D) and number of minima (shape – S) contained within each epoch (signal between 2 consecutive zero crossings) of the waveform. The D-S information is stored for a given frame of the waveform by means of a codebook. After a signal has been analysed using TDSC, each code within the codebook will have a number of occurrences associated with it to describe its D-S characteristics. It is this frequency information, the S-matrix, which is then used for classification. A more detailed explanation of how TDSC was developed and the other features it can extract from the full bandwidth signal is given by Chesmore[14]. Figure 3 shows how the TDSC analysis fits into the classification system.

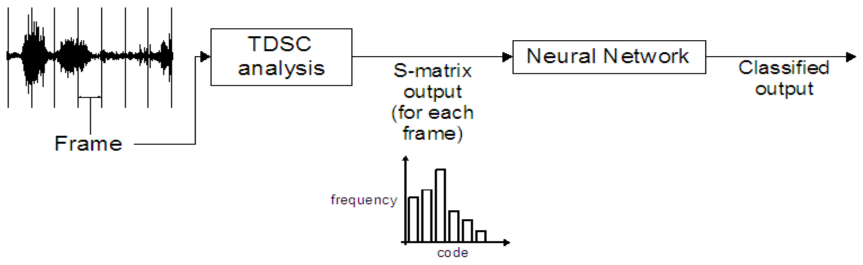


Figure 3: Proposed classification system. The S-matrices for each frame of the waveform are classified individually.

It was decided that a neural network approach in the classification would be adopted. Initially, an unsupervised Self-Organising Map (SOM) network was used but this struggled to differentiate between the test pieces of audio data. Significant improvements in classification were gained by introducing supervised learning into the system. A Learning Vector Quantisation (LVQ) network was implemented using the LVQ1 learning rule[15, 16]. Eight different categories of sounds were placed into 4 groups: group 1 contained air traffic, air conditioning and ventilation units, and building works; group 2 contained road and rail traffic; group 3 contained birdsong and also recordings of crickets; and group 4 contained some speech examples. The grouping of the sounds was chosen based on how consistent the signal was throughout the duration of the recording. After training was completed using a training set of 40 recordings, the network was tested using a 30-second test audio file which combined audio from each of the 4 groups. Network accuracy for each of the individual groups was poor for all but group 1 (88%). However, when combined results were observed for how well the system could recognise non-bioacoustic audio (groups 1 and 2), the accuracy rose to 93%. This shows that it is possible to perform an initial classification using the relatively simple methods discussed above. Work is now focused on developing the system further to incorporate classifiers to differentiate between the various bioacoustic and non-bioacoustic signals. Feed-forward neural networks with backpropagation training are being experimented with and are showing positive initial results.

5. APPLICATIONS

The uses of ISRIE could range from assisting acoustic consultants and planners in making the right decision on the most appropriate control measures in a project where noise concerns may arise, through to assisting soundscape artists and sound engineers with the recording of isolated sound events for either artistic reasons or for the subjective evaluation of different soundscapes. The usefulness of ISRIE in environmental noise impact assessments, such as PPG 24[17], BS 4142[18] and noise nuisance applications have previously been discussed[19]. Over the course of this research project, different stakeholders have also been interviewed in order to assess what measurement parameters would be required from such an instrument to log and what would be the additional benefits from the use of such an instrument.

A. BS 4142
In BS 4142 assessments, ISRIE could potentially be used to obtain the specific noise level L_{Aeq} of a source and the background noise level L_{A90} without requiring the need to measure these descriptors separately. The instrument would offer individual logged values of these two environmental noise level descriptors in order to establish the arithmetic difference between the intruding mechanical noise level and the typical background noise level without the presence of any mechanical plant or industrial noise. Also, in practice, there are instances where it is not possible to obtain separate measurements of these two descriptors, because either the mechanical source cannot be turned off in order to measure the background noise level, or the mechanical noise cannot accurately be quantified at the receptor's location due to interference from other sources, such as transportation related noise. ISRIE would be capable of deriving these parameters through its discrimination and classification algorithms as discussed above.

B. PPG 24
In PPG 24 assessments, the existing environmental noise levels are established over a 24-hour measurement period, when planning a new housing development. The measurements are normally unmanned for economic reasons since they cover such an extensive measurement period. Firstly, it is apparent that in mixed soundscapes, where for example there is almost an equal contribution of railway and road traffic noise, it is difficult to quantify the contributing noise sources, or even determine which is the dominant noise source. Therefore, it is not always feasible to establish the most representative noise source category in which the noise environment should be assessed in. ISRIE would be useful in obtaining these individual contributions in L_{Aeq} terms in order to decide which is the prominent noise source in that specific environment. Secondly, ISRIE would automatically log and classify individual events that exceed a certain criterion, such as 82 dB $L_{A,max,S}$ and assess whether these transient events are intrusive sources of noise, e.g. mechanical, or non-intrusive, e.g. birdsong or sounds from other animal life. This type of automated assessment is not possible with the use of current technology since the noise survey is normally unmanned and these individual transient events can only be evaluated and assessed at the post-processing stage.

C. Noise Nuisance
Environmental Health Officers (EHOs) of Local Authorities in the UK would make use of an advanced sound instrument for various reasons. Firstly, ISRIE would enable them to investigate complex noise complaints in the case where it is not clear which mechanical plant noise source affects the complainant's house in a highly built-up area. Secondly, the problem of low frequency noise, potentially originating from tunneling or drilling works, can be an issue for some residents in a community. These noise complaints can be difficult to assess with the current technology of sound level meters and ISRIE's characterisation capability would work well in these types of problem where the source is of tonal character. Thirdly, ISRIE would aid in monitoring the noise from music events and assist EHOs in reaching decisions upon the licensing of commercial premises that may give rise to noise complaints.

D. Other Engineering Consultancy Problems
The use of a conventional sound level might not be adequate in some cases since there can be interference from other noisy equipment when trying to quantify a particular noise source in an industrial area. There are also instances, where the noise of certain installations, such as

electrical transformers, cannot easily be quantified because either these installations are near sources of transportation noise, e.g. motorways, or because there are other electro/mechanical installations nearby that may contribute to the overall measured level. Also, as part of the Land Compensation Act, difficulties can arise when trying to establish only the road traffic components at houses that are situated miles away from a newly constructed or modified road. ISRIE would be capable of solely measuring the traffic noise components from the remaining background noise, something that is not possible with the current sound level meters. Similar measurement problems can arise when trying to quantify noise solely emanating from racing tracks that might affect nearby communities.

E. Soundscape Recordings

Recordings of soundscapes is developing in many applications ranging from creating archived sound recordings of a variety of animal sounds through to the recordings of any other types of soundscape for recreating experiences in art installations, museums and galleries. The need for carrying out recordings of sounds in isolation is important in many applications. At the moment, in order to separate different sounds, noise suppression techniques are used in order to filter out the remaining sound, or the recording is delayed until the level of the intrusive noise has dropped to such a level that it is not significantly contributing to the overall level. ISRIE would be useful in recording these sounds as isolated events and hence providing a reference instrument for sound recording.

F. Future Policy

ISRIE could enable planners to consider the balance between ‘positive’, e.g. natural sounds and ‘negative’ sounds, e.g. mechanical-like sounds in a mixed sound environment as part of a regeneration plan for improving the quality of life in urban agglomerations or assist in the design of new spaces of personal enjoyment and recreation in metropolitan cities. The first step would be to establish which types of sound are considered ‘wanted’ and ‘unwanted’ in that environment. Then, ISRIE would be used as an instrument to establish the current percentage of wanted and unwanted sounds through its source discrimination and classification algorithms as presented above. Finally, the management of these sounds would involve standard noise abatement techniques along with the potential introduction of more wanted sounds. In the end, ISRIE could be used to assess whether the desired ‘mix’ of wanted and unwanted sounds was achieved.

5. CONCLUSIONS

The need of a network sensor system with the development of algorithms and techniques for automatically characterising sounds in a complex sound environment is more evident than ever before. This paper has presented a number of suggested measurement platforms for the measurement of sounds along with promising techniques for signal separation and classification. The use of ISRIE could ultimately revolutionalise the way we currently perceive soundscapes and could affect the way we measure, assess and record sounds in the future.

ACKNOWLEDGMENTS

University of Southampton, York and Newcastle gratefully acknowledge the Engineering and Physical Sciences Research Council (EPSRC) for sponsoring this research work.

REFERENCES

1. H. Atmoko, T. Gui Yun and B. Fazenda, "Accurate sound source localization in a reverberant environment using multiple acoustic sensors", Meas. Sci. Technol. Feb. 2008, pp. 1-10.
2. Terence Betlehem, "Acoustic Signal Processing Algorithms for Reverberant Environments", PhD Thesis, Department of Information Engineering, School of Information Sciences and Engineering, Australian National University, Nov. 2005.
3. Yunhong Li, "Broadband Beamforming and Direction Finding Using Concentric Ring Array", PhD Thesis, the Faculty of the Graduate School, University of Missouri-Columbia, Jul. 2005.
4. Acoustic sound source localisation: Download: Acoustic Camera: Applications and System Overview (PDF), Available at: http://www.acoustic-camera.com/pdfs/ac_brochure2009.pdf, Accessed: May 2009.
5. Michael Gerzon. Periphony: With-height sound reproduction. Journal Audio Eng. Soc., 21(1), pp2–10, 1973.
6. Michael Gerzon. The design of precisely coincident microphone arrays for stereo and surround sound. In Proc. 50th Convention of the Audio Eng. Soc., 1975.
7. S. Rickard and Z. Yilmaz. On the approximate w-disjoint orthogonality of speech. In Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP '02), 1, pp 529–532, 2002.
8. J. Merinaa and V. Pulkki. Spatial impulse response rendering. In Proc. of the 7th Int. Conf. on Digital Audio Effects (DAFx'04), pp 139–144, October 2004.
9. Ville Pulkki. Spatial sound reproduction with directional audio coding (DirAC). Journal Audio Eng. Soc., 55(6), June 2007.
10. O. Yilmaz and S. Rickard. Blind separation of speech mixtures via time-frequency masking. IEEE Journal. Sig. Proc., 52(7), pp1830–1847, 2004.
11. R. Beale and T.O. Jackson, Neural Computing: An Introduction, Hilger 1998.
12. M. Cowling and R. Sitte, "Comparison of techniques for environmental sound recognition", *Pattern Recognition Letters* 24, pp. 2895-2907 (2003).
13. I. Farr and E. D. Chesmore, "Automated bioacoustic detection and identification of wood-boring insects for quarantine screening and insect ecology", in *Proceedings of the Institute of Acoustics* 29, Pt. 3, pp. 201-208 (2007).
14. E.D. Chesmore, "Application of time domain signal coding and artificial neural networks to passive acoustical identification of animals", *Applied Acoustics* 62, pp. 1359-1374 (2001).
15. T. Kohonen, "Improved Versions of Learning Vector Quantization", *International Joint Conference on Neural Networks* 1, pp. 545-550 (1990).
16. H. Demuth and M. Beale, Neural Network Toolbox User's Guide, The MathWorks, Inc. 2001.
17. Planning Policy Guidance 24: Planning and noise, Department of the Environment, 1994.
18. BS 4142: 1997: Method for rating industrial noise affecting mixed residential and industrial areas, BSI.
19. C. Karatsovis and S J C Dyne, "Instrument for soundscape recognition, identification and evaluation: an overview and potential use in legislative applications", in *Proceedings of the Institute of Acoustics,* 2008, Vol. 30, Pt.2.

References

Abdi, H. A neural network primer. *Jounal of Biological Systems*, 2(3): pages 247–283 (1994).

Addison, P.S. *The Illustrated Wavelet Transform Handbook*. Institute of Physics Publishing (2002).

Allegro, S., Büchler, M., and Launer, S. Automatic sound classification inspired by auditory scene analysis. In *Eurospeech* (2001).

Atmoko, H., Tian, G.Y., and Fazenda, B. Advanced instrumentation for sound monitoring. In *Proceedings of the 2007 IEEE International Conference on Networking, Sensing and Control*, pages 449–454 (2007).

Atmoko, H., Tan, D.C., Tian, G.Y., and Fazenda, B. Accurate sound source localization in a reverberant environment using multiple acoustic sensors. *Measurement Science and Technology*, 19: pages 1–10 (2008).

Bardeli, R., Wolff, D., Kurth, F., Koch, M., Tauchert, K.H., and Frommolt, K.H. Detecting bird sounds in a complex acoustic environment and application to bioacoustic monitoring. *Pattern Recognition Letters*, 31(12): pages 1524–1534 (2010).

Beale, R. and Jackson, T.O. *Neural Computing: An Introduction*. Hilger (1998).

Beltrán, N., Duarte-Mermoud, M., Bustos, M., Salah, S., Loyola, E., Peña-Neira, A., and Jalocha, J. Feature extraction and classification of chilean wines. *Journal of Food Engineering*, 75: pages 1–10 (2006).

Bond, F. and Cahn, C. On the sampling the zeros of bandwidth limited signals. *Information Theory, IRE Transactions on*, 4(3): pages 110–113 (1958).

Botteldooren, D., Coensal, B.D., and Muer, T.D. The temporal structure of urban soundscapes. *Journal of Sound and Vibration*, 292: pages 105–123 (2006).

British Standards Institute. BS 4142: Method for rating industrial noise affecting mixed residential and industrial areas (1997).

Bunting, O. *Sparse Separation of Sources in 3D Soundscapes*. Ph.D. thesis, University of York (2011).

Bunting, O., Stammers, J., Chesmore, D., Bouzid, O., Tian, G.Y., Karatsovis, C., and Dyne, S. Instrument for soundscape recognition, identification and evaluation (ISRIE): technology and practical uses. In *Proceedings of Euronoise 2009* (2009).

Burson, S. Natural soundscape monitoring in yellowstone national park december 2005-march 2006 (2006). Grand Teton National Park Soundscape Program Report No. 200601.

Cain, R., Jennings, P., and Poxon, J. The development and application of the emotional dimensions of a soundscape. *Applied Acoustics*, In Press (2011).

Carles, J.L., Barrio, I.L., and de Lucio, J.V. Sound influence on landscape values. *Landscape and Urban Planning*, 43: pages 191–200 (1999).

Chedad, A., Moshou, D., Aerts, J.M., Hirtum, A.V., Ramon, H., and Berckmans, D. Recognition system for pig cough based on probabilistic neural networks. *Journal of Agricultural Engineering Research*, 79: pages 449–457 (2001).

Chesmore, E.D. Application of time domain signal coding and artificial neural networks to passive acoustical identification of animals. *Applied Acoustics*, 62: pages 1359–1374 (2001).

Chesmore, E.D. Classification of *Tibicen* cicada (2004). Unpublished data.

Chesmore, E.D. and Ohya, E. Automated identification of field-recorded songs of four british grasshoppers using bioacoustic signal recognition. *Bulletin of Entomological Research*, 94: pages 319–330 (2004).

Cohen, L. *Time-Frequency Analysis*. Prentice-Hall PTR (1995).

Council Directive (EC). 2002/49/EC of 25 june 2002 on the assessment and management of environmental noise (2002).

Couvreur, C., Fontaine, V., Gaunard, P., and Mubikangiey, C.G. Automatic classification of environmental noise events by hidden markov models. *Applied Acoustics*, 54(3): pages 187–206 (1998).

Cowling, M. and Sitte, R. Comparison of techniques for environmental sound recognition. *Pattern Recognition Letters*, 24: pages 2895–2907 (2003).

Defréville, B., Roy, P., Rosin, C., and Pachet, F. Automatic recognition of urban sound sources. In *Audio Engineering Society 120th Convention* (2006).

Demuth, H., Beale, M., and Hagan, M. *Neural Network Toolbox User's Guide*. The MathWorks, Inc. (2008).

Department for Communities and Local Government. Planning policy guidance 24: Planning and noise. Last accessed online 28 August 2011: http://www.communities.gov.uk/planningandbuilding (1994).

Dieckmann, U., Plankensteiner, P., and Wagner, T. Sesam: A biometric person identification system using sensor fusion. *Pattern Recognition Letters*, 18(9): pages 827–833 (1997).

Dietrich, C., Palm, G., Riede, K., and Schwenker, F. Classification of bioacoustic time series based on the combination of global and local decisions. *Pattern Recognition*, 37(12): pages 2293–2305 (2004).

Dimoulas, C., Kalliris, G., Papanikolaou, G., Petridis, V., and Kalampakas, A. Bowel-sound pattern analysis using wavelets and neural networks with application to long-term, unsupervised, gastrointestinal motility monitoring. *Expert Systems with Applications*, 34: pages 26–41 (2008).

Dufournet, D. and Jouenne, P. Madras, an intelligent assistant for noise recognition. In *Internoise* (1997).

Dufournet, D., Jouenne, P., and Rozwadowski, A. Automatic noise source recognition. *Journal of the Acoustical Society of America*, 103(5): page 2950 (1998).

Evans, N. *Automated Vehicle Detection and Classification using Acoustic and Seismic Signals*. Ph.D. thesis, University of York (2010).

Farr, I.J. *Automated Bioacoustic Identification of Statutory Quarantined Insect Pests*. Thesis, University of York (2007).

Gage, S.H., Maher, R., and Sanchez, G. Ecoears: Ecological & environmental acoustic remote sensor — application for long-term monitoring and assessment of wildlife. In *Technnical Symposium & Workshop: Threatened, Endangered and At-Risk Species on DoD and Adjacent Lands* (2005).

Ge, J. and Hokao, K. Applying methods of image evaluation and spatial analysis to study the sound environment of urban street areas. *Journal of Environmental Psychology*, 25: pages 455–466 (2005).

Ghiurcau, M.V., Rusu, C., Bilcu, R.C., and Astola, J. Audio based solutions for detecting intruders in wild areas. *Signal Processing*, 92(3): pages 829–840 (2012).

Gold, B. and Morgan, N. *Speech and Audio Signal Processing: Processing and Perception of Speech and Music.* Wiley (2000).

Güler, I. and Übeyli, E.D. Implementing wavelet/probabilistic neural networks for doppler ultrasound blood flow signals. *Expert Systems with Applications*, 31: pages 130–136 (2006).

Hagan, M., Demuth, H., and Beale, M. *Neural Network Design.* PWS (1995).

Haitsma, J. and Kalker, T. A highly robust audio fingerprinting system. In *Proceedings of the International Symposium on Music Information Retrieval.* Paris, France (2002).

Haitsma, J., Kalker, T., and Oostveen, J. Robust audio hashing for content identification. In *Content Based Mulimedia Indexing.* Brescia, Italy (2001).

Hall, D.A., Irwin, A., Edmondson-Jones, M., Phillips, S., and Poxon, J.E. An exploratory evaluation of perceptual, psychoacoustic and acoustical properties of urban soundscapes. *Applied Acoustics*, In Press (2011).

Ham, F.M. and Kostanic, I. *Principles of Neurocomputing for Science & Engineering.* McGraw-Hill (2001).

Hansen, L. and Salamon, P. Neural network ensembles. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 12(10): pages 993–1001 (1990).

Hawickhorst, B., Zahorian, S., and Rajagopal, R. A comparison of three neural network architectures for automatic speech recognition. In *Proceedings of the Artificial Neural Networks in Engineering Conference*, volume 5, pages 221–226. Intelligent Engineering Systems through Artificial Neural Networks USA (1995).

He, Q., Feng, Z., and Kong, F. Detection of signal transients using independant component analysis and its application in gearbox condition monitoring. *Mechanical Systems and Signal Processing*, 25(5): pages 2056–2071 (2007).

Hertz, J., Krogh, A., and Palmer, R.G. *Introduction to the Theory of Neural Computation*. Perseus Books (1991).

Hubbard, B.B. *The World According to Wavelets*. A K Peters, 2nd edition (1998).

Jennings, P. and Cain, R. A framework for improving urban soundscapes. *Applied Acoustics*, In Press (2012).

Kang, M.A. and Servign, S. Animated cartography for urban soundscape information. In *GIS '99: Proceedings of the 7th ACM international symposium on Advances in geographic information systems*, pages 116–121. ACM, New York, NY (1999).

Karatsovis, C. and Dyne, S. Instrument for soundscape recognition, identification and evaluation: an overview and potential use in legislative applications. In *Proceedings of the Institute of Acoustics Spring Conference: Widening Horizons in Acoustics*. Reading, UK (2008).

Kim, H.G., Burred, J.J., and Sikora, T. How efficient is mpeg-7 for general sound recognition? In *AES 25th International Conference* (2004).

King, R. and Gosling, W. Time-encoded speech. *Electronics Letters*, 14(15): pages 456–457 (1978).

King, R. and Phipps, T. Shannon, tespar and approximation strategies. *Computing & Security*, 18(5): pages 445–453 (1999).

Kohonen, T. Improved versions of learning vector quantization. In *International Joint Conference on Neural Networks*, volume 1, pages 545–550 (1990).

Krijnders, J., Niessen, M., and Andringa, T. Sound event recognition through expectancy-based evaluation ofsignal-driven hypotheses. *Pattern Recognition Letters*, 31(12): pages 1552–1559 (2010).

Kurt, I., Ture, M., and Kurum, A.T. Comparing performances of logistic regression, classification and regression tree, and neural networks for predicting coronary artery disease. *Expert Systems with Applications*, 34: pages 366–374 (2008).

Lee, C.H., Chou, C.H., Han, C.C., and Huang, R.Z. Automatic recognition of animal vocalizations using averaged mfcc and linear discriminant analysis. *Pattern Recognition Letters*, 27(2): pages 93–101 (2006).

Linder, R., Albers, A.E., Hess, M., Pppl, S.J., and Schnweiler, R. Artificial neural network-based classification to screen for dysphonia using psychoacoustic scaling of acoustic voice features. *Journal of Voice*, 22(2): pages 155–163 (2008).

López-Cózar, R. and Callejas, Z. Two-level speech recognition to enhance the performance of spoken dialogue systems. *Knowledge-Based Systems*, 19: pages 153–163 (2003).

Ma, L., Milner, B., and Smith, D. Acoustic environment classification. *ACM Transactions on Speech and Language Processing*, 3(2): pages 1–22 (2006).

Mallat, S. *A Wavelet Tour of Signal Processing*. Academic Press, 2nd edition (1999).

Marchant, B. Time-frequency analysis for biosystems engineering. *Biosystems Engineering*, 85: pages 261–281 (2003).

Mazarakis, G.P. and Avaritsiotis, J.N. Vehicle classification in sensor networks using time-domain signal processing and neural networks. *Microprocessors and Microsystems*, 31: pages 381–392 (2007).

Moca, V.V., Scheller, B., Mureşan, R.C., Daunderer, M., and Pipa, G. Eeg under anesthesia–feature extraction with tespar. *Computer Methods and Programs in Biomedicine*, 95(3): pages 191–202 (2009).

Nooralahiyan, A.Y., Kirby, H.R., and McKeown, D. Vehicle classification by acoustic signature. *Mathematical and Computer Modelling*, 27: pages 205–214 (1998).

Norris, M.J. and Denham, S.L. Sound texture detection using self-organizing maps. Center for Theoretical and Computational Neuroscience, University of Plymouth, UK (2003).

Ohya, E. Identification of tibicen cicada species by a principal components analysis of their songs. *Anais da Academia Brasileira de Ciêcias*, 76(2): pages 441–444 (2004).

Peltonen, V., Tuomi, J., Klapuri, A., Huopaniemi, J., and Sorsa, T. Computational auditory scene recognition. In *Proceedings of the International Conference on Acoustic, Speech and Signal Processing (ICASSP)*. Orlando, FL (2002).

Rabiner, L.R. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2): pages 257–286 (1989).

Rafiee, J., Rafiee, M., and Tse, P. Application of mother wavelet functions for automatic gear and bearing fault diagnosis. *Expert Systems with Applications*, 37(6): pages 4568–4579 (2010).

Rafiq, M.Y., Bugmann, G., and Easterbrook, D.J. Neural network design for engineering applications. *Computers & Structures*, 79(17): pages 1541–1552 (2001).

Raimbault, M., Lavandier, C., and Bérengier, M. Ambient sound assessment of urban environments: field studies in two french cities. *Applied Acoustics*, 64: pages 1241–1256 (2003).

Ray, S.R. and Hsu, W.H. Self-organized-expert modular network for classification of spatiotemporal sequences. *Intelligent Data Analysis*, 2: pages 287–301 (1998).

Ruvolo, P., Fasel, I., and Movellan, J.R. A learning approach to hierarchical feature selection and aggregation for audio classification. *Pattern Recognition Letters*, 31(12): pages 1535–1542 (2010).

Schafer, R.M. *The Tuning of the World*. Knopf, New York (1977).

Schafer, R.M. (editor). *The Vancouver Soundscape*. A.R.C. Publications (1978).

Schclar, A., Averbuch, A., Rabin, N., Zheludev, V., and Hochman, K. A diffusion framework for detection of moving vehicles. *Digital Signal Processing*, 20(1): pages 111–122 (2010).

Selin, A., Turunen, J., and Tanttu, J.T. Bird sound classification and recognition using wavelets. *Dissertationes Classis 4: Historia Naturalis*, 47: pages 185–204 (2006).

Stammers, J. Signal analysis and classification for ISRIE. Presentation at Applied Soundscapes Symposium, University of Salford, 17 September (2009).

Takaguchi, T. and Nishimura, M. Us 7660717 speech recognition system and software thereof. United States Patent (2010).

The World Soundscape Project Website. Last accessed online 3 October 2011: http://www.sfu.ca/∼truax/wsp.html (2007).

Tong, F., Tso, S., and Xu, X. Tile-wall bonding integrity inspection based on time-domain features of impact acoustics. *Sensors and Actuators A: Physical*, 132(2): pages 557–566 (2006).

Umapathy, K., Krishnan, S., and Rao, R. Audio signal feature extraction and classification using local discriminant bases. *Audio, Speech, and Language Processing, IEEE Transactions on*, 15(4): pages 1236–1246 (2007).

Van Hirtum, A. and Berckmans, D. Automated recognition of spontaneous versus voluntary cough. *Medical Engineering & Physics*, 24(7-8): pages 541–545 (2002).

Verfaille, V., Guastavino, C., and Traube, C. An interdisciplinary approach to audio effect classification. In *Proceedings of the 9th International Conference on Digital Audio Effects (DAFx-06)*, pages 107–113 (2006).

Wang, Y., Lee, C.M., Kim, D.G., and Xu, Y. Sound-quality prediction for non-stationary vehicle interior noise based on wavelet pre-processing neural network model. *Journal of Sound and Vibration*, 299(4–5): pages 933–947 (2007).

Wu, J.D. and Liu, C.H. Investigation of engine fault diagnosis using discrete wavelet transform and neural network. *Expert Systems with Applications*, 35(3): pages 1200–1213 (2008).

Yang, W. and Kang, J. Acoustic comfort evaluation in urban open public spaces. *Applied Acoustics*, 66: pages 211–229 (2005).

Zils, A. and Pachet, F. Automatic extraction of music descriptors from acoustic signals using eds. In *Audio Engineering Society 116th Convention* (2004).