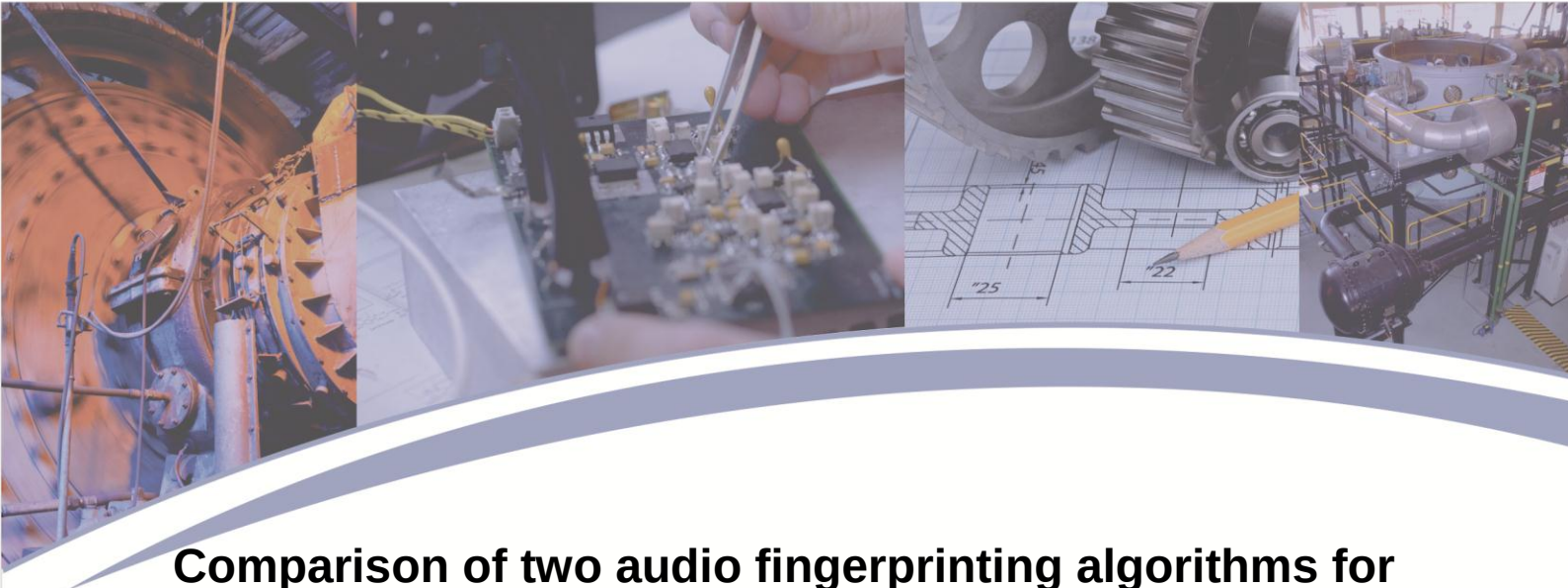




Comparison of two audio fingerprinting algorithms for advertisement identification

by:

Mr H.A van Nieuwenhuizen

A dissertation submitted for the partial fulfilment of the requirements for the degree

MASTER OF ENGINEERING

in

COMPUTER AND ELECTRONIC ENGINEERING

North-West-University Potchefstroom Campus

Supervisor: Prof. W.C Venter

Potchefstroom

2011

Declaration

I, Heinrich Abrie van Nieuwenhuizen, hereby declare that the dissertation entitled “Comparison of two audio fingerprinting algorithms for advertisement identification” is my own original work and has not already been submitted to any other university or institution for examination.

H.A van Nieuwenhuizen
Student number: 20252188
Signed on 18th day of November 2011 at Potchefstroom

Proofreading

16 November 2011

Ek, Elma de Kock, het die M-verhandeling van H.A van Nieuwenhuizen (20252188), Comparison of two audio fingerprinting algorithms for advertisement identification, se teksversorging hanteer.

Ek beskik oor 'n Honneursgraad in Afrikaans en Nederlands met taalpraktyk as vak.

Dankie

I, Elma de Kock, proofread the M-dissertation of H.A van Nieuwenhuizen (20252188), Comparison of two audio fingerprinting algorithms for advertisement identification.

I have an Honours in Afrikaans and Dutch with language practice as a subject.

Thank you

Elma de Kock

Sel: 083 302 5282

e-pos: elma.dekock@gmail.com

Acknowledgments

I would like to acknowledge everyone who made this dissertation possible:

Prof. Willem Venter who gave 24 hour guidance. Supported me in the long 2 years of completion and keeping the subject interesting.

Special thank you to Sune von Solms for your support and immense effort on this dissertation, without her this dissertation would not have been as successful as it is. Ludwig de Bruyn and Trevor Nel for their efforts and support as well, the input is very much appreciated.

Melvin Ferreira for the life lessons, whose inputs made me the engineer I am today. My friends Samuel van Loggrenberg, Jean du Toit, Sammy Rabie and Henno Esterhuyse for the inputs and great times that kept me sane.

The TeleNet research group, for the support structure and motivation, especially Leenta Grobler for your personal guidance.

My parents Hendrik and Magda, whose love and financial support made everything possible. My twin brother Reinach for calming me down and keeping me sane.

Leoni Boshoff, the love of my life and my fiancé. Thank you for your love and support, through all the late nights and long hours.

Abstract

Although the identification of humans by fingerprints is a well-known technique in practice, the identification of an audio sample by means of a technique called audio fingerprinting is still under development. Audio fingerprinting can be used to identify different types of audio samples of which music and advertisements are the two most frequently encountered. Different audio fingerprinting techniques to identify audio samples appear seldom in the literature and direct comparisons of the techniques are not always available

In this dissertation, the two audio fingerprinting techniques of Avery Wang and Haitsma and Kalker are compared in terms of accuracy, speed, versatility and scalability, with the goal of modifying the algorithms for optimal advertisement identification applications. To start the background of audio fingerprinting is summarised and different algorithms for audio fingerprinting are reviewed. Problems, issues to be addressed and research methodology are discussed. The research question is formulated as follows – **“Can audio fingerprinting be applied successfully to advertisement monitoring, and if so, which existing audio fingerprinting algorithm is most suitable as a basis for a generic algorithm and how should the original algorithm be changed for this purpose?”**

The research question is followed by literature regarding the background of audio fingerprinting and different audio fingerprinting algorithms. Next, the importance of audio fingerprinting in the engineering field is motivated by the technical aspects related to audio fingerprinting. The technical aspects are not always necessary or part of the algorithm, but in most cases, the algorithms are pre-processed, filtered and downsampled. Other aspects include identifying unique features and storing them, on which each algorithm's techniques differ.

More detail on Haitsma and Kalker's, Avery Wang's and Microsoft's RARE algorithms are then presented.

Next, the desired interface for advertisement identification Graphical User Interface (GUI) is presented. Different solution architectures for advertisement identification are discussed. A design is presented and implemented which focuses on advertisement identification and helps with the validation process of the algorithm.

The implementation is followed by the experimental setup and tests. Finally, the dissertation ends with results and comparisons, which verified and validated the algorithm and thus affirmed the first part of the research question. A short summary of the contribution made in the dissertation is given, followed by conclusions and recommendations for future work.

Keywords: Audio Fingerprinting; Automatic Music Recognition; Content-based Audio Identification; Perceptual Hashing; Robust Matching

CONTENTS

LIST OF FIGURES	VIII
LIST OF TABLES	IX
LIST OF ACRONYMS.....	X
LIST OF VARIABLES	10
CHAPTER 1 – INTRODUCTION.....	11
1.1 BACKGROUND.....	11
1.2 RESEARCH QUESTION	13
1.3 OBJECTIVES	13
1.4 RESEARCH METHODOLOGY	14
1.4.1 Dissertation plan of action	14
1.4.2 Clarify technical aspects relating to audio fingerprinting	15
1.4.3 Study different audio fingerprinting algorithms	15
1.4.4 Selection of audio fingerprinting algorithm	15
1.4.5 Selection of Software environment	16
1.4.6 Experimental setup	16
1.4.7 Implementation of the generic algorithm	16
1.4.8 Acquire results.....	16
1.4.9 Reach a verdict.....	16
1.5 ORGANISATION OF DISSERTATION	17
CHAPTER 2 – AUDIO FINGERPRINTING	18
2.1 THE ORIGINAL PROBLEM / IDEA	18
2.2 ALGORITHMS	19
2.3 ADVERTISING VS. MUSIC.....	20
2.3.1 Sound.....	20
2.3.2 Speech and music	21
2.4 WHY AUDIO FINGERPRINTING IS RELEVANT IN THE ENGINEERING FIELD	21
CHAPTER 3 – TECHNICAL ASPECTS RELATED TO AUDIO FINGERPRINTING	22
3.1 FREQUENCIES.....	22
3.1.1 Pre-processing.....	22
3.1.2 Filters	23
3.1.3 Applying the filter coefficients [19]	26
3.1.4 Downsampling.....	28
3.2 UNIQUE FEATURES	30
3.2.1 Fingerprints.....	30
3.2.2 Pattern recognition.....	31
3.2.3 Unique features in the downsampled signal.....	32
3.2.4 Database.....	33
CHAPTER 4 – AUDIO FINGERPRINTING	34
4.1 GROUP 1: HAITSMA AND KALKER’S ALGORITHM [2]	35
4.2 GROUP 2: AVERY WANG’S SHAZAM ALGORITHM [1]	37
4.3 GROUP 3: MICROSOFT’S ROBUST AUDIO RECOGNITION ENGINE (RARE) [7].....	38
CHAPTER 5 – INTERFACE DESIGN AND SOLUTION ARCHITECTURE.....	40
5.1 DATABASE SETUP AND COMPARISON	41
5.2 ENVIRONMENTS	42
5.2.1 Matlab	42
5.2.2 VB.NET.....	43
5.3 EXPERIMENTAL SETUP	43
5.3.1 Data types.....	43
5.3.2 Tests.....	43
CHAPTER 6 – IMPLEMENTATION	45
6.1 FUNCTIONALITY 1: PRE-PROCESSING AUDIO	46

6.2	FUNCTIONALITY 2: LANDMARK DISCOVERY	47
6.2.1	<i>Parameters</i>	48
6.2.2	<i>Spectrogram</i>	54
6.2.3	<i>Local max and spreading</i>	55
6.3	FUNCTIONALITY 3: LANDMARKS TO HASH	58
6.4	FUNCTIONALITY 4: SAVE HASH	60
6.5	FUNCTIONALITY 5: COMPARE LANDMARKS	61
6.6	GUI BUTTONS	62
6.6.1	<i>Button 1: Add to RAM wav or mp3</i>	62
6.6.2	<i>Button 2: Add single audio files</i>	62
6.6.3	<i>Button 3: Identify unlabelled segment</i>	63
6.6.4	<i>Button 4: Save to Textfile</i>	63
CHAPTER 7 – RESULTS		64
7.1	VERIFICATION AND VALIDATION	64
7.1.1	<i>Verification</i>	64
7.1.2	<i>Validation</i>	65
7.2	ADVERTISEMENT IDENTIFICATION	68
CHAPTER 8 – CONCLUSIONS AND RECOMMENDATIONS		70
8.1	CONCLUSION.....	70
8.1.1	<i>Achieved objectives</i>	70
8.1.2	<i>Summary of contribution</i>	71
8.1.3	<i>Interpreting results</i>	71
8.1.4	<i>To the point</i>	72
8.2	RECOMMENDATIONS	72
8.2.1	<i>Advertisement identification</i>	72
8.2.2	<i>Video identification</i>	73
8.2.3	<i>Speaker identification</i>	73
ADDENDUM		74
BIBLIOGRAPHY		79
APPENDIX A – CONFERENCE CONTRIBUTIONS FROM THIS DISSERTATION		82

List of Figures

Figure 1 : Frequency analysis of an original wav [3]	12
Figure 2 : Frequency analysis of a mp3-sourced file [3]	12
Figure 3 : Dissertation plan	14
Figure 4 : Typical Frequency Response [15].....	20
Figure 5 : Stereo 44.1 kHz conversion to Mono 8 820 Hz	23
Figure 6 : Filter impulse and frequency response.....	26
Figure 7 : Upsampling, filtering and downsampling signal to 8 820 Hz	30
Figure 8 : Fingerprint.....	30
Figure 9 : Fingerprint pattern recognition	31
Figure 10 : Example of bit representation [30].....	32
Figure 11 : Example of spectrogram peaks [31].....	32
Figure 12 : Content-based audio identification framework [32]	34
Figure 13 : Haitsma and Kalker's algorithm [2].....	36
Figure 14 : Haitsma and Kalker's audio fingerprint block.....	36
Figure 15 : Hash details [1]	37
Figure 16 : Microsoft fingerprint extraction scheme	39
Figure 17 : Oriented PCA [7]	39
Figure 18 : Advertisement identification functionality.....	40
Figure 19 : Implementation	45
Figure 20 : Pre-processing Audio	46
Figure 21 : Finding Landmarks	47
Figure 22 : Number of landmarks vs. frequency standard deviation.....	49
Figure 23 : Number of landmarks vs. decay rate.....	49
Figure 24 : Number of landmarks vs. maximum number of peaks per frame	50
Figure 25 : Number of landmarks vs. target distance frequency	50
Figure 26 : Number of landmarks vs. target distance time.....	51
Figure 27 : Target region	51
Figure 28 : Spectrogram with 50% overlap	52
Figure 29 : Number of landmarks vs. maximum peaks per second.....	52
Figure 30 : Number of landmarks vs. maximum pairs per peak.....	53
Figure 31 : Spectrogram architecture	54
Figure 32 : Local max.....	55
Figure 33 : Gaussian coefficients	56
Figure 34 : Local maximum spread of one local maximum	56
Figure 35 : Creating the threshold	57
Figure 36 : Hash calculation	58
Figure 37 : Hash calculation example	59
Figure 38 : Database structure	60
Figure 39 : Discover landmarks for unknown label	61
Figure 40 : Compare hashes	61
Figure 41 : Generic vs. Phillips	67
Figure 42 : Average of Gaussian white noise addition.....	68
Figure 43 : Recognition rate -- Additive Noise [1].....	75

List of Tables

Table 1 : The number of real multiplications necessary for the convolution of two n point sequences [19]	27
Table 2 : DFT cost effective	28
Table 3 : Equaliser	44
Table 4 : Parameters	48
Table 5 : Representation of database	60
Table 6 : buttons	62
Table 7 : Error rating for different kinds of signal degradations	66
Table 8 : Gaussian white noise addition	67
Table 9 : Advertisements results.....	69
Table 10 : Kaiser window coefficients	74
Table 11 : BER for different kinds of signal degradations [2]	75
Table 12 : Advertisements identified.....	76

List of Acronyms

API	Application programming interface
BER	Bit error rate
COLA	Constant overlap-add
CD	Compact Disk
dB	Decibels
DC	Direct current
DDA	Distortion discriminant analysis
DFT	Discrete Fourier Transform
DSP	Digital Signal Processing
FFT	Fast Fourier Transform
FIR	Finite impulse response
HAS	Humans Auditory System
Hz	Hertz
IDFT	inverse discrete Fourier transform
Mp3	MPEG-2 Audio Layer III
OFM	Radio Oranje
PCA	Principal component analysis
P2P	Peer-to-peer
RAM	Random access memory
RARE	Microsoft's Robust Audio Recognition Engine
SI	International System of Units
THRIP	Technology and Human Resources for Industry Programme
VB.NET	Visual Basic .NET Framework
Wav	Waveform
GUI	Graphical User Interface

List of Variables

A_dec	Decay rate
F_sd	Frequency standard deviation
Fft_hop	FFT hop is the distance before the next frame start
Fft_ms	FFT frame size in time domain
Maxespersec	Maximum prominent frequency peaks allowed per second
Maxpairsperpeak	A maximum of pairs per anchor frequency peak
Maxpkperframe	Maximum prominent frequency peaks per frame
Targetdf	Target distance frequency
Targetdt	Target distance time

Chapter 1 – Introduction

This chapter serves as an introduction to this dissertation. A background of audio fingerprinting is provided and different algorithms for audio fingerprinting are reviewed. Problems, issues to be addressed and the research methodology are discussed in this chapter, followed by the deliverables, benefits and the plan of action.

1.1 Background

The identification of humans through fingerprints is a well-known technique in practice, but the identification of an audio sample by means of a technique called audio fingerprinting is still under development. Audio fingerprinting can be used to identify different types of audio samples of which music and advertisements are the two most often encountered. Different audio fingerprinting techniques to identify audio samples appear in the literature from time to time, but direct comparisons of the techniques are not always available. In this dissertation, the two audio fingerprinting techniques of Avery Wang [1] and Haitsma and Kalker's [2] are compared with regard to advertisement identification in terms of accuracy, speed, versatility and scalability.

Fingerprinting systems are not a new concept as they have been around for more than a hundred years. In 1893, Sir Francis Galton was the first to prove that no two fingerprints of human beings are alike [2]. This notion was then further developed by using any unique feature to identify an object, including the iris and even ears. Soon people also realised the potential of constructing fingerprints of audio signals to identify and compare them. This principle is called audio fingerprinting.

When one hears a song on the radio or from a compact disk (CD), it might sound similar to each other, but the truth is that they are not the same mathematically – especially with added noise or adjustments made to the audio signal [2]. When introduced to this subject, a frequent question arises: “Why should audio fingerprinting be used to identify an audio signal?” This is generally followed by the question of whether an easier technique, such as cross-correlation, would be sufficient. The answers to these questions are not quite that simple, since if one would use any ordinary mathematical comparison equations on a normalised and an identical Waveform (wav) Audio File, it would probably work, but the solution for a reliable, robust and accurate result that is extracting unique features, is possible through the use of audio fingerprinting.

For example, if one performs a frequency analysis on the wav file – *Prove It All Night* from *Winterland Night* [3] in **Figure 1**, one will observe that there is a steady decrease in the line as it goes from 10-22 kHz.

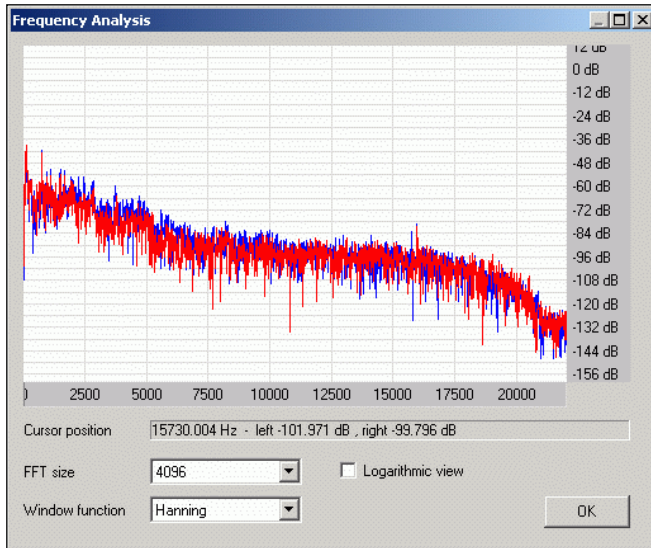


Figure 1 : Frequency analysis of an original wav [3]

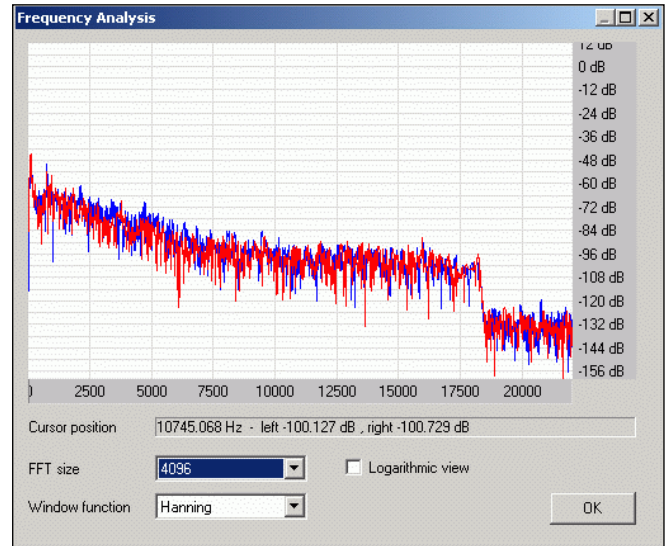


Figure 2 : Frequency analysis of a mp3-sourced file [3]

Reversing the situation by decoding the MPEG-2 Audio Layer III (mp3) of the same song to a wav and performing the same “frequency analysis”, a noticeable drop is clear to a much lower volume (dB) level at around 17–19 kHz. The result is due to the fact that an mp3 recording does most compressions at high frequencies, because they are least noticeable and have a high number of bits, so it would make sense for the most compression to take place there, as shown in **Figure 2**

You can see more differences than you can hear in **Figure 1** and **Figure 2**. The above describes the necessity of Audio Fingerprinting.

Audio fingerprinting can be summarised as follows:

- I. Audio fingerprints of the audio segments that must be identified are generated. These fingerprints are generated from small audio segments, usually between 3 and 30 seconds in length (depending on the technique used), and are stored in a database.
- II. To identify an unknown segment fingerprints are generated from the unknown segment and these fingerprints are compared to all fingerprints stored in the database in order to identify an unknown audio segment.
- III. When a match is found between the audio fingerprint of the unknown audio sample and an audio fingerprint in the database, the unknown sample can be identified.

1.2 *Research question*

Audio fingerprinting is a new concept and therefore its full potential is not yet realised. Although the amount of literature on different audio fingerprinting algorithms has increased, there are seldom comparisons between the algorithms or discussions on different applications other than music identification. The need therefore exists to compare the algorithms for different applications for the algorithms. Wes Hatch identified a couple of different applications for audio fingerprinting as discussed in section 2.4 **Why audio fingerprinting is relevant in the engineering field**, but not all the different applications were implemented [4]. One of the applications that Hatch mentioned that is lucrative and is understudied, is the broadcast monitoring application. Currently, people are employed by broadcasting stations to monitor different radio stations. As human error is a high risk and this can be an unfulfilling job, the need exists to apply audio fingerprinting to the broadcast monitoring problem. In the case of this research, the motivation for investigation was the automated identification of radio advertisements.

After analysing different advertisements refer to **Table 12**, it was determined that advertisements use little or no musical instruments, which result in fewer frequency components. Applying audio fingerprinting to these advertisements raises many questions, including “Which audio fingerprinting algorithm should be used?” and “Does the algorithm as developed for music have to be changed for advertisement purposes?”

This provides one with the research question:

Can audio fingerprinting be applied successfully to advertisement monitoring, and if so, which existing audio fingerprinting algorithm is most suitable as a basis for a generic algorithm and how should the original algorithm be changed for this purpose?

1.3 *Objectives*

The research has the following objectives:

- Understanding all technical aspects relating to audio fingerprinting.
- Acquiring sufficient information concerning different audio fingerprinting algorithms to make a comparison between Haitsma and Kalker’s [2] and Avery Wang’s [1] algorithm.
- Selecting an audio fingerprinting algorithm from comparisons on which to base the generic algorithm.
- Designing the application and implementing the chosen audio fingerprinting algorithm. Then to optimise the generic algorithm for advertisement identification.
- Verifying and validating the algorithm.

1.4 Research methodology

The scientific method followed to achieve the objectives specified, is presented below

1.4.1 Dissertation plan of action

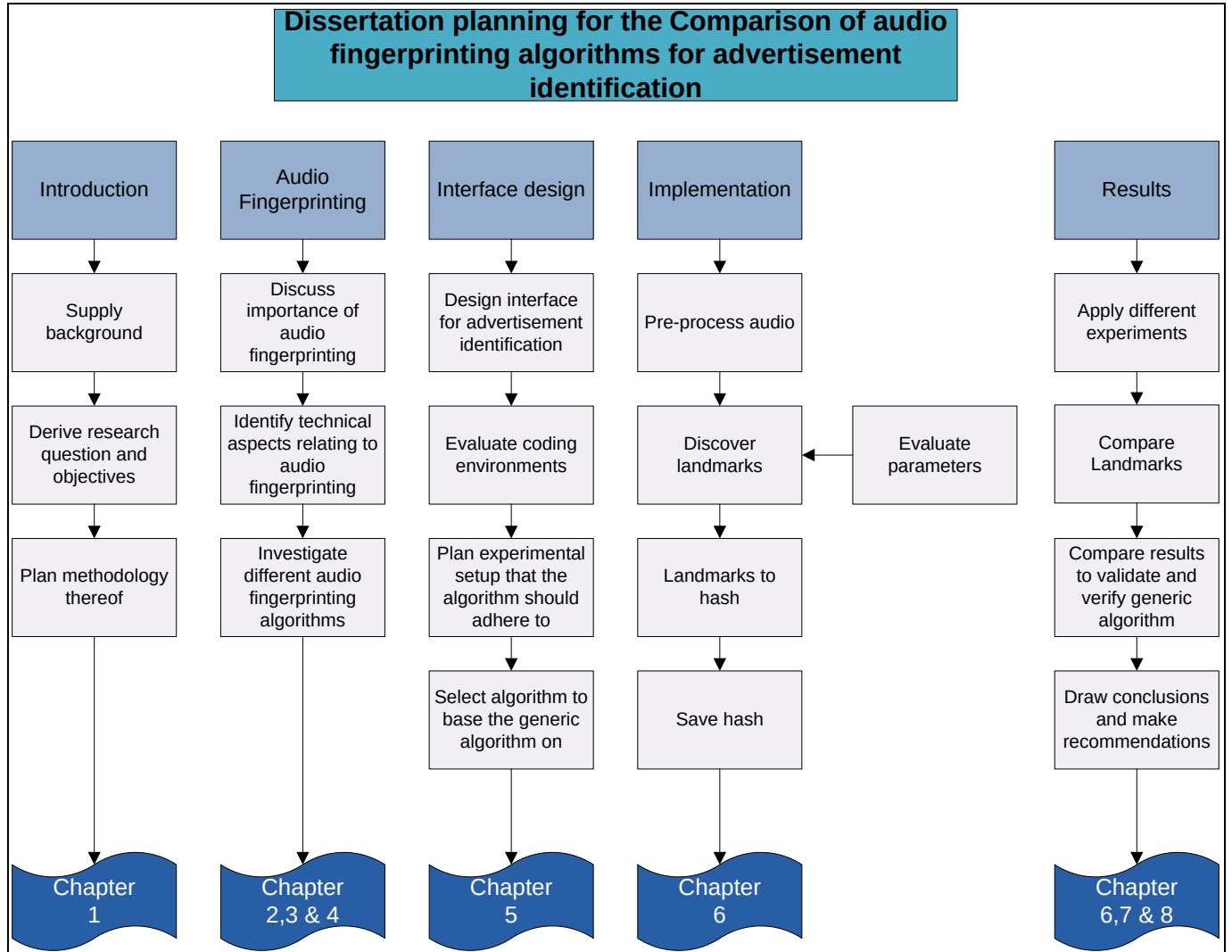


Figure 3 : Dissertation plan

1.4.2 Clarify technical aspects relating to audio fingerprinting

Audio fingerprinting techniques utilise technical aspects known as standard practice in the digital signal processing (DSP) communities. Some of these aspects need to be designed and sufficient literature is needed to optimise the technical aspects. The technical aspects can be considered as a shared basis for all audio fingerprinting algorithms (it might slightly differ, but not in the case of this dissertation). The technical aspects for the algorithm discussed in this dissertation are:

- Filters
- Applying the filter coefficients
- Downsampling

The above technical aspects is for handling less data and increasing the speed of the algorithm (see section **3.1.4 Downsampling** for more detail). The other aspects are:

- Defining unique features
- Storing extracted features in a database

The above two aspects are essential for the audio fingerprinting algorithms and each algorithm uses the aspects in different ways (refer to section **3.2 Unique Features** for more detail).

1.4.3 Study different audio fingerprinting algorithms

After scrutinising the literature and technical aspects relating to audio fingerprinting, different fingerprinting algorithms must be identified. Based on P.J.O Doets' [5] research on the work of P Cano, E Batlle, T Kalker, and J Haitsma [6], there are 3 groups with 3 key algorithms (see **Chapter 4 – Audio Fingerprinting** for a more detailed description). Based on P.J.O Doets' recommendations, three algorithms are identified and considered. This includes, Haitsma and Kalker's algorithm developed for Phillips [2], Avery Wang's algorithm developed for Shazam [1] and Microsoft's RARE algorithm [7], but for this dissertation only 2 algorithms are considered that of Avery Wang's [1] and that of Haitsma and Kalker's [2] refer to section **2.2 Algorithms**.

1.4.4 Selection of audio fingerprinting algorithm

After ample research has been performed, different audio fingerprinting algorithms and technical aspects relating to audio fingerprinting are identified. A selection can be made from the identified algorithms. A generic algorithm will then be produced and based on the selected algorithm. The generic algorithm will be optimised for advertisement identification. The selection is based on theoretical calculations of the database speed. Avery Wang's algorithm was selected for its robustness and more obvious hash table.

1.4.5 Selection of Software environment

Audio fingerprinting algorithms are DSP intensive, therefore an environment that supports memory management and direct access to data should be considered. The Matlab™ environment is considered for prototyping, as it has a clear debug feature and the required toolboxes (see **5.2.1 Matlab** for more information on this remark). The .Net Framework environment is used for the implementation of the generic algorithm, as it is constantly updated and very compatible with everyday software. The framework provides a comprehensive and consistent programming model and a common set of Application programming interfaces (APIs), and this helps to build applications quickly [8] (see section **5.2.2 VB.NET** for more detail on this remark).

1.4.6 Experimental setup

An experimental setup has to be defined to ensure the correct implementation of the generic algorithm and the ability for verification and validation of the generic algorithm. To validate whether the generic algorithm is a reliable audio fingerprinting algorithm, its results will be compared to Haitsma and Kalker's and Avery Wang's algorithm. To verify the algorithm, the functionalities are reviewed to ensure the correct implementation of the functionalities forming the heart of the algorithm (refer to **7.1 Verification and validation**).

1.4.7 Implementation of the generic algorithm

With the experimental setup defined, a proper implementation of the generic algorithm can commence. **Chapter 6 – Implementation**, gives an explanation of the method and the optimisation of the parameters for advertisement identification.

1.4.8 Acquire results

After the implementation of the generic algorithm, results can be obtained. The generic algorithm is verified and validated and thereafter results in terms of advertisement identification can be acquired.

1.4.9 Reach a verdict

After scrutinising the results to determine if the algorithm is an audio fingerprinting algorithm based on the robust landmark method, and able to detect advertisements successfully, a verdict can be reached. Further conclusions are drawn about the algorithm and the practical advertisement application, followed by recommendations for future work.

1.5 Organisation of dissertation

Chapter 1 gives an introduction, enlightening the reader on why audio fingerprinting is used and explaining why we want to use audio fingerprinting for advertisement detection.

In Chapter 2 a slight overview of the literature of audio fingerprinting (more literature is presented as needed in the following chapters) and different algorithms are presented. The differences between advertising (mostly voice) and music are discussed. This is followed by an explanation of why audio fingerprinting is important to the engineering field.

Chapter 3 presents literature about technical aspects relating to audio fingerprinting. The design of the filter, the operation of applying the filter coefficient and the downsampling technique is presented mathematically, followed by the explanation and use of unique features and a background on how the database is used.

A more in depth literature study on audio fingerprinting is presented in Chapter 4. The selected algorithms are discussed in more depth.

In Chapter 5, the desired interface is designed and different environments are discussed for the generic algorithm. The experimental setup is explained for validation and verification purposes.

Chapter 6 presents the implementation process of the generic algorithm. The process is described in terms of five functionalities: pre-processing of audio, landmark discovery, landmark to hash, save hash and compare landmarks. In the functionality of landmark discovery, the parameters are described and their values are motivated.

In Chapter 7 the results are presented. The results are compared to that of Haitsma and Kalker's and Avery Wang's algorithms to validate that the generic algorithm is indeed an audio fingerprinting algorithm. To verify the algorithm, the functionalities are reviewed to ensure the correct implantation of the functionalities and algorithm.

In Chapter 8 conclusions are drawn from the dissertation and the research question is answered. The methodology is discussed and how this helped to achieve the objectives. Finally, recommendations are made for future work.

Before the bibliography is the addendum, the addendum holds relevant results which are not described in a specific section.

This chapter is the introduction of the dissertation. A slight background about the dissertation was given. The research question was compiled which led to identifying the objectives. The research methodology was discussed and the chapter ended with a short organisation of the dissertation. In the next chapter the reasons why audio fingerprinting is used and the difficulties surrounding music and advertisements are discussed.

Chapter 2 – Audio Fingerprinting

In this chapter, literature regarding the background of audio fingerprinting and different audio fingerprinting algorithms are presented. The research question regarding the use of audio fingerprinting in advertisements follows, concluding with a discussion of the importance of audio fingerprinting.

By shape, colour and smell is one of the basic functions how we humans identify objects with. The notion of identifying objects by using any unique feature was further developed and it was soon realised that sound, speech and music was as unique as fingerprints. The same concept for the recognition of fingerprints was designed for audio, hence audio fingerprinting. The potential of constructing fingerprints of audio signals to identify and compare them have several benefits, according to Wes Hatch [4] (see **2.4 Why audio fingerprinting is relevant in the engineering field** for a full description of the benefits). In general audio fingerprinting requires short audio segments, usually between 3-30 seconds in length (depending on the algorithm), to correctly identify a match. These segments are converted to audio fingerprints and are compared to a database of known audio fingerprints, to identify the original audio source (see **Figure 12** for a visual description).

The audio fingerprints of the segments do not necessarily have to be of high quality to be a match. Distortions and interference of the original signal makes matching of the fingerprints less reliable, but it can still be recognisable to a certain extent. The distortions and interferences can be compared to a smudged or partial human fingerprint.

2.1 *The original problem / idea*

The idea of audio fingerprinting came into existence when it was realised that humans had the capability to identify music with little data. The concept was turned into popular game shows like “Face the Music” [9] and others. This led to the obvious idea that music was unique and identifiable.

The problem statement then was, how is it possible for a computer to identify music. Tagging mp3s by inserting metadata into music or a watermark made audio data recognisable by a computer. The problem with the latter technique is that the music had to be tagged for a computer to identify a song, this was a step down from what humans were capable of.

As music became quite popular on radio and the internet, the need to monitor the songs and advertisements played arose. The best tool available at the time, were humans monitoring data in monitoring stations. This had its downside, as humans made errors and tagging was impossible as only audio data was received through a signal.

It became necessary to create a robust and accurate way of identifying audio segments. Because it is clear that music and advertisements are unique, these characteristics were studied further and used to teach a computer to identify them. The solution was audio fingerprinting.

Audio fingerprinting technology is capable of identifying audio signals with the audio's unique features. The features are unique to each audio signal and, in similarity to human fingerprints, are also referred to as audio fingerprints.

In addition to the acquirement of these features, no further processing of the audio signal is necessary. When audio fingerprinting technology is implemented, the audio signal itself is not modified. Recognition of the title is performed exclusively on the basis of content.

Audio fingerprinting can also distinguish between various versions of a particular music recording. This aspect is used to distinguish between the normal and live version of a recording and whether an artist is lip-syncing. These tests would be very difficult for a human to perform.

2.2 *Algorithms*

In this dissertation, the researcher will be concentrating on advertisement identification rather than the popular use of music identification.

Cano et al. presented a good survey of audio fingerprinting algorithms, which allowed P.J.O Doets to complete a comparison of audio fingerprints for extracting quality parameters of compressed audio. P.J.O Doets, M. Menor Gisbert and R.L. Lagendijk classified audio fingerprinting techniques in three groups and ranked the algorithms within each group to the following criteria [5], [6]:

- The algorithm is robust to compression, i.e. the algorithm is capable of identifying an audio signal distorted by compression.
- The algorithm is reported to be robust to common distortions.
- The fingerprinting system is described well enough to be implementable.

Using the above criteria, they have selected one algorithm to represent each group:

Group 1: Systems that use features based on multiple subbands. An example is Phillips' Robust Hash algorithm, which is reported [2] to be very robust against distortions. Phillips Robust Hash technique is based on Haitsma and Kalker's algorithm [2].

Group 2: Systems that use features based on a single band such as the spectral domain, for example Avery Wang's Shazam [1] and Fraunhofer's AudioID algorithms.

Group 3: Systems using a combination of subbands or frames, which is optimised through training. An example is Microsoft's Robust Audio Recognition Engine (RARE) that uses distortion discriminant analysis (DDA) and oriented principal component analysis (PCA) [10].

The features of group 3 are basically the same than those of group 1, only with a different training method. Therefore, it seems unnecessary to compare group 3 to the other groups as it should deliver the same results as group 1. Apart from that, the information about the implementation of the group 3 audio fingerprinting is not readily available [10].

2.3 Advertising vs. music

The popular purpose of audio fingerprinting is to identify and monitor music for promoting artists and royalty collection. In this dissertation, audio fingerprinting is used for the monitoring of advertisements. There are subtle differences between advertisements and music and to understand the difference better there has to be a better understanding of what they actually are and how they differ from each other. Both advertisements and music are composed of frequencies and interpreted by humans as sound.

2.3.1 Sound

Vibrations composed of frequencies, better known as sound, are detectable by the ears. Frequency determines the pitch [11] of the sound. Sound, per definition, is a travelling wave of pressure that oscillates, transmitting through solids, liquids, or gas, and is composed of frequencies within the range of hearing (20 Hz – 20 kHz) and on a level sufficiently strong enough to be heard [12].

The human ear does not hear all the frequencies equally well. Humans hear sounds best at around the 3,000 – 4,000 Hz sampling rate, where human speech is focused [13], [14] (see **Figure 4**).

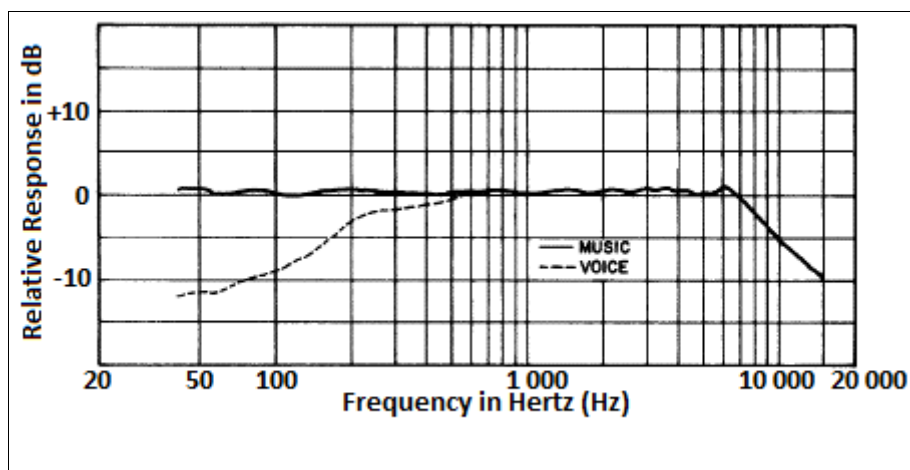


Figure 4 : Typical Frequency Response [15]

With a better understanding of how humans interpret frequencies as sound, the main differences between advertisements and music can now be easily determined. The main difference is that music has a whole variety of frequencies, which means more unique features whereas advertisements consist mostly out of speech.

2.3.2 Speech and music

The human ear is able to identify different frequencies very accurately as well as the combination of frequencies, which we know as speech. Humans interpret frequency combinations as similar, meaning that humans are still able to recognise the word “hello” even if it varies with an accent or a bit of noise, etc.

Music is mathematical or physical relationships in frequency. The human ear is able to recognise all these frequencies and, if the human heard the music before, he/she could identify it without listening to the whole song again. As humans identify sounds with similarity, a song may vary, e.g. be converted to mono and still be recognised by a person. How humans are able to familiarise themselves with different words, sounds and songs even if the frequency combinations differ, is not included in the scope of this dissertation.

2.4 Why audio fingerprinting is relevant in the engineering field

Audio fingerprinting is important to the engineering field, because it leads to beneficial applications for the current fast pace lifestyle. Wes Hatch [4] has identified the following benefits:

- Broadcast Monitoring, this entitles royalty collection, sensitive screening for a code of conduct by the broadcast and complaints commission and easy advertisement verification.
- Media plugins, generating tags, CD covers and information etc. on music tracks.
- P2P (peer-to-peer) filtering, can be used to combat piracy through scanning pirate sites and still identifying tracks even when the tracks has been doctored and mislabelled.
- Video fingerprinting is slow, it can thus benefit from the use audio fingerprinting techniques to obtain the above benefits in real time.

In this chapter, a brief background of audio fingerprinting was provided, as well as the necessary knowledge of why it is important and the difference between advertisements and music. To acquire a better understanding and more detail of audio fingerprinting, it is time to shift the attention to the technical aspects relating to audio fingerprinting.

Chapter 3 – Technical Aspects Related to Audio Fingerprinting

In this chapter, the technical aspects related to the algorithms are discussed. The technical aspects are not a necessary part of the algorithm, but are needed to construct a successful algorithm. The technical aspects are sometimes referred to as the tools used to construct the algorithm. These aspects entail the filter design, the filter application and the downsampling of the audio. Other important aspects discussed are the identification of the unique features and storage of these features.

Firstly, a slight background on frequencies is presented as it is necessary for the understanding of the technical aspects used in audio fingerprinting. As previously mentioned in section **2.3.1 Sound**, an audio segment is nothing more than a combination of frequencies. In the audio segment, the main interests are the speech frequencies, therefore it makes sense to downsample the audio segment to this frequency. To do this the data needs to be filtered.

3.1 Frequencies

Frequency is the number of sound vibrations in 1 second. The International System of Units (SI) unit of frequency is hertz (Hz) named after the German physicist Heinrich Hertz [16]. The frequency range of interest for this study is the frequency of waves between approximately 20 Hz and 20 kHz [17], which is audible by human ears. Frequencies below 20 Hz can easier be felt rather than heard when the energy from the amplitude of the vibration is high enough. Frequencies above 20 kHz can sometimes be detected by adolescents, but a human's ability to hear high frequencies are the first to be affected by hearing loss due to age and/or continued exposure to very loud noises.

3.1.1 Pre-processing

Pre-processing requires a signal to be downsampled to 8 820 Hz with a lowpass finite impulse response (FIR). As human speech is focused between 3,000 – 4,000 Hz refer to section **2.3.1 Sound**, the sample rate of 8 820 Hz is chosen with regard to the Nyquist theorem [19], [20] (twice the sample rate of the desired frequency), thus allowing 820Hz for transition width refer to section **3.1.2 Filters**. A FIR filter is used for the following properties:

- No feedback is required, resulting in rounding errors that are not compounded by its summed iterations [18].
- They are very stable. As a result of no feedback required, all the poles are situated at the origin which means they are positioned within the unit circle [18].
- As the signal is downsampled to 8 820 Hz, the cut-off requirement of 820 Hz is met (refer to section **3.1.2 Filters**). It is linearly designed through the use of symmetric coefficients, thus an equal delay to all frequencies [18].

- Most wav files and MP3s decoded to wav files have a sample rate of 44.1 kHz. When it is downsampled to 8 820 Hz, a downsampling factor of exactly 5 is obtained, which results in high speed increases because less data is being handled.

In the pre-processing stage, the audio segment is converted to mono if the segment is in stereo, by adding the two channels and using the average. If a signal is provided in mono, the pre-processing stage is ignored. The audio description is then stored into a lookup table. As the application is a practical approach to identify advertisements, the popular sampling frequency of audio is in stereo at 44.1 kHz in a 16-bit format. **Figure 5** presents an example of downsampling. All audio samples are downsampled to 8 820 Hz (discussed in more detail in **3.1.2 Filters** and **3.1.4 Downsampling**) irrespective of the format which it was presented in.

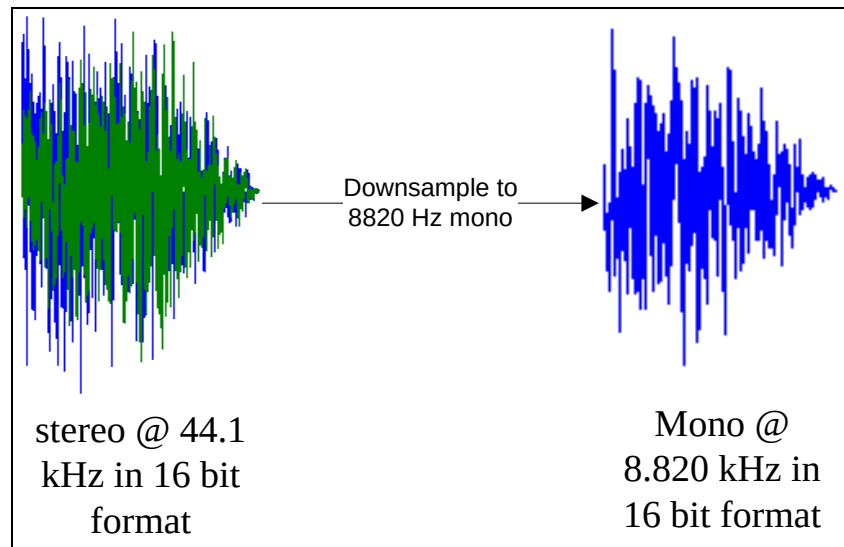


Figure 5 : Stereo 44.1 kHz conversion to Mono 8 820 Hz

3.1.2 Filters

Designing a filter requires the 5 following steps [19]:

- I. Filter specifications – This includes stating the type of filter, desired frequency with its desired amplitude and/or phase responses, the sampling frequency, and the length of the input data.
- II. Coefficient calculation – The coefficients of a transfer function $H(z)$ are calculated, which satisfies the specification given in I. The choice of the coefficient calculation method will be influenced by several factors, the most important of which are the critical requirements in step I.
- III. Realisation – Converting the transfer function obtained in step II into a suitable filter structure.
- IV. Analysis of finite lengths effects – Analysing the effects of the number of filter coefficients and the input data. Also analysing the filter performance through the operation of filtering using fixed lengths.
- V. Implementation – Producing the software code and performing the actual filtering.

Following the above steps, the desired filter requirements are calculated.

Specification of the filter requirements

$$F_p = 8000 \text{ Hz} \quad F_T = 44.1 \text{ kHz}$$

$$F_s = 8820 \text{ Hz} \quad \Delta F = F_s - F_p$$

$$\alpha_p = 0.1 \text{ dB} \quad \alpha_s = 60 \text{ dB}$$

$$\omega_c = \frac{2 \times \pi \times F_c}{F_s} \quad F_c = \frac{F_p + F_s}{2}$$

F_p – Passband edge

F_T – Sampling frequency

F_s – Stopband edge

F_c – Cut off frequency – ω_c (radians)

α_p – Passband ripple

α_s – Stopband attenuation

ΔF – Frequency difference

The audio fingerprinting algorithms apply a lowpass finite impulse response (FIR) filter, as it is the lower frequencies humans identify with. A FIR filter is a signal-processing filter, which impulse response is of finite length [19]. There are various ways of implementing a lowpass FIR filter, one of the popular methods is the window-based FIR filter design [20]. The most widely used adjustable window is the Kaiser window, presented by J.F. Kaiser [21] also known as the optimal window.

The Kaiser window also allows for fewer coefficients for the same optimal response than the other windows [19]. As mentioned in section 2.3.1 **Sound**, advertisements consist mostly out of voiced sound segments, and the bandwidth allocated for a single voice-frequency transmission channel is usually 4 kHz, including guard bands, allowing a sampling rate of 8 kHz [22], [23]. Since advertisements are read in much the same way as a normal conversation, the sound level is typically between 55 and 60 dB [24].

The listed requirements allow for the desired voice-frequency transmission channel in accordance with its Nyquist rate and the stopband edge (F_s) allows fast downsampling. Determining the beta (β) of the Kaiser window requires the passband (δ_p) and stopband (δ_s) peak ripple, which are acquired through:

$$20 \log(1 + \delta_p) = 0.1 \text{ dB} , \text{ giving } \delta_p = 0.0115 \quad (1)$$

$$-20 \log(\delta_s) = 60 \text{ dB} , \text{ giving } \delta_s = 0.001 \quad (2)$$

Thus

$$\delta = \min(\delta_p, \delta_s) = 0.001 \quad (3)$$

delivering

$$\beta = 0.1102(\alpha - 8.7) = 5.65 \quad (4)$$

With the above information the Kaiser filter coefficients can now be calculated. The length of the filter must always be odd and is calculated as follows:

$$N \geq \frac{\alpha - 7.95}{14.36 \left(\frac{\Delta F}{F_T} \right)} = 194.9355 \approx 195 (\text{round up to nearest integer}) \quad (5)$$

The number of coefficients (N) are not determined exactly, as round off errors could cause a decrease in accuracy. In practice the number of coefficients are increased with 2 when odd ($N + 2$) and 1 when even ($N + 1$) to ensure an accurate filter.

thus $N = 197$.

The Kaiser window coefficients ($\omega[n]$) are given by [21]:

$$\omega[n] = \frac{I_o \left\{ \beta \sqrt{1 - \left(\frac{n}{M} \right)^2} \right\}}{I_o \{\beta\}}, \quad -M \leq n \leq M \quad \text{where } M = \text{round} \left(\frac{N}{2} - 1 \right) \quad (6)$$

where $I_o(x)$ is the modified zeroth-order Bessel function, which can be expressed in a power series form:

$$I_o(x) = 1 + \sum_{k=1}^L \left[\frac{\left(\frac{x}{2} \right)^k}{k!} \right]^2, \quad \text{Where } k \text{ is the steps, } L < 25 \text{ and } x = \frac{n+1}{2}. \quad (7)$$

$L < 25$ is due to Kaiser (Rabiner and Gold, 1975). See [19] for an efficient implementation of this equation. As the modified zeroth-order Bessel function $I_o(x)$ has mirror values, only one half has to be calculated

$$I_o(x) = \begin{cases} x, & M \leq 0 \\ -x, & M > 0 \end{cases} \quad (8)$$

The multiplication of the Kaiser window coefficients ($\omega[n]$) with an impulse response of an ideal low pass filter (h_{LP}), gives the desired filter response, where h_{LP} is

$$h_{LP}[n] = \begin{cases} \frac{\sin(\omega_c(n-M))}{\pi(n-M)}, & 0 \leq n \leq N-1 \\ 0, & \text{otherwise} \end{cases} \quad (9)$$

thus

$$h[n] = \omega[n] \times h_{LP}[n] \quad (10)$$

In **Figure 6** the impulse and frequency response of the calculated filter can be seen.

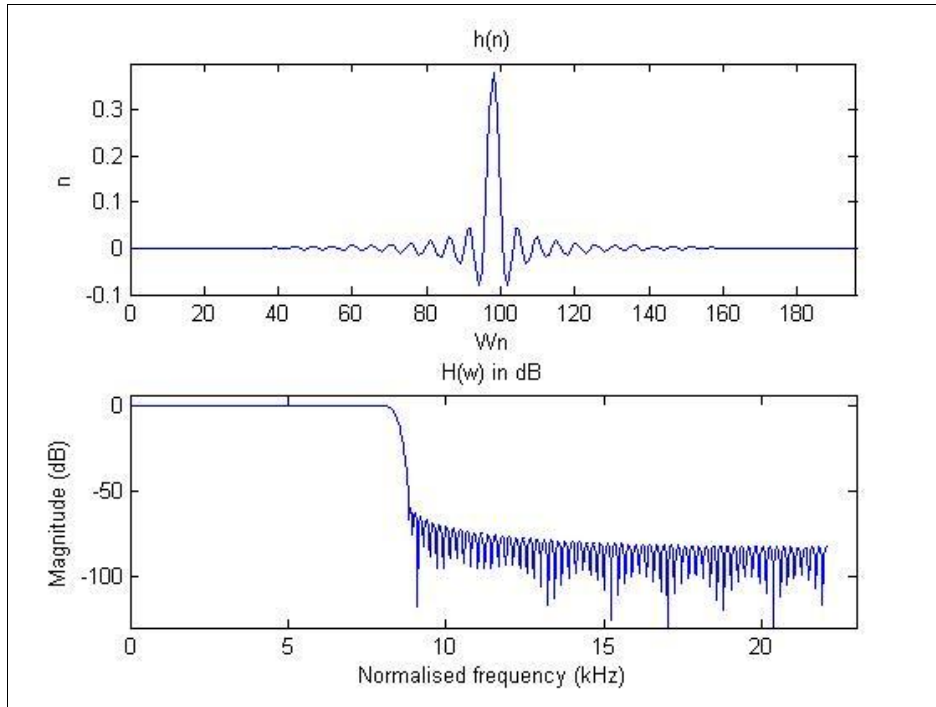


Figure 6 : Filter impulse and frequency response

After the filter coefficients are calculated, it needs to be applied to the audio (see filter coefficients in **Addendum**). Next, the operation of applying the filter coefficients is discussed.

3.1.3 Applying the filter coefficients [19]

To apply the filter coefficients, the coefficients need to be convoluted with the signal. There are two methods of convolving a signal: one is the direct method and the other fast convolution.

The direct method can be described through the equation $y(n) = \sum_{m=0}^n h(n-m)x(n) = x(n) \otimes h(n)$

When multiplying the two polynomials in the above fashion the convolved signal is calculated. This is known as the Cauchy product. The necessary computations for the direct method are that each value of $x(n)$ should be multiplied with $h(n)$ giving $n \times n = n^2$ multiplications.

The fast convolution algorithms use fast Discrete Fourier transform (DFT) algorithms. The best method for the long signals will be via the circular convolution theorem, but for FIR filters there is a special case of circular convolution which is known as the overlap-add method [19].

The overlap-add method can be described by $y(n) = \sum_k x_k[n-kL]h(n-m)$ where x_k is the signal x divided in k sections.

Consider the fast convolution algorithm of the same n -point sequences, increasing each sequence with zeros to $2n-1$ (the length of the product of convolution from two signals are the sum of their lengths added, -1)

approximating $2n - 1 \approx 2n$. The number of complex multiplications for an n -point DFT was shown to be equal to $\frac{n}{2} \log_2 n$, so for the $2n$ -point DFT it is equal to $\frac{2n}{2} \log_2 2n$.

The fast algorithm requires two DFTs and one inverse discrete Fourier transform (IDFT). The algorithm therefore requires computation of three $2n$ -point DFTs involving $3n \log_2 2n$ complex multiplications.

It is also necessary to evaluate the complex multiplications of $\text{DFT}\{x(n)h(n)\}$, which increase the number of complex multiplications to $3n \log_2 2n + 2n$. Each complex multiplication of the form $(A + jB)(C + jD)$ requires 4 multiplications, hence $4 \times (3n \log_2 2n + 2n) = 12n \log_2 2n + 8n$ are the total real multiplications necessary.

Table 1 is drawn to illustrate when the fast convolution algorithm gains advantage.

Table 1 : The number of real multiplications necessary for the convolution of two n point sequences [19]

N	<i>Direct method</i>	<i>Fast convolution</i>	<i>Ratio, fast: direct</i>
8	64	448	7
16	256	1 088	4.25
32	1 024	2 560	2.5
64	4 096	5 888	1.4375
128	16 384	13 312	0.8125
256	65 536	29 696	0.4531
512	262 144	65 536	0.250
1024	1 048 576	143 360	0.1367
2048	4 194 304	311 296	0.0742

From **Table 1** it can be derived that for any filter larger than 128 points, the fast convolution should be considered. As the filter has 197 points coefficients (see **3.1.2 Filters**), the fast overlap-add method is used.

The overlap-add method performs multiple DFT operations: one DFT of the filter and multiple DFTs on the signal depending on the signal's length. The filter length has to be shorter than the signal and the size of the DFT used. The signal is divided into segments and inserted into block lengths L , where $\log_2(L)$ has to be less than the signal length N . The reason for this is that the cost of the convolution can be associated with the number of complex multiplications involved in the equation. The DFT operation requires large computational resources, so the cost of one DFT is:

$$12DFTsize \log_2(DFTsize \times 2) + 8DFTsize \quad (11)$$

The cost ratio for each operation ($N \times L$) is thus:

$$\frac{N}{L} 12DFTsize \log_2(DFTsize \times 2) + 8DFTsize \quad (12)$$

The overlap-add method therefore becomes advantageous when $\log_2 L$ is less than N [25]. This constant overlap-add (COLA) constraint ensures that the successive frames will overlap in time in such a way that all data are weighted equally. With the use of the Kaiser window, however, there is no overlapping. For the Kaiser window, in contrast, there is no perfect hop size other than 1, meaning the DFT frames do not overlap [26]. The DFT size is a

power-of-two value greater or more than filter size. Values for DFT size that are not powers of two are rounded upwards to the nearest power-of-two value to obtain the DFT size.

The first L samples of each summation are output in sequence. The block chooses the parameter L based on the filter order (number of coefficients) and the DFT size. The two waveforms must both contain the same number of points so that their periodic convolution corresponds to the linear convolution theorem, this is achieved through

$$DFT\ size = L + (filter\ length - 1), \quad (13)$$

thus

$$L = DFT\ size - (filter\ length - 1). \quad (14)$$

Therefore, the 197 point Kaiser filter requires the next-power of two DFT sizes, so the minimum DFT size is 256 and if a 30 second advertisement is used $N = 30 \times 44100 = 1323000$ where $L = DFT\ size - (filter\ length - 1)$. With the acquired information, the DFT size with the least cost can now be calculated. With this example the following table, **Table 2**, is computed:

Table 2 : DFT cost effective

<i>DFT size</i>	<i>L</i>	<i>Cost</i> $12DFTsize \log_2(DFTsize \times 2) + 8DFTsize$	<i>Number of operations</i> $\frac{N}{L}$	<i>Total cost</i> $*1.0e+008$
256	60	29952	22050	6.6044
512	316	66048	4187	2.7652
1024	828	144384	1598	2.3070
2048	1852	313344	715	2.2384
4096	3900	675840	340	2.2927
8192	7996	1449984	166	2.3991

In **Table 2**, it is clear to see that for the overlap-add method with the number of samples N specified above, the fastest DFT size is 2048 points. After the DFT size is calculated, the overlap-add method is implemented. The audio is broken down into blocks lengths L and zero padded to the DFT size and the DFT of $h(n)$ and for each of the block lengths $x_k[n - kL]$ is calculated. Every DFT $\{x_k[n - kL]\}$ is multiplied with DTF $\{h(n)\}$ where the filter signal is the IDFT of the multiplied solution stacked back into the signal:

$$y(n) = \sum_k IDFT\{DFT\{x_k[n - kL]\}DFT\{h(n - m)\}\} \quad (15)$$

After the filter signal $y(n)$ is calculated, it is downsampled before the audio fingerprinting algorithm is implemented [19].

3.1.4 Downsampling

Downsampling is necessary for the following reasons:

- Reduces data, which makes the data easier to handle.
- Decreases database size.
- Eliminates unwanted higher frequencies.
- Increases speed of the algorithm.

With the high frequencies filtered out, the downsampling can commence. For speed purposes and the transition band F_b with width 820 Hz, the signal can be downsampled with a whole integer 5 (Q referred to as downsample factor). The downsampled signal is every Q^{th} sample of the filtered signal, where the down sampled signal (y_{ds})

$$y_{ds}(n) = \sum_Q y(Q) \quad 0 \leq n \leq \frac{\text{number of samples}}{5} \quad (16)$$

It should be noted when there is not a rational number for the downsampling factor i.e. 441/80 (downsampling 44.1 kHz to 8 kHz), the technique used is that from Matlab™, *upfirdn*. The *upfirdn* is a function, which upsamples the array with the downsampling factor's **denominator** through zero padding, applying the FIR filter and then downsamples the array with the downsampling factor's **numerator**.

Here follows a mathematical explanation on the advertisement identification function *upfirdn*: Denote array as $F(x) = \{x_0, x_1, \dots, x_n\}$. To downsample the array without losing valuable information the downsampling factor σ is calculated accordingly:

$$\sigma = \frac{\text{desired frequency}}{\text{actual frequency}}$$

$$\sigma = \frac{P}{Q} \quad (17)$$

By calculating the smallest rational factor, the following values are derived: the upsampling amount, Q , and the downsampling amount, P , using σ results and minimum CPU processing, which results in a speed advantage. The array $F(x)$ is zero padded in between array elements with amount Q , then filtered with a FIR and downsampled by discarding every P^{th} value. See **Figure 7** for an example

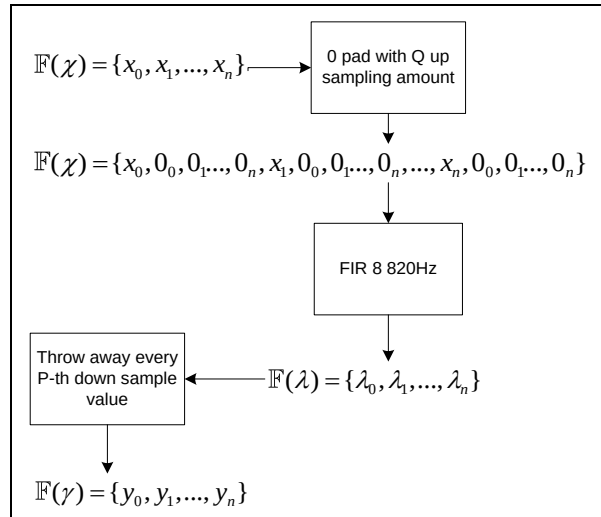


Figure 7 : Upsampling, filtering and downsampling signal to 8 820 Hz

3.2 Unique Features

Unique features allow identification of the object without taking the whole object into account. Features that are truly unique, do not necessarily hold any information on the object except that it belongs to the object. If any information about the object must be acquired, unique features must be determined and relative information should already be stored, in order to link the unique features with the required information e.g. fingerprints. The same goes for audio segments, but before discussing how to identify the unique features in the downsampled signal $y_{ds}(n)$ a short review follows on how unique features are used to identify objects.

3.2.1 Fingerprints

A fingerprint is an imprint of the friction ridges of all or any part of the finger [28]. A fingerprint itself does not hold the name, blood type or any personal information of a person, but because it is truly unique, as proven by Sir Francis Galton [2], the data is linked to the fingerprint in question. Through the uniqueness of a fingerprint, it is possible to identify a person through his/her fingerprint (taken elsewhere) if the information was previously required and linked to his/her fingerprint.

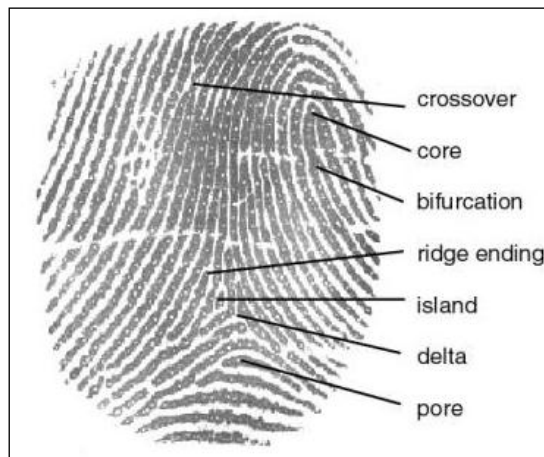


Figure 8 : Fingerprint

In modern times, fingerprints are mostly used to solve crimes. When a fingerprint is lifted at a crime scene, the fingerprint is run through a large database (using pattern recognition) and if there is a fingerprint match, the criminal is identified. Another example is in security systems, where issuing your fingerprint allows you clearance when the system finds your match and positively identifies you. For an example of the unique features of fingerprints refer to **Figure 8**.

3.2.2 *Pattern recognition*

According to Richard O Duda, Peter E Hart, and David G Stork [29] pattern recognition is "the act of taking in raw data and taking an action based on the category of the pattern". Pattern recognition aims to categorise data (patterns) based either on a priori knowledge or on statistical information extracted from the patterns. The patterns to be classified are usually groups of measurements or observations, defining points in a proper multidimensional space. Pattern recognition's popular use is fingerprinting, which has the following fundamental steps [28]:

- Pre-processing – There are functions that are required before main data analysis and extraction for information can commence. The pre-processing functions highlight the unique features and present them in a useful way.
- Main data analysis – After pre-processing, the unique features are mathematically converted and stored. This allows for quicker matching of the data.
- Searching – When handling unknown data, the data goes through the pre-processing and conversion to a mathematical representation. The only difference is now the database is searched with this information instead of stored.

For the final results of the above steps refer to **Figure 9**

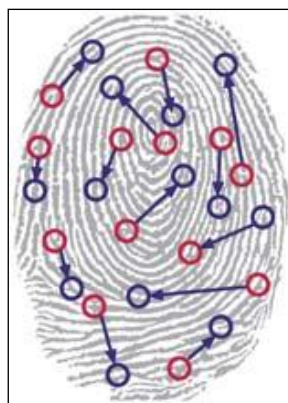


Figure 9 : Fingerprint pattern recognition

3.2.3 Unique features in the downsampled signal

As humans identify words, speech and music with frequencies, all the different algorithms tries to capture this unique feature. A standard practice to emphasise frequencies is to compute a spectrogram (refer to *equation 18*) and space the energy in a logarithmic manner. Different algorithms utilise this feature in different ways, in section 2.2 **Algorithms** it is agreed that group 3 would deliver similar results as group 1 thus it is unnecessary to compare group 3.

Group 1 utilises the spectrogram by computing binary representations of the audio's energy (refer to **Figure 10**). Each algorithm has its own way of calculating the bit presentation. Haitsma and Kalker's algorithm [2] is of importance in group 1. More detail on their bit representation is presented in the section **4.1 Group 1: Haitsma and Kalker's algorithm [2]**.

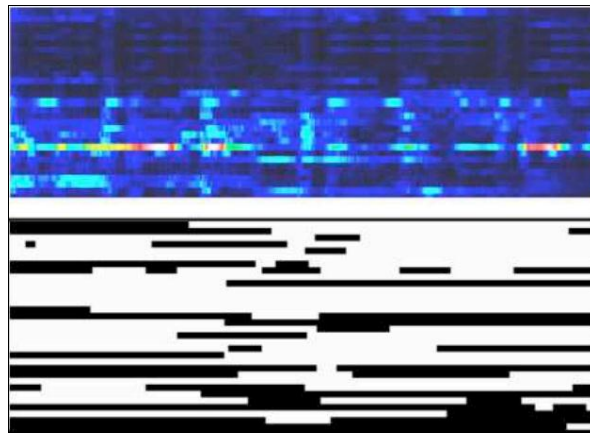


Figure 10 : Example of bit representation [30]

Group 2 utilises the spectrogram by identifying prominent frequency peaks (refer to **Figure 11**). Different algorithms from group 2 utilise these peaks differently. The study of Avery Wang's algorithm is of importance for group 2. Wang's utilisation will be studied further in section **4.2 Group 2: Avery Wang's Shazam**.

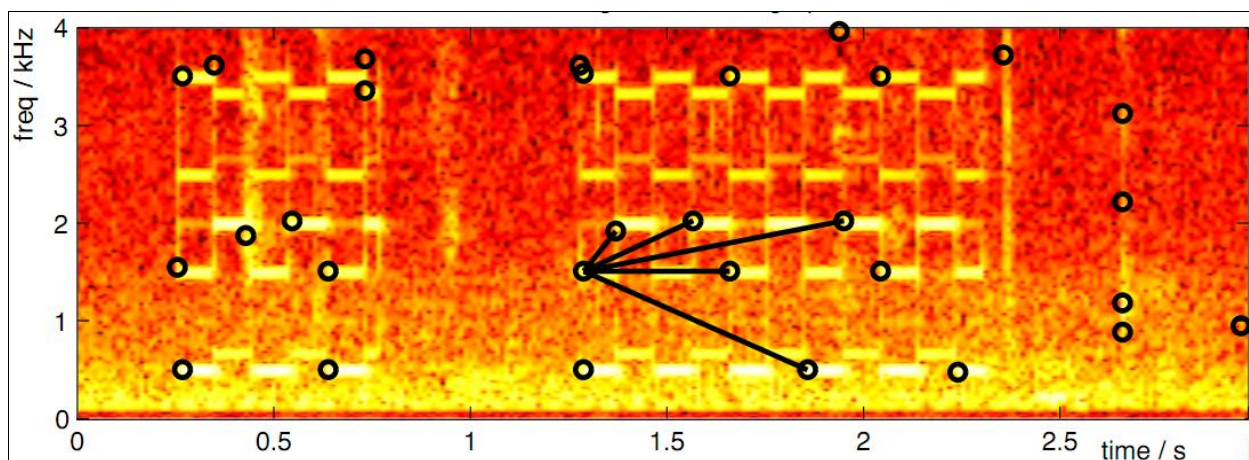


Figure 11 : Example of spectrogram peaks [31]

3.2.4 Database

All the algorithms require the use of a database, as with the fingerprint example the necessary data is stored with the identifier (fingerprints). The algorithms have different ways of storing their data. In group 1, it is a binary code grouped into blocks (audio fingerprint) defined by the specific algorithm. In group 2, the peaks are grouped in a mathematical manner as defined by the algorithm. When the database has sufficient data, the searching and comparing can commence. When working with an unidentified audio sample, it follows the same procedure as specified by its algorithm, the only exception is that the “fingerprints” are not stored but compared.

One flaw in group 1 is that the unknown sample has to be compared to every fingerprint in the database, whereas in group 2 the data only has to be recovered from the fingerprints with the same mathematical solution.

In this chapter, a better understanding of the technical aspects for audio fingerprinting was discussed. The filter calculations and method of applying the filter was discussed and motivated. A brief history of the unique features for audio fingerprinting was given. This chapter ended with a brief description of the different group's fingerprints and database techniques. With a better knowledge of the technical aspects, the audio fingerprinting algorithms can now be discussed in more depth.

Chapter 4 – Audio Fingerprinting

In this chapter, the chosen algorithms are discussed in depth. First Haitsma and Kalker's algorithm is presented, followed by Avery Wang's and lastly a brief description of Microsoft's RARE algorithm.

All of the algorithms follow the same basic principle of fingerprinting discussed in paragraph 3.2.1 **Fingerprints**. In **Figure 12** a more detailed description is presented.

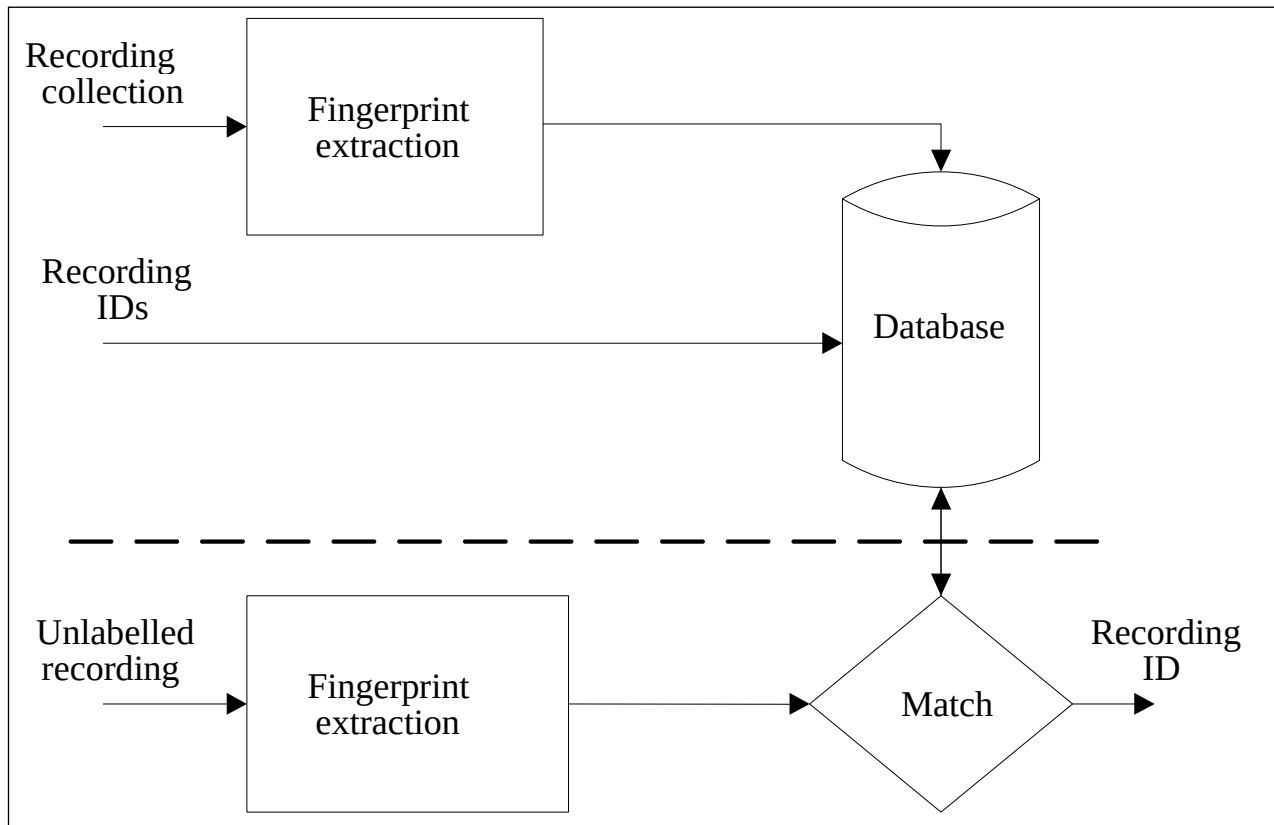


Figure 12 : Content-based audio identification framework [32]

This framework is used after the technical aspects have been implemented. The fingerprints are extracted and stored with the necessary metadata in a database. The same procedures follow when identifying the unlabelled recording where the fingerprint is extracted and compared to that of the database. With the acquired basic knowledge of audio fingerprinting, it is time to proceed to the various algorithms.

4.1 *Group 1: Haitsma and Kalker's algorithm [2]*

Haitsma and Kalker's algorithm proposes a fingerprint scheme based on a general streaming approach. It takes an audio signal and frames it into windows of 370ms in length for every 11.6ms, thus obtaining an overlapping factor of 31/32.

Next, the spectral information of all 32 frames is calculated by means of Fast Fourier Transforms (FFT). Only the absolute value of the spectral information is used because the Humans Auditory System (HAS) is not sensitive to phase response.

The 32-bit sub-fingerprint is the compact representation of single frame and 3 sec worth of sub-fingerprints are defined as a fingerprint block typically containing 256 sub-fingerprints [18]. For each frame, 33 non overlapping bands are selected to extract the 32-bit sub-fingerprint. These bands lie in the range from 300 Hz to 2 000 Hz and have logarithmic spacing. The logarithmic spacing is chosen because it is well known that HAS operates in the approximate bark scale (The bark scale is a psych acoustical scale, it is named after Heinrich Barkhausen who suggested the first independent measurements of loudness) [2].

The sub-fingerprint generated from one frame is not sufficient for an identification match. The algorithm requires a sequence of sub-fingerprints as previously defined as a fingerprint block, typically 8192 fingerprint bits. As suggested, for a reliable match the Bit error rate (BER) should be below 35%, thus of the 8192 bits 2867 fingerprint bits of a fingerprint block can be erroneous [18].

Researches realised that the energy better known as the spectral domain is the most unique representation of audio signals and that the same audio signal in different formats has similar spectral representations [1], [2], [4], [10], [18], [32]-[35].

The energy or spectrogram is the squared magnitude of the DFT.

$$Spectrogram(t, \omega) = \sum_k |DFT(t, \omega)|^2 \quad (18)$$

Haitsma and Kalker realised through experimental results that energy differences are very robust against various types of processing. For a mathematical representation [2], we denote the energy of band m of frame n by $E(n, m)$ and the m^{th} bit of the sub-fingerprint of frame n by $F(n, m)$. The bits of the sub-fingerprint are formally defined as

$$F(n, m) = \begin{cases} 1 & \text{if } E(n, m) - E(n, m+1) - (E(n-1, m) - E(n-1, m+1)) > 0 \\ 0 & \text{if } E(n, m) - E(n, m+1) - (E(n-1, m) - E(n-1, m+1)) \leq 0 \end{cases} \quad (19)$$

The above mathematical representation supplies the means of calculation of the 32 bits of a sub-fingerprint i.e. when the first bit and frame of the sub-fingerprint is calculated – one has the following representation

$$F(0,0) = \begin{cases} 1 & \text{if } E(0,0) - E(0,1) > 0 \\ 0 & \text{if } E(0,0) - E(0,1) \leq 0 \end{cases} \quad (20)$$

See **Figure 13** for a flow diagram of the mathematical representation.

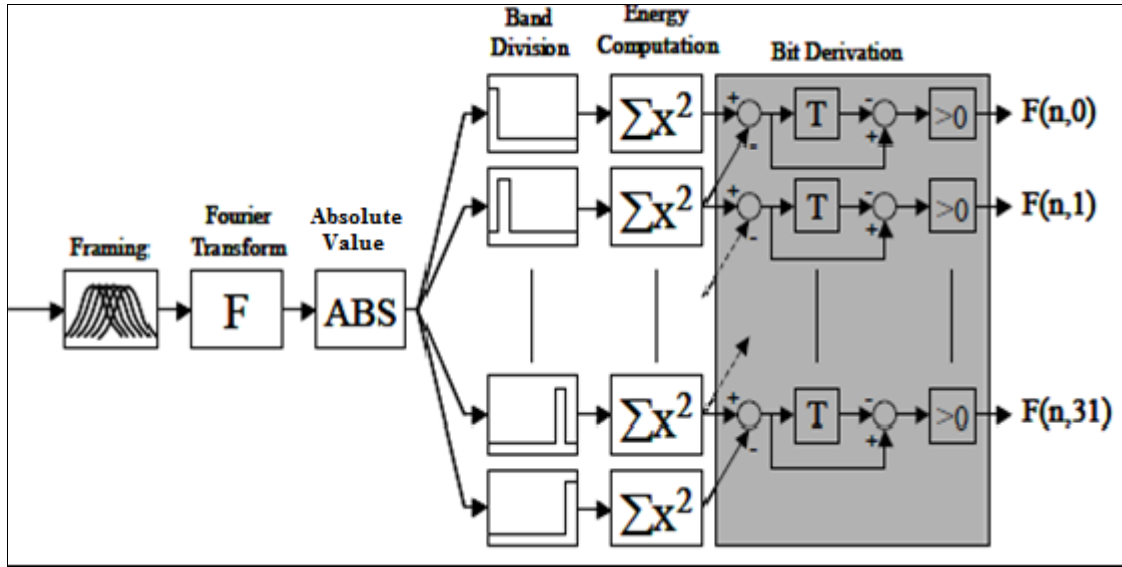


Figure 13 : Haitsma and Kalker's algorithm [2]

Haitsma and Kalker's algorithm requires 3 seconds of an unidentified sample to identify a match. To distinguish between the similar and dissimilar audio signals there has to be high probability difference. Haitsma discovered a 35% BER threshold τ , to distinguish between similar and dissimilar audio signals. When represented mathematically the audio fingerprint block is represented as F , x and y as the unidentified sample and the database match respectively.

Objects are similar if

$$\|F(x) - F(y)\| \leq \tau \quad (21)$$

and objects are dissimilar if

$$\|F(x) - F(y)\| > \tau . \quad (22)$$

From the reception of the audio signal, it is downsampled to a mono audio with a 5 kHz sampling frequency. As the algorithm already includes protection against degrading (through overlapping frames) a simple FIR filter can be used.

$F(0,0)$	$F(0,1)$	.	.	.	$F(0,31)$	1	0	.	.	.	0
$F(1,0)$	$F(1,1)$	.	.	.	$F(1,31)$	0	1	.	.	.	1
.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.
$F(32,0)$	$F(32,1)$	.	.	.	$F(32,31)$	1	1	.	.	.	1

Figure 14 : Haitsma and Kalker's audio fingerprint block

This type of block (see **Figure 14**) is inserted into the database. The identification of each fingerprint block is XOR-ed with the rest in the database to find a match.

4.2 Group 2: Avery Wang's Shazam algorithm [1]

Avery Wang claims that for a database of 20 thousand music tracks implemented on a PC, the searching time is 5 to 500 milliseconds. The code for this algorithm is not directly available, but the generic code for Matlab™ for this algorithm was generated by Dan Ellis [36]. Robert Macrae of C4DM Queen Mary University London adjusted the Matlab code for use in the windows environment [36]. In our application, the code was altered for advertisement identification in VB.NET.

The Shazam algorithm makes use of the audio signals' energy, better known as its spectrogram. The FFT size is typically 512 points, which are referred to as windows or frames. This is the shared basis of group 2. The differences between the fingerprint algorithms in group 2 typically involve how much the frames overlap, how the fingerprint is defined in the frame as well as the storing and searching for the fingerprints.

Avery Wang's Shazam algorithm uses the energy peaks in the frame to form spectral pair landmarks. The algorithm uses spectral peaks for their robustness against noise and approximate linear superposability [1]. The local maxima within a defined section are grouped into pairs [33] and the hash values are computed from the pairs, see **Figure 15**, and then compared. The entry with the most hits is the match (Typically more than 9 spectral peaks are considered a match [33]).

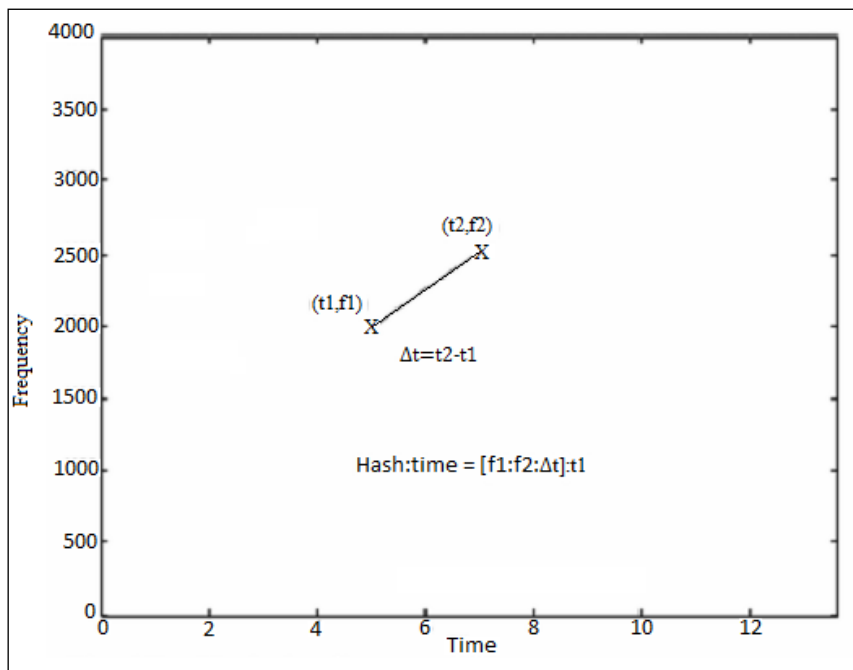


Figure 15 : Hash details [1]

To acquire these spectral peaks, the audio is pre-processed (filtered and downsampled). Next the DFT of the downsampled audio is calculated. This in turn is converted to the spectral range (energy).

The spectral data is then normalised in a basic logarithmic fashion for more explicit values. The algorithm creates a mask of the average of the frames and subtracts this mask from the spectral data to combat noise degradations. Next, a threshold τ of the maximum spectral values of the first 10 frames is created. The threshold consists of local maximums of the 10 frames acquired through a spread function [36]. The local maximums are then smoothed with a Gaussian curve to create the threshold.

The visualisation of the data is DFT size columns by n rows, whereas n depends on the length of the array (audio signal).

$$\tau \begin{bmatrix} S_{\max 1} \\ S_{\max 2} \\ \dots \\ S_{\max n} \end{bmatrix} = \begin{bmatrix} \dots & S_{\max 1} & \dots & n \\ S_{\max 2} & \dots & \dots & n \\ \dots & \dots & \dots & n \\ n & n & n & n \end{bmatrix} \quad (23)$$

If a spectral value is above the threshold, that spectral value forms a spectral peak. The peaks are grouped and referred to as landmarks, and then the landmarks are translated to a hash code (refer to **Figure 15**). The landmarks are then finally stored in the hash table.

4.3 *Group 3: Microsoft's Robust Audio Recognition Engine (RARE) [7]*

Microsoft's RARE (Robust Audio Recognition Engine) algorithm is based on distortion discriminant analysis (DDA) [10], [37]. The RARE engine identifies audio in real-time (database of ¼ million songs) with approximately using only 10% CPU on 850 MHz Pentium 3.

It should be noted that Microsoft has patented their algorithm.

The audio is pre-processed (filtered and downsampled to 11 kHz mono). The algorithm creates a time-frequency map (better known as a spectrogram of size 2048) and uses the magnitude (dB). The audio is de-equalised by flattening the spectrogram, the goal is to remove slow variations in the frequency space not audible by humans. RARE use 6 seconds of audio for their sub-fingerprint converting it to 64 floating point. A frame is created with a length of 372ms and repeats every 186ms, giving an overlap factor of 0.5. This is demonstrated in **Figure 16**.

DDA is the linear method which maximises the signal-to-noise variance and is equivalent to that of a convolutional neural network whose weights are trained using oriented principal component analysis (PCA) (refer to **Figure 17**) [7], [38].

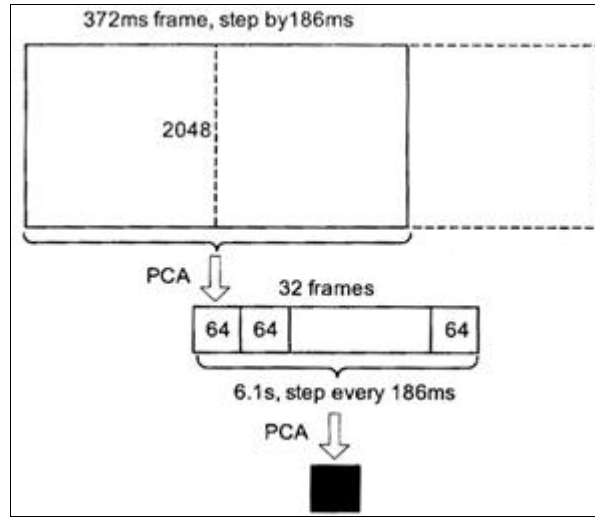


Figure 16 : Microsoft fingerprint extraction scheme

The variance σ of the signal along song A and the distorted version is the sum of the squared differences of song A and the distorted version. To find the smallest variance, song A is distorted with 11 distortions (not specified) and trained accordingly. In reference to **Figure 17** the new trained song A is the blue line (a good projection direction) and this is what is meant by DDA.

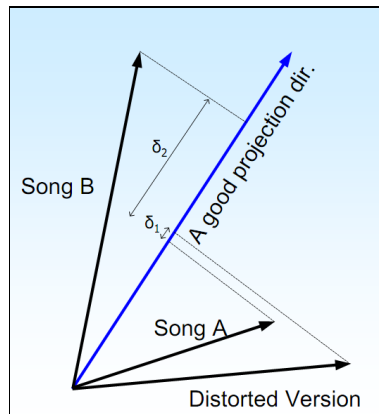


Figure 17 : Oriented PCA [7]

The RARE algorithm is not chosen for this dissertation as it shares similar features of group 1 and should deliver similar results. Also the algorithm is patented and not readily available [10].

In this chapter three algorithms were discussed: that of Haitzma and Kalker, Avery Wang's and RARE. It was concluded that RARE does not have to be studied further. After knowledge of the algorithms were acquired, it is safe to move onto the design of our advertisement application and to discuss the various experiments.

Chapter 5 – Interface design and Solution architecture

In this chapter, the interface for advertisement identification is presented. Different solution architectures for advertisement identification are discussed. The presented design will focus on advertisement identification and will help with the validation process of the algorithm.

In this chapter, the solution architecture will be described and reviewed in greater depth. **Figure 18** displays the desired interface design.

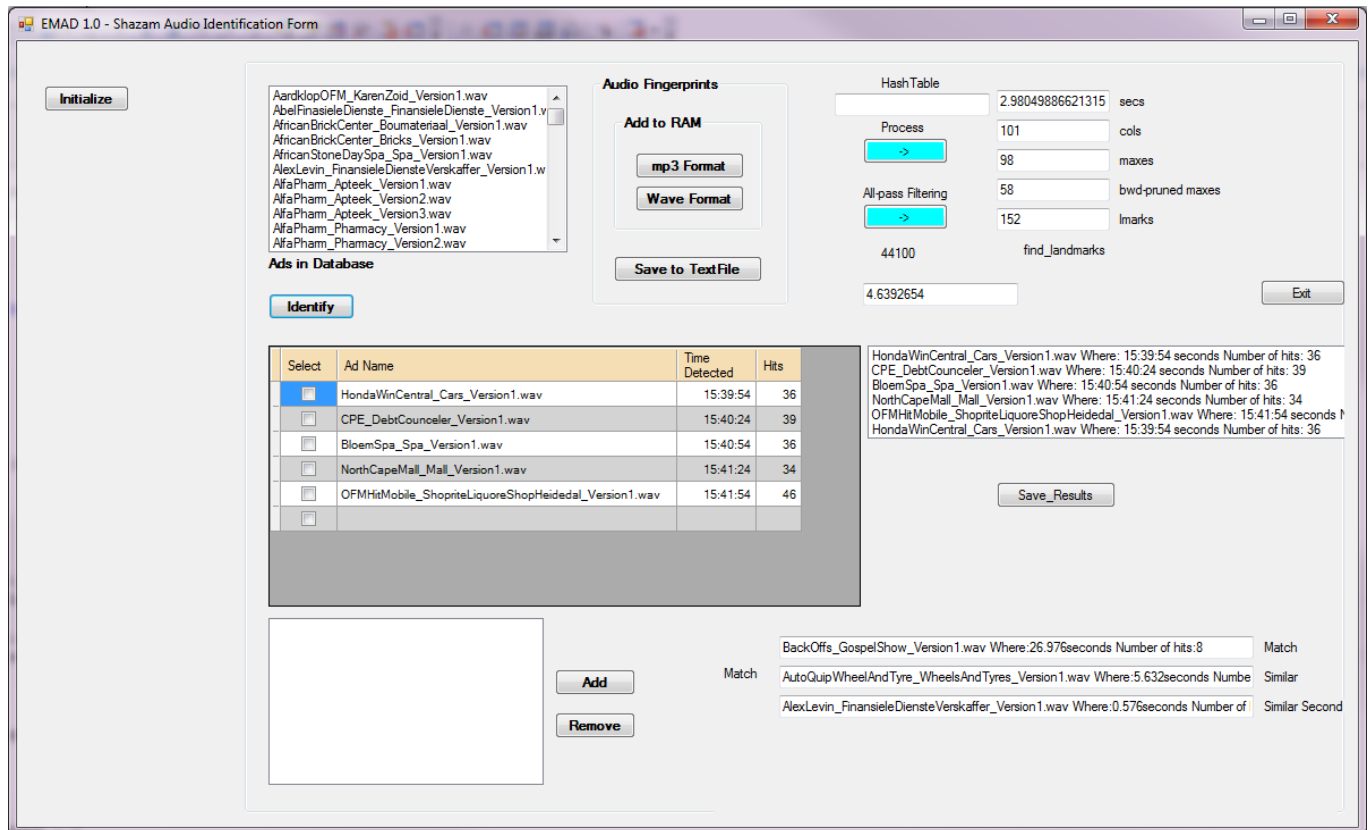


Figure 18 : Advertisement identification functionality

In **Chapter 4 – Audio Fingerprinting**, the basic principles of audio fingerprinting were discussed (**Figure 12**). The principles entailed creating fingerprints of labelled recordings and storing them with the desired information. The unlabelled recording fingerprints are then compared to that of the database to find a match. The designed algorithm works according to the following steps:

- I. **Add to RAM:** The first step in advertisement recognition is to add advertisements to the database.
- II. **Identify and Process:** The next step is to identify an unlabelled recording. Both these basic audio fingerprinting functions are incorporated to the GUI seen in Figure 18. For advertisement identification purposes the radio signal received by Radio Oranje (OFM) is renamed in a special way namely Hour_minute_second.WAV or Hour_minute_second.mp3, which allows the exact time of day to be identified (identify). If an unlabelled recording does not meet this format it can be

added (Add) for identification, which is not necessary for advertisement identification but rather for experimental purposes. There are two options with which to identify the unlabelled recording, one where the recording follows the normal procedure (process) and the other where an all-pass filter is used (all-pass) for experimental reasons.

- III. Lastly, the application allows the database (Save to TextFile) and results (Save Results) to be saved to text files for debugging purposes.

With the desired GUI in mind, the next step is to choose an algorithm to implement for advertisement identification. As the algorithms all prove to be robust and a radio signal is almost noiseless, the deciding factor for choosing the algorithm will be speed. With the increasing trend in technology is in-memory processing as the availability in RAM increases, so does the speed of the in-memory processing. The biggest factor impacting on the speed of the algorithm is the use of the database.

5.1 Database setup and comparison

Haitsma and Kalker's algorithm uses the bit representation and one fingerprint block contains 256 sub-fingerprints worth about 3 seconds. These sub-fingerprints are used with the advertisement ID. As 3 seconds is the minimum time needed to confirm a match [2], this will be the benchmark for our comparisons. This gives the amount of computations for retrieval as

$$256 \times 32bit = 8192bits \quad (24)$$

Where every bit has to be XORed with every bit in the database,

$$8192^2 \times \frac{\text{seconds worth of database}}{3} = \text{Cost}. \quad (25)$$

Avery Wang's algorithm uses a hash table calculated from the landmarks. The total value of hashes per 3 second of audio segment differs. However, it was decided that no more than 40 landmarks will be accepted per second according to Dan Ellis [36], resulting in 120 landmarks each with a 20 bit hash code. The Shazam database cost is calculated as follows

$$120 \times 20bit \text{ hash} = 2400bit \text{ hash}. \quad (26)$$

To translate this to an amount of bit calculations for a fair comparison, it is multiplied with 20 as the base number for the hash value is 20. For a 3 second fingerprint $2400bit \text{ hash} \times 24 = 57600$ (24 entries per hash to acquire desired database size). However, the cost calculation differs (refer to **Chapter 4**) as a hash table does not have to compare all the bits, thus the cost for retrieving a 3 second fingerprint is

$$57600^2 = \text{cost}. \quad (27)$$

As seen in **equation 27** Wang's algorithm's cost will never be higher than 3317.8×10^6 , whereas Haitsma and Kalker's is equal to the same cost as Wang's at 148 seconds worth of database, but continuously increasing with database size. Therefore, the implementation for advertisement identification is a generic code replicating the algorithm of Avery Wang's algorithm. Next, the generic code's database is constructed.

The database should hold at least 5 000 advertisements with an average length of 30 seconds and be as robust as possible, robustness can be perceived as a high number of landmarks per advertisement. As the hash table can only hold $2^{20} \times \text{maxentries}$ (max entries per hash value) = 1048576×24 this leaves us with 5033 landmarks per advertisement. It is not feasible for the whole database to be populated, as the probability of all the combinations of hash values to exist in advertisement data is very small, a 20% deduction is made to the theoretical calculations. A realistic number of landmarks allowed for advertisements are thus 4 026. The algorithm requires 3 seconds of an unlabelled record to identify it. If the unlabelled recording delivers more than 20 of the same landmarks of recording in the database, the recording is identified.

To compensate for compression and other degradations, the desired number of landmarks per advert is thus between 400 and 2 000 (experimental results refer to **6.2.1 Parameters**), which in turn allows a theoretical 10 000 advertisements to be stored. This ensures the database is more than capable of containing 5 000 advertisements.

Haitsma and Kalker's algorithm was implemented by colleague Prof. Willem Venter, therefore all the results from Haitsma and Kalker's algorithm will be drawn from their paper [2] and Prof. Willem Venter's results. Next, the environments for the implementation are motivated.

5.2 *Environments*

A prototyping environment allows for easier implementation and debugging. After the prototyping is successful, the algorithm is ported to a developing environment where the algorithm is fine-tuned and implemented for various advantages. In the next section, the environments (with regard to audio fingerprinting) used in developing the algorithm for advertisement identification is discussed.

5.2.1 *Matlab*

Matlab is a prototyping environment, with the advantage of containing a great deal of engineering toolboxes, including a digital signal processing toolbox. It was easy to use Matlab for short experiments to determine whether an audio fingerprinting algorithm is successfully implemented. Variables are easy to access and could easily be manipulated in debug mode. It is also easy to compare our algorithm to that of Dan Ellis [36], since he also produced a robust landmark audio fingerprinting algorithm in Matlab. The disadvantages of Matlab are that it is expensive and slower than other environments due to the large overheads that it has. After a successful prototype was developed in Matlab, the prototype was translated to improve the speed of the algorithm. The translation was also necessary for a user friendly GUI and the capability of easy integration with other systems.

5.2.2 VB.NET

After the algorithm was prototyped in Matlab, it was translated to Visual Basic .NET Framework (VB.NET). VB.NET has more speed and can integrate different features from the .net framework, including mp3 libraries. The disadvantages of VB.NET is that it does not have a signal processing toolbox and is more difficult than Matlab when it comes to debugging and viewing large variables. VB.NET requires more research in the technical aspects relating to audio fingerprinting as most of the aspects has to be derived from first principles. The implementation of the algorithm in VB.NET provided a platform for an easy design of a specified interface. There is also a large speed advantage gain due to less overheads and the fact that the application is specifically implemented only for audio fingerprinting. With the algorithm implemented in a more practical environment, the experimental setup is explored before proceeding to the implementation.

5.3 *Experimental setup*

We can now discuss the experiments in more detail as it is very important for the verification and validation of the algorithm.

5.3.1 *Data types*

Currently the following formats are supported: mp3, mp2, mp1, ogg, flac, ac3, aac, oga, wav and pcm files as well as streams through the use of libZplay [39].

All formats are however recoded to wav 44.1 kHz 16 bit, where after the audio data is pre-processed (see **3.1 Frequencies**) and then **6.2 Functionality 2: Landmark discovery** is used to discover its unique features and the algorithm continues from there.

5.3.2 *Tests*

The application is tested in terms of accuracy, robustness, speed, versatility and scalability. For verification purposes the same test procedures follows than in Haitsma and Kalker's paper [2]. The audio segments are exposed to the following degradations:

- **MP3 Encoding/Decoding** at 128 kbps and 32 kbps using Xilisoft Video Converter Platinum Copyright © 2008Xilisoft Corporation. **MP3 Encoding/Decoding** is important as it is the most popular format and it undergoes compression (refer to **Figure 1 and Figure 2**). This degradation is used to test current reliability and robustness of the algorithm.
- **All-pass Filtering** used with the following system function: $H(z) = \frac{0.81z^2 - 1.64z + 1}{z^2 - 1.64z + 0.81}$

❖ With the **All-pass Filtering**, it is necessary to see whether the algorithm can handle more data and how the additional frequencies would affect the algorithm.

- **Equalisation:** A 10-band equaliser using AC₃Filter Tools [40] with the following settings is seen in **Table 3**:

Table 3 : Equaliser

Frequency (Hz)	31	62	125	250	500	1k	2k	4k	8k	16k
Gain (dB)	-3	+3	-3	+3	-3	+3	-3	+3	-3	+3

- ❖ **Equalisation** boosts or weakens the energy of specific frequency bands. As the algorithm is based on the energy of the audio, this property is investigated. The equalisation still allows the HAS to identify the audio.
- **Band-pass Filtering** using a band-pass filter, with cut-off frequencies of 100 Hz and 6 000 Hz with a transition bandwidth of 100 Hz and attenuation factor of 100 dB. As the **band-pass filter** allows the most prominent frequencies for voice, it is investigated for advertisement identification. AC₃Filter software was used [40].
- **Time Scale Modification** of +4% and -4% using MAGIX Audio Cleaning lab 16 deluxe [41]. It is important to note that the tempo changes and the pitch remain unaffected. Through the adjustment of the time scale it changes the data in such a way that it may look mathematically (not on FFT level) the same but sounds quite different, the property is therefore investigated.
- **Linear Speed Change** of +1%,-1%, +4% and -4%. It is the same as **Time Scale Modification** except that both pitch and tempo are changed. The degradation was added with MAGIX Audio Cleaning lab 16 deluxe [41]. This degradation changes the data in such a way that it sounds similar, but is mathematically more different than the **Time Scale Modification**. This property evaluates how invariant the algorithm is.
- **Resampling** consisting of subsequently down- and upsampling to 22.05 kHz and 44.1 kHz respectively uses an AC₃Filter [40]. The resampling should have little or no effect as the audio are downsampled in the algorithm. The property investigates how well the algorithm handles artefacts.
- **White Gaussian noise** is added to the audio segment to test robustness, ranging from 15 dB to -15 dB in intervals of 3 dB. This test will be applied to different lengths of unidentified audio segments to determine how much of the unidentified audio segment is needed, the same test was done in Wang's paper [1].

In this chapter, the database setup and comparison between the different algorithms were discussed. From this it was concluded that a generic algorithm based on that of Avery Wang's algorithm should be implemented for advertisement identification. Further, the environments used for the implementation were motivated, followed by the experimental setup for validation and verification purposes. Now the researcher will proceed to the implementation of the algorithm.

Chapter 6 – Implementation

In this chapter the implementation of the fingerprinting algorithm is presented. Considering the desired interface and solution architecture, an implementation of the functions is delivered in this section.

Before advertisements can be detected, an initialisation phase is executed. In the initialisation phase, the application calculates filters coefficients. Section 3.1.2 **Filters** discusses the configurations and calculations needed. The application then checks to see whether a lookup table and hash table exist. If the lookup (containing advertisement descriptions) and hash table (database with landmarks) do exist, the advertisement ID is set to the next available index in the lookup table and the hash table and lookup table are read into the RAM. A new empty hash table and a new empty lookup table are created if neither exists.

A brief overview of the implemented audio fingerprinting algorithm is presented in **Figure 19**. A button is represented by a **B** abbreviation and the abbreviation **F** represents the implementation of the functions of the algorithm.

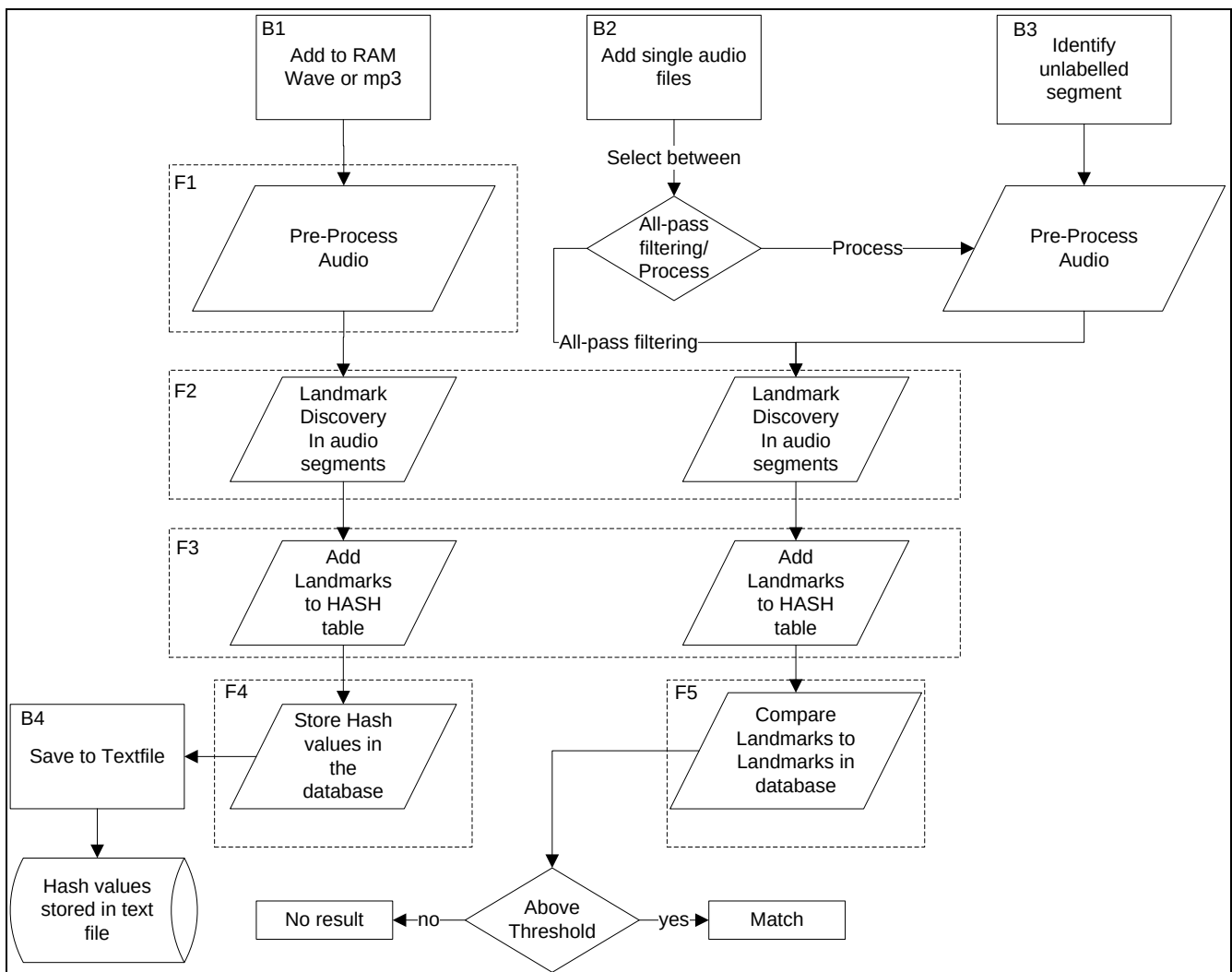


Figure 19 : Implementation

6.1 Functionality 1: Pre-processing audio

The first step is to determine whether the audio data is mono, if not it is converted to mono (see **Chapter 3 – Technical Aspects Related to Audio Fingerprinting** for calculations and refer to **Figure 20**). The mono data is then filtered with a lowpass FIR filter (8 820 Hz cutoff frequency) using the Kaiser window design method. For the Kaiser Window coefficients see **Addendum**. Finally, the data is downsampled to 8 820 Hz preferably with an integer divider.

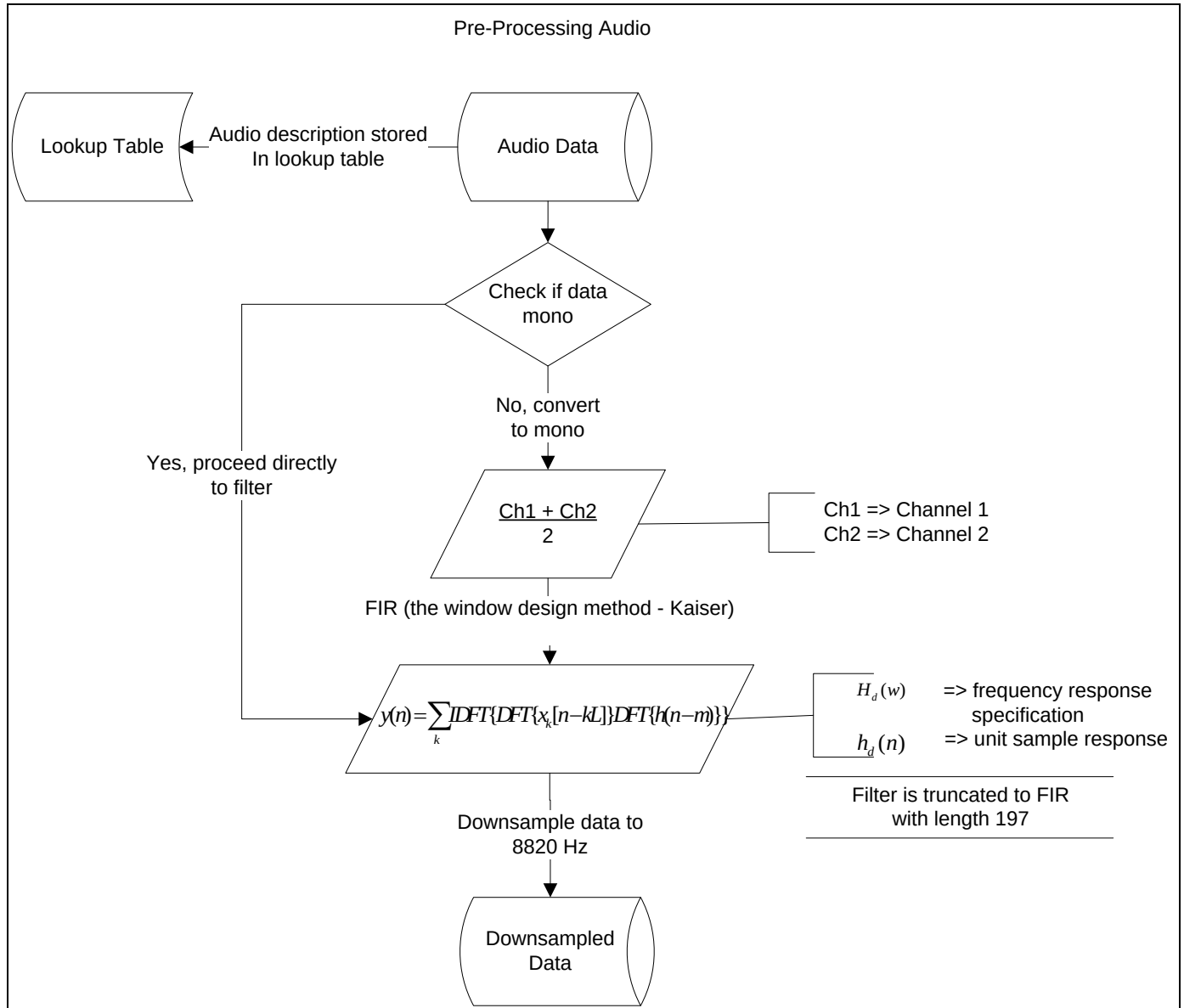


Figure 20 : Pre-processing Audio

The downsampled data along with the audio ID (the number of corresponding references in the lookup table) is then transferred to the landmark discovery function where it will be processed further to identify its landmarks.

6.2 Functionality 2: Landmark discovery

From the downsampled audio segments the software identifies prominent frequency peaks, also referred to as maximums peaks. The prominent frequency peaks are paired with others known as landmarks and mathematically formulated to hash values.

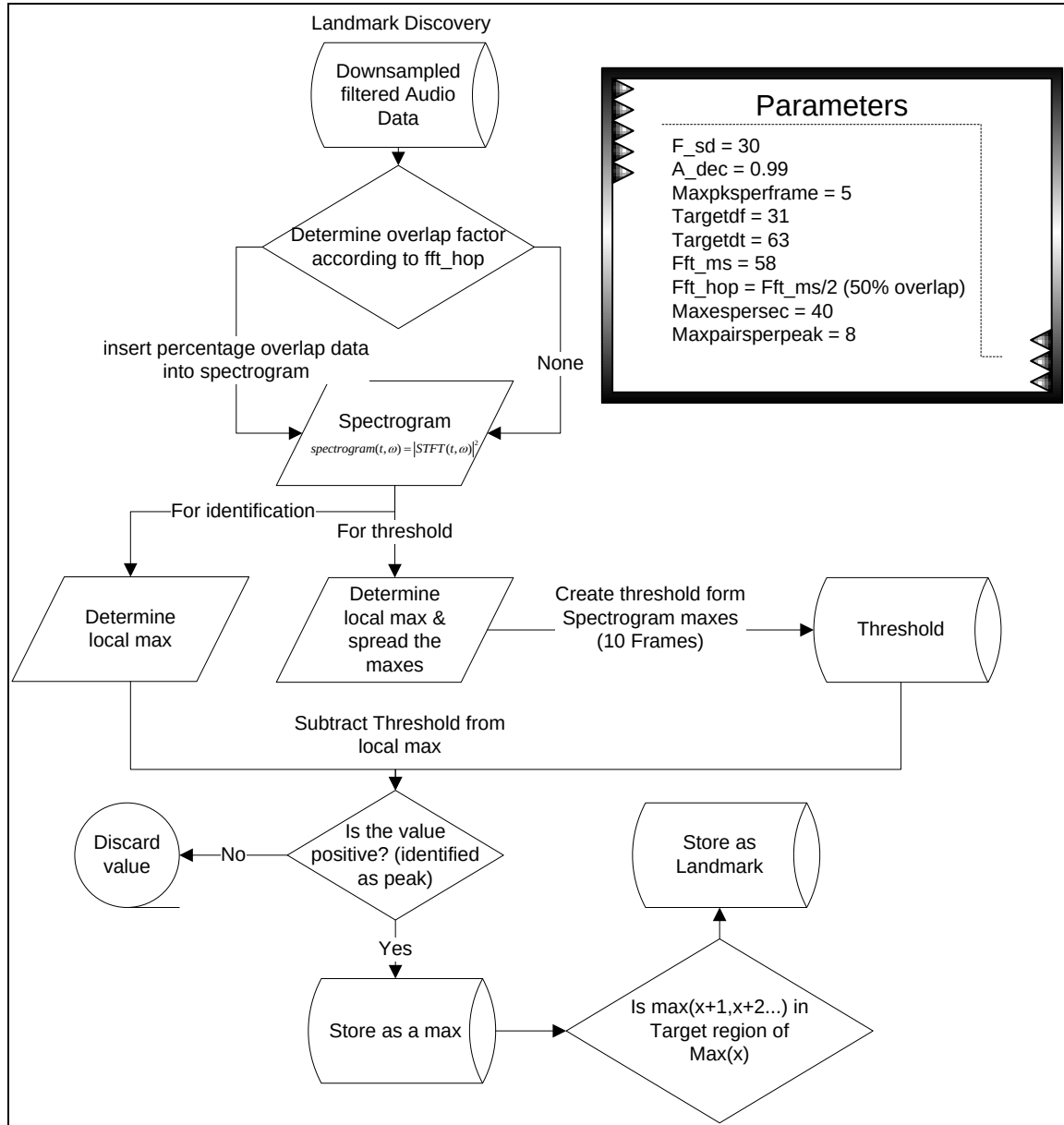


Figure 21 : Finding Landmarks

Before further examining the flow diagram in **Figure 21**, a brief description of the parameters will be presented as well their effect on the number of landmarks. To see the motivation for the selection on the parameters refer to **5.1 Database setup and comparison**. The arguments for their values are justified through the requirement for advertisement identification.

6.2.1 Parameters

As described in the above section, there are parameters that have an effect on the robustness, accuracy, speed and the database size. In **Table 4** the different parameters are briefly described.

Table 4 : Parameters

Parameter	Description
F_sd	Frequency standard deviation. Larger value results in fewer landmarks.
A_dec	Decay rate. Value closer to one result in fewer landmarks. The decay decreases the threshold to identify peaks. This is a counter measure to finding duplicate landmarks.
Maxpkperframe	Maximum prominent frequency peaks per frame. Higher value results in more landmarks to a certain extent due to the practical limitations of finding more than 10 max peaks per frame.
Targetdf	Target distance frequency. Larger value results in more landmarks, but is limited by the database size. Increasing the range of the frequency distance between peaks allows more landmarks to be formed. Further, allowing targetdf a larger range than -32...31 will result in a value greater than 6 bit, which will change the database size.
Targetdt	Target distance time. Larger value results in more landmarks, but is limited by database size. For the same reasons as targetdf allowing the range larger than 0...63 will increase the target region, but change the database size.
Fft_ms	FFT frame size in time domain. Larger value results in larger a database. When the frame size is increased, it increases the target region and becomes less accurate as targetdt will receive duplicate landmarks.
Fft_hop	FFT hop is the distance before the next frame start. Thus the optimal overlap will always be $\text{fft_hop} = \text{fft_ms} / 2$ (50% of fft_hop) resulting in the maximum possible landmark gain by the fft_hop parameter.
Maxespersec	Maximum prominent frequency peaks allowed per second. Limit landmarks in extreme cases where there would be a large amount of landmarks per second. This parameter also allows for a safe database estimate.
Maxpairsperpeak	A maximum of pairs per anchor frequency peak. Larger value results in more landmarks. The parameter allows the primary peak to form more landmarks with other peaks.

6.2.1.1 Frequency standard deviation

The frequency standard deviation (f_{sd}) is selected to be 30, to coincide with our landmark requirement, refer to **Figure 22** . The frequency standard deviation is the standard deviation (in frequency bins) with a Gaussian distribution [42]-[44]. This distribution forms the threshold for identifying the prominent frequency peaks described in section 6.2.3 **Local max and spreading**. A larger value means fewer peaks were found, it gives a frequency peak just too much weight for other peaks to be above the threshold.

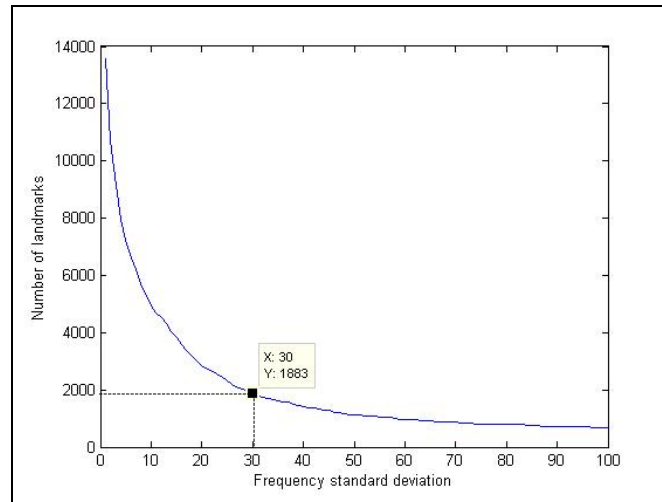


Figure 22 : Number of landmarks vs. frequency standard deviation

6.2.1.2 Decay rate

The decay rate (a_{dec}) decays the updated threshold, discussed in section 6.2.3 **Local max and spreading**. The threshold is updated with every frame to eliminate the duplication of prominent frequency peaks in the consecutive frames. There is a small chance that a unique pair of frequency peaks could be eliminated with the updated threshold, therefore the threshold is decayed with a factor of 0.99 to correspond with our landmark requirements (see **Figure 23**).

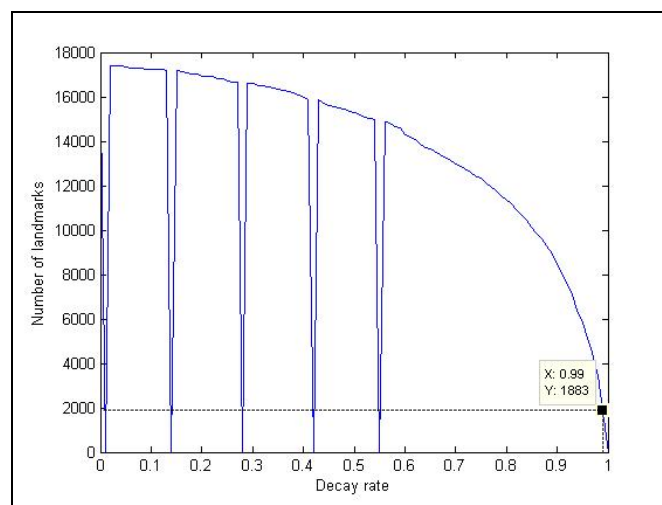


Figure 23 : Number of landmarks vs. decay rate

6.2.1.3 Maximum prominent frequency peaks per frame

The maximum amount of prominent frequency peaks per frame (maxpkperframe) seldom reaches more than 5. Limiting the maximum number of peaks per frame results in an exchange for a more even spread of peaks per frame vs. a really unique set of landmarks. The maximum peaks are set at 8 to allow a really unique set of landmarks. This allows for more landmarks to be formed, this could result in a case where all the landmark space is used in the first second, to eliminate this problem, the researchers used the maximum prominent frequency peaks per second. With 17 frames in a second, a maximum of 85 peaks can be generated, but is limited to 40 to correspond with our landmark requirement.

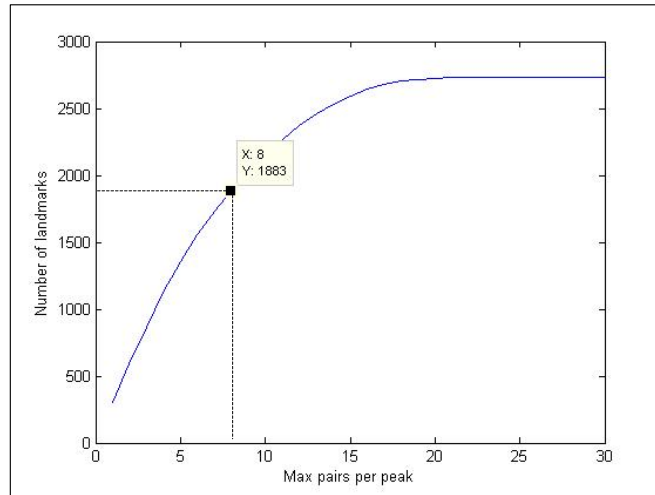


Figure 24 : Number of landmarks vs. maximum number of peaks per frame

6.2.1.4 Target distance frequency

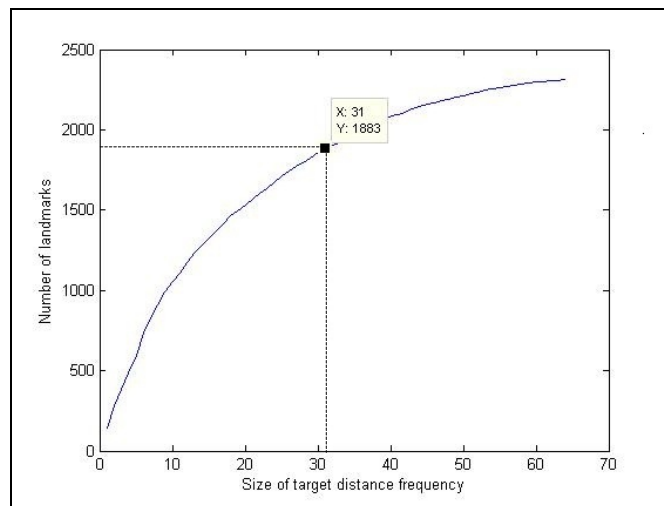


Figure 25 : Number of landmarks vs. target distance frequency

Target distance frequency (targetdf) with target distance time (targetdt) determines the target region. The larger the target region, the more pairs for the anchor frequency peak, hence more landmarks. The target distance frequency is however limited to 6 bits for this application. Increasing the range above 6 bits will result in a larger database. The target distance is thus limited to 31 bins for it ranges from -32 to 31 (6 bits) (see **Figure 25**).

6.2.1.5 Target distance time

Refer to **6.2.1.4 Target distance frequency**, for the way in which the target region is described. The main difference between target distance time and target distance frequency is that the target distance time is handled in the time domain and always progresses in a positive direction, whereas target distance frequency is handled in the frequency domain and can be positive or negative. In **Figure 26** it can be seen that the Target distance time's bins are 63 (2^6 meaning we have 6 bits to describe the time domain).

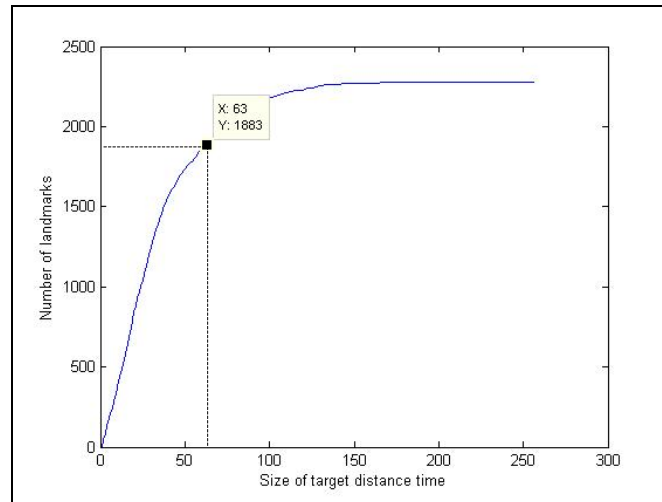


Figure 26 : Number of landmarks vs. target distance time

In **Figure 27** the calculated target region can be seen as described by target distance time and frequency. The pair peaks are located inside the target region with which the anchor peak can pair to form a landmark.

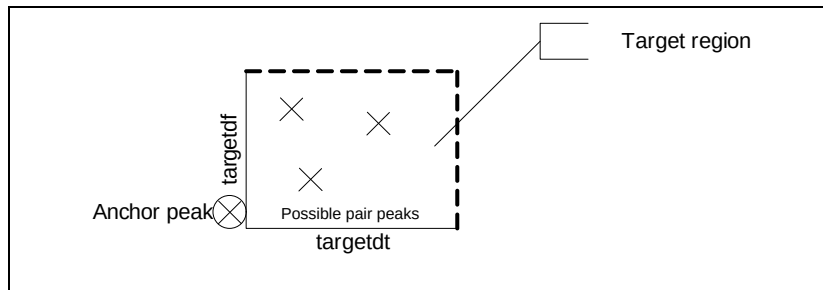


Figure 27 : Target region

6.2.1.6 FFT frame size in the time domain

The FFT frame size is 512. Translating this into the time domain means that the frame (fft_ms) is 58ms in length ($512/8820$ Hz). The audio is downsampled from its 44.1 kHz sample frequency to an 8820 Hz sample frequency resulting in an integer divider and faster downsampling time. The FFT data contains mirrored data with its DC (Direct current) component [20] so only half is necessary (256 points). The 256 point FFT frame results in 8 bits being used in the database for describing the starting time of a landmark. Changing the frame size will result in a larger or smaller database, but for this application changing the frame size is not allowed as desired database size is already acquired (see section **5.1 Database setup and comparison**).

6.2.1.7 FFT frame hop distance

For this application purposes the frame hop (fft_hop) distance is equal to that of the FFT frame. The frame hop distance is the distance between two frames. Values higher than the FFT frame results in discarding of data (not advised) whereas smaller values result in an overlap. This application has a 50% overlap (**Figure 28**), which helps, when an unknown audio segment input is made, to combat against frames going out of sync. See section **6.5 Functionality 5: Compare landmarks** for a description.

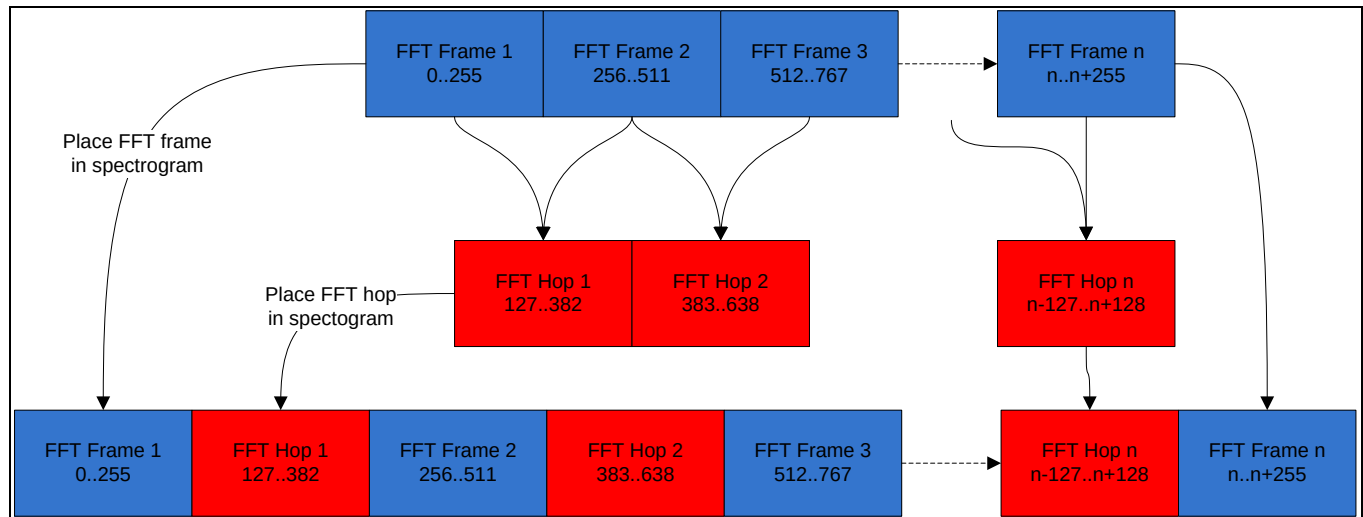


Figure 28 : Spectrogram with 50% overlap

6.2.1.8 Maximum prominent frequency peaks per second

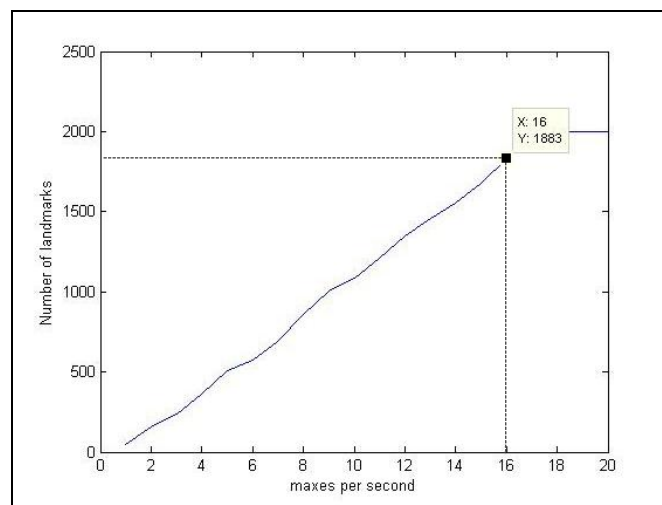


Figure 29 : Number of landmarks vs. maximum peaks per second

Maximum prominent frequency peaks per second (maxespersec) is the maximum number of peaks allowed per second, refer to **5.1 Database setup and comparison** for the motivation of setting the maximum to 20 per second. As seen in **Figure 29** the parameter has a linear effect to an extent. The reason for this is that the parameter is there for precautionary purposes. The probability of the parameter not having an effect is 10% (18 out of 20 as an effect as seen in **Figure 29**).

6.2.1.9 *Maximum number of pairs per anchor peak*

The maximum number of pairs per anchor peak (maxpairsperpeak) limits the peak to form higher amount of pairs than specified allowing either an even spread of peaks or the discovery of truly unique pairs of peaks. As seen in **Figure 30**, the highest the number of pairs can reach before exceeding the maximum landmarks as defined for this application is 8. Refer to section **6.2.1.3 Maximum prominent frequency peaks per frame** for motivation for the number of pairs being 8. The other parameters constraints start having an effect round about 15.

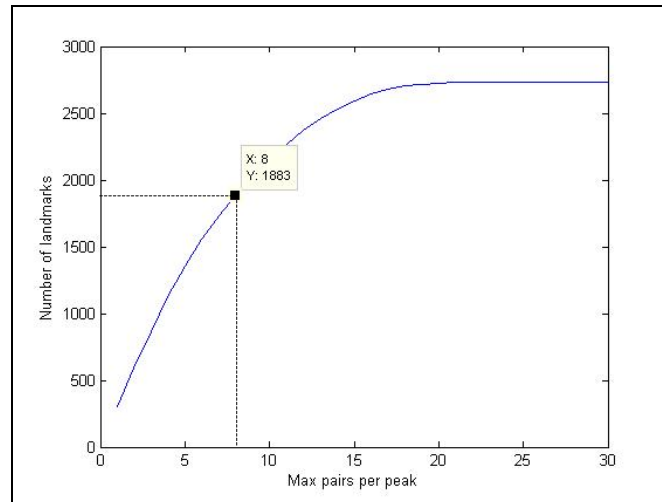


Figure 30 : Number of landmarks vs. maximum pairs per peak

6.2.2 Spectrogram

A spectrogram is known as the energy of the signal, refer to **equation 18**. As discussed in **6.2.1.7 FFT frame hop distance**, the spectrogram code has the ability to perform an overlap to combat out of sync frames. See how the overlap is implemented in **Figure 31**.

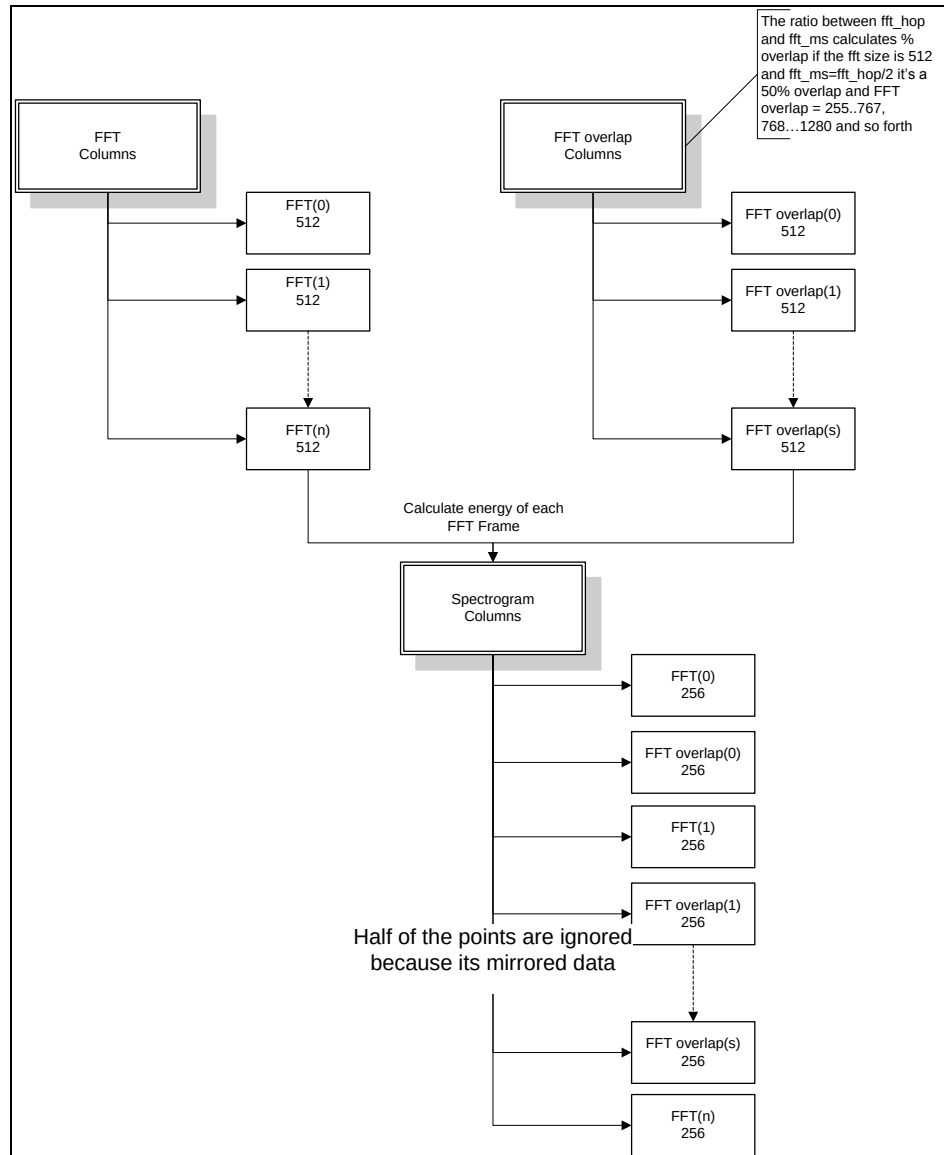


Figure 31 : Spectrogram architecture

The next step is to emphasise the peaks through converting the spectrogram data to the log domain. It is important that the log values should not be negative as this decays the threshold in a harsh manner. The negative values are replaced with an arbitrarily small number. To combat different compressions (noise, low quality mp3 etc.) the average of the spectrogram must be subtracted from the spectrogram data. Note that no negative values are allowed at this point (negative values are substituted with 0.000001). This is followed by discovering the local maximum frequency peaks and spreading them to create the thresholds.

6.2.3 Local max and spreading

A local maximum frequency peak is created by a linear search at the previous and next value. If the centre value is higher than both the previous and next value, it is a local maximum.

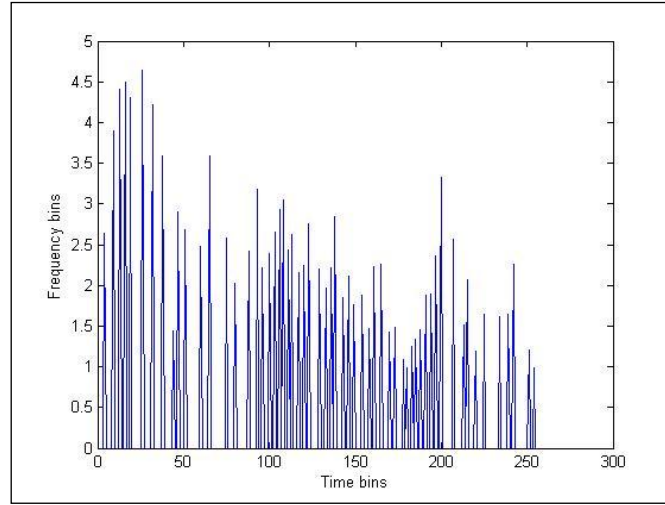


Figure 32 : Local max

As seen in **Figure 32**, the local maximums identified will have trouble to create a threshold. If only local maximum values are used, there will be large gaping holes. Too many peaks will be identified; therefore, it is smoothed through a Gaussian half-width filter that is used in the spread function. The length of the kernel (coefficients) of the Gaussian filter needs to be at least $6\sigma - 1$, and is recommended for a 0.1% error as $8\sigma + 1$ [42], [43], [44].

The Gaussian filter's equation is given by:

$$G(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{[n]^2}{2\sigma^2}} \quad \text{Where } [n] \text{ is the number of coefficients (28) and } \sigma \text{ the standard deviation}$$

As the aim with the Gaussian filter is to smooth each of the maximum values, only one convolution is necessary per maximum value. As it is, the maximum value's weight is spread, the Gaussian filter's centre is placed on this value and then the weight is spread by the following coefficients

$$G(m) = e^{-\frac{[n]^2}{2\sigma^2}} \quad (29)$$

thus spreading the weight is

$$y[\text{max value}] = x[\text{max value}] \times G\left[\frac{m}{2}\right], -\frac{n}{2} \leq m \leq \frac{n}{2}, \frac{m}{2} - \frac{n}{2} \leq \text{max value} \leq \frac{m}{2} + \frac{n}{2} \quad (30)$$

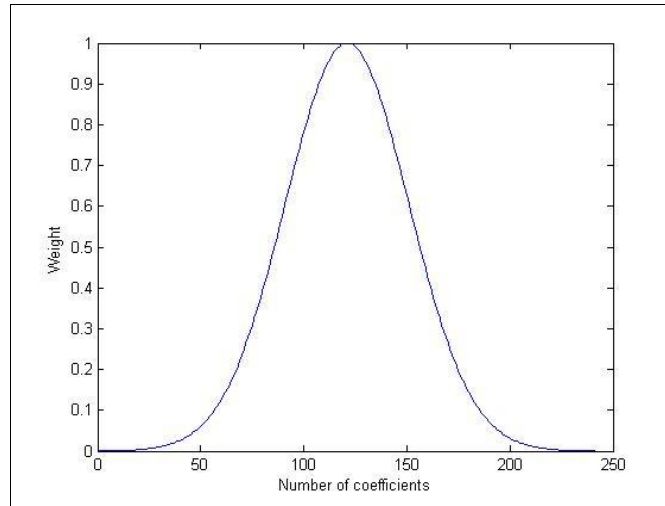


Figure 33 : Gaussian coefficients

The Gaussian curve is 241 bins in length. The Gaussian curve length is calculated by the frequency standard deviation (f_{sd}) 4×times - (0.1% of the weight). This curve's centre is always placed on the local max. See **Figure 34** for the effect of one local maximum spread.

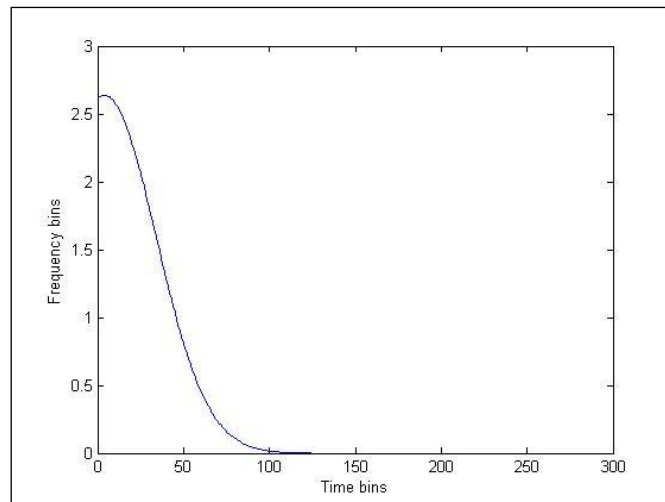


Figure 34 : Local maximum spread of one local maximum

All the highest values of each local maximum's spread are used to create a threshold (see **Figure 35**). To identify the prominent frequency peaks the following steps are followed:

- I. A threshold is computed of the first 10 frames, as presented in **Figure 35**.
- II. Create local maximums of every frame as in **Figure 32**.
- III. Local maximums above the threshold in point I, is considered a prominent frequency peak. To compute the threshold, the Gaussian curve is applied to every local maximum peak. This spreads the weight and delivers **Figure 34**, which is referred to as the spread. To get a better threshold this calculation is performed for 10 frames. The maximum values of all these spreads are used as the threshold (refer to **Figure 35**).

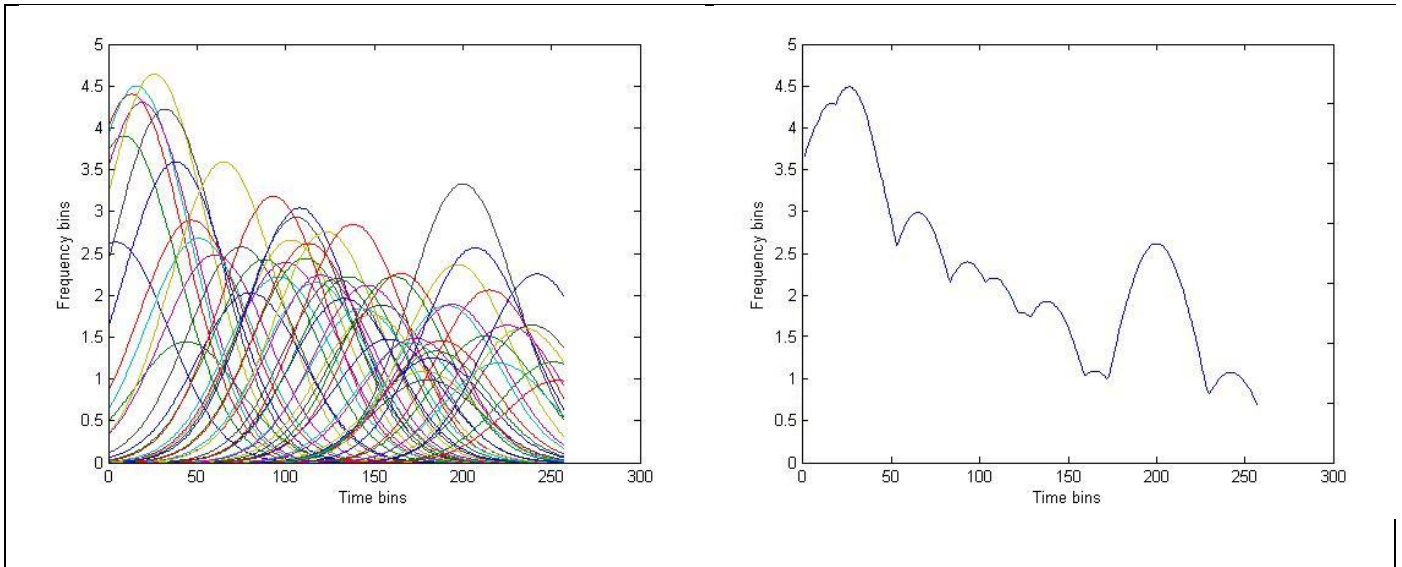


Figure 35 : Creating the threshold

When the peaks are above the created threshold, they are identified as prominent frequency peaks. When creating landmarks, the first prominent peak is selected as the anchor peak and if the other prominent peaks are in the defined target region they are selected as pairing peaks. The anchor peak is paired with a pairing peak, considered as a landmark, where the limitations on the maximum pair of peaks are constrained by the maximum number of pairs per anchor peak (maxpairsperpeak).

The landmark is created by using the anchor peak's and pairing peak's time and frequency (refer to **Figure 15**). After every time a prominent frequency maximum is found, the consecutive frames' threshold gets updated by spreading that prominent frequency peak and adding it to the threshold. This is done to eliminate recurring landmarks. The decay rate (a_{dec}) decreases the threshold with the configured ratio to allow more landmarks. All the peaks are stored and subjected to another round of updating and decaying, but in reverse order, to make certain all peaks are above the final threshold.

The ID (linked with the metadata in the lookup table) of the advertisement and starting time of the landmark is stored in the hash table. The hash value is calculated with the following variables in the starting frequency: frequency and time differences (see section **6.3 Functionality 3: Landmarks to hash**).

6.3 Functionality 3: Landmarks to hash

The calculation of the hash value is important, as it must be designed and calculated in such way that each landmark must have a unique hash value. The smallest base value to space the three variables is 2^6 , which will be referred to as b . This allows even spacing between the variables ($F1, \Delta F, \Delta t$). $\Delta F + \Delta t$ is not allowed to compute to the same hash value, so it is necessary to space ΔF , $(\Delta t_{max} + 1)$ times, giving

$$Hash = \Delta F \times b + \Delta t \quad (31)$$

No formulation of $\Delta F + \Delta t$ can have the same hash value except if both variables equals 0. The same applies to $F1$, where no formulation of $F1 + \Delta F \times b + \Delta t$ are allowed to compute to the same hash value. $F1$ has to be spaced $\Delta F_{max} \times b + \Delta t_{max} + 1$ times, which equals 4 033. $F1$ is thus spaced b^2 times as the next power of two values is 4 096, giving

$$Hash = F1 \times b^2 + \Delta F \times b + \Delta t \quad (32)$$

With this simple equation, no value will overlap, but the maximum value is now larger than the database. This problem is solved through defaulting any hash value above 2^{20} (database size) to 2^{20} . The reason for this is that we lose only 4 160 possible unique hashes. As these hashes have a very low probably to be calculated in any way, it is deemed better than raising the database size to 2^{21} (which leave almost double unoccupied space) or adding the 4 160 to the databases' size (this will only cause the database to require more reads on a low level basis and is unnecessary). In **Figure 36** a visual demonstration of hash calculation can be seen.

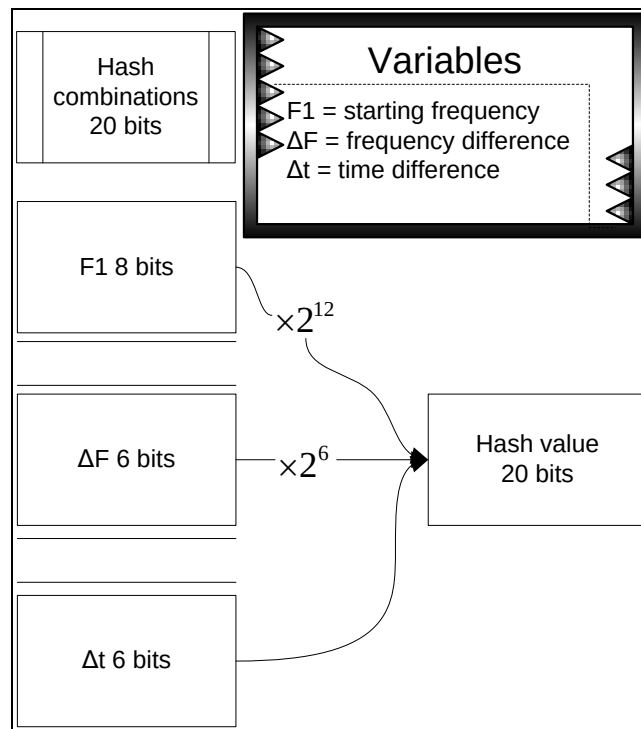


Figure 36 : Hash calculation

Refer to **Figure 37** for a demonstration of how the hash value is calculated.

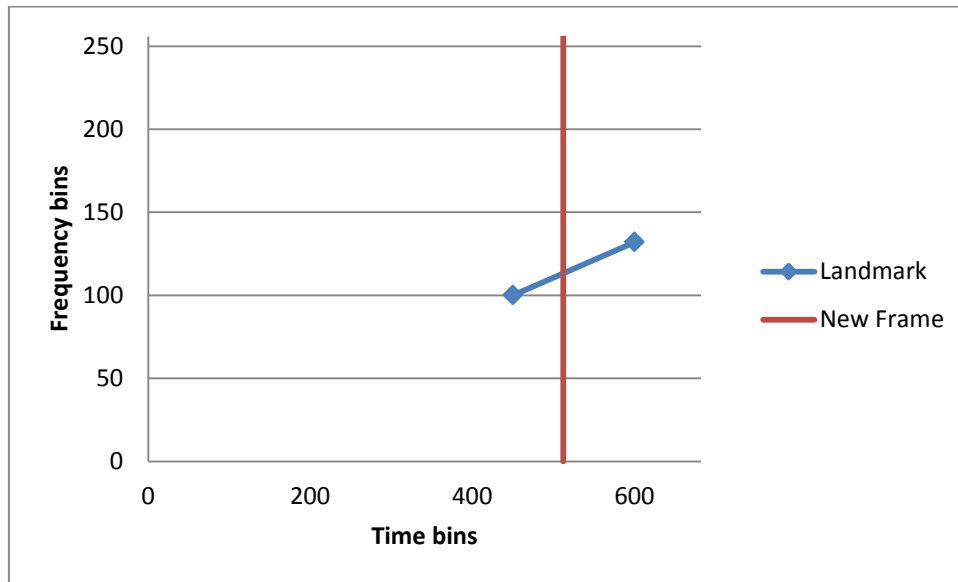


Figure 37 : Hash calculation example

In this example, frame 0 and 1 are used. The starting time(t) is 0 for it starts in frame 0. This does not form part of the hash value and is stored along with the song ID at position hash value in the database.

The two prominent frequency peaks have the following coordinates in terms of bins: anchor peak's coordinates 450,100 (12.6ms, 861 Hz) and the pairing peak's 600,132 (16.8ms, 1 137 Hz) resulting in a starting frequency (F_1) of a 100 bins (861Hz), frequency difference (ΔF) of 32 bins (276 Hz) and a time difference (Δt) of 1 frame (58ms). The time difference (Δt) is very sensitive and is therefore calculated by subtracting the difference in frames.

This gives us a hash value of 411 649, meaning at position 411 649 in the hash table the song reference and starting time will be stored.

6.4 Functionality 4: Save hash

After calculating the hash value, the application uses the hash as an index where the advertisement ID and starting time (number of frame) is stored. **Table 5** shows the hash value (index), advertisement IDs (Lookup table's index) and starting time (number of frame). A maximum number of 24 entries can be stored in one hash value. This allows for more advertisements to be added, but only to a certain extent. If handling music or higher frequency data it would be wiser to increase the database size to 2^{32} or use larger FFTs, as this will still have more or less the same accuracy and robustness, but will take more time. As one of the advertisement requirements is speed, this method was not chosen.

Table 5 : Representation of database

Index	Information
Hash value	{Advertisement ID, Starting time} $\times$ 24

If more than the maximum number of entered hashes is discovered at a hash index, they are simply discarded as this has a high probability to be a common hash, which all advertisements contain. Refer to **Figure 38** for a visual representation of the database.



Figure 38 : Database structure

6.5 *Functionality 5: Compare landmarks*

When an unlabelled recording needs to be identified, it follows the same procedure than when adding a recording to the database. However, the data is duplicated and runs through twice to discover landmarks, one with the data unchanged and the other with a 50% offset as seen in **Figure 39**.

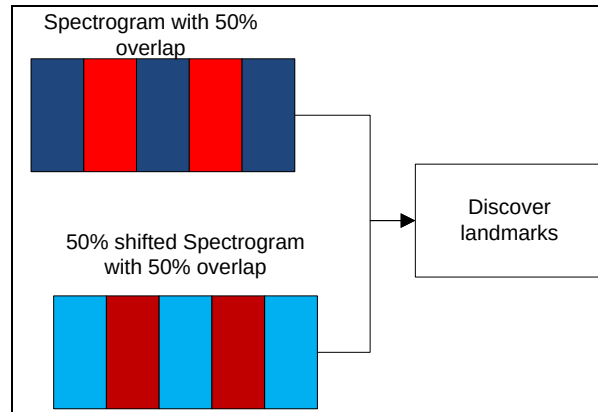


Figure 39 : Discover landmarks for unknown label

The 50% offset solves the out-of-sync frames problem. If the unknown label had an offset of 50% in comparison to the original sample, then it would miss most landmarks with time differences that are not calculated correctly. After the landmarks are discovered and the hashes calculated, it is time to compare it to that of the database. One of the advantages of a hash table, as seen in **Figure 40**, is that it does not have to search the entire database, as only the identical hash values are retrieved.

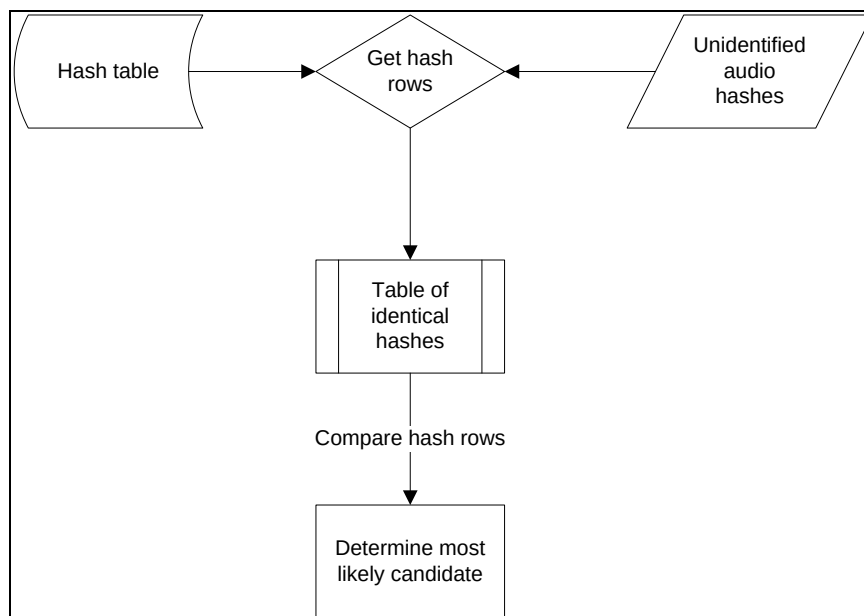


Figure 40 : Compare hashes

As each hash value (database index) has a maximum number of entries (maxentries), a table is created with the matching hashes. A search and compare algorithm is used to determine the most likely candidate.

First, the advertisement ID that is most prevalent (number of times it exists in this table which corresponds to hashes calculated for unidentified audio sample, is referred to as hits) is determined, after that the 2nd and 3rd most

prevalent advertisements are determined. Then the most prevalent starting time is determined for the prevalent advertisements. If the total advertisement ID with the most hits is more than our determined threshold (which is 20 for a 3 sec unidentified audio), then the unidentified audio is identified to the corresponding advertisement ID. The advertisement ID is used as an index in the lookup table and the metadata is recovered and displayed, then the prevalent starting time is considered as the moment where the fingerprint existed in the original audio.

From time to time there will be a 2nd or/and 3rd identified sample, also with enough hits to pass the threshold. In that case it is very likely that the same fingerprint exists in more than one advertisement, e.g. if an advertisement used a popular song in the advert and both are in the database both will be identified. For experimental reasons 3 different audio segments were added on top of each other and all three were identified. The other reasons for the 2nd and 3rd segments to be prominent in the compare of landmarks, are to determine what the highest false positive value is. This threshold is also used to differentiate between false positives and matches.

6.6 GUI buttons

The buttons in **Table 6** describe the user side of the functionalities.

Table 6 : buttons

Buttons	Description
B1 Add to RAM wav or mp3	Add to Random access memory (RAM) Wav or mp3: add the advertisements to the ram
B2 Add single audio files	Add unidentified advertisement with any length for experiments
B3 Identify unlabelled	Identify, identifies all advertisements in special renamed 5 min signals
B4 Save to Textfile	Save to database, stores value in ram to hard disk

6.6.1 Button 1: Add to RAM wav or mp3

When the user presses button 1, a dialog box will open for the user to select the audio he/she wishes to add to the database to be identified. After the audio has been selected, it is processed through functionality 1, 2, 3 and 4. The fingerprints are currently only stored in the RAM. The audio which has been added is displayed in a list box in the upper left corner (refer to **Figure 18**).

6.6.2 Button 2: Add single audio files

Practically, this button is actually three buttons, but describing each on its own would be redundant. When the user clicks on the *add* button, a dialog box opens and the user can select the audio it wants to be identified. The unlabelled data is now shown in a list box in the bottom left corner. The user then selects an unlabelled audio in the list box and chooses between different processes and all-pass filtering (discussed in **5.3.2 Tests**).

The unlabelled data is processed through functionalities 1, 2, 3 and 5. The results are then displayed in three bars in the bottom right corner. The first bar shows the matched audio's description, where the landmarks were situated in the audio, as well as the number of landmarks identified. The other two bars display the same information but for the nearest false positives. With this information, the threshold of the number of landmarks identified can be calculated. This button's purpose is purely experimental and for debugging purposes.

6.6.3 Button 3: Identify unlabelled segment

Button 3 is specifically for the advertisement purposes. The user adds any number of 5 min segments of radio signal named in a special format *hour_minutes_seconds.mp3/wav*. This allows the exact time and date of when the advertisement was broadcasted to be identified. The data follows the same functionalities as button 2, but the 5 min segment has a reiterated process. A 3-second segment is used every 15 seconds in the 5 min segment, and the 3-second segment is processed through functionalities 2, 3 and 5. The results are displayed in a list box in the middle with time and number of landmarks identified.

6.6.4 Button 4: Save to Textfile

All the data displayed in the upper left corner is still only in the RAM. If the data is satisfactory, button 4 saves the hash table and lookup table to a text file. The next time the application is started, it will load the stored data. A text file was used for debugging purposes. It would be better if a more suitable database was used, but that is not in the scope of this dissertation.

In this chapter, a detailed implementation of the generic algorithm was presented. The various parameters and their effects were discussed and motivations for their values were presented. All the functionalities were explained on a step by step basis, as well as the way in which the algorithm is used for advertisement identification purposes. Finally, the buttons in die GUI were explained from a user point of view. The generic algorithm is implemented and the researcher has the necessary experiments for validating and verifying the algorithm. The next step is to analyse the results to verify and validate the generic algorithm.

Chapter 7 – Results

In this chapter, following the experimental setup, test and implementation, the results can now be discussed. The aim of this section is to explain the results and to verify and validate the algorithm.

The application is developed in VB.NET (Visual basic .Net framework) to have the same speed advantages as other applications, which are not produced in a prototype environment (Matlab). Haitsma and Kalker's algorithm was programmed by a colleague of the TeleNet research group, Professor Willem Venter. Verification and validation techniques are based on the paper of Robert G. Sargent [45]. For the verification process, the different functionalities of the application are to be reviewed. For the validation process, three techniques, namely comparison to other models, face validity and predictive validation will be used.

7.1 Verification and validation

The verification model for a software product is defined as “ensuring that the computer program of the computerised model and its implementation are correct” [45]. The validation model of a software product is defined as “substantiation that a computerised model within its domain of applicability possesses a satisfactory range of accuracy consistent with the intended application model” [45].

7.1.1 Verification

In the verification process, we will prove that the functionalities of the application were successfully implemented. The functionalities form part of the model or in this case of the algorithm, the functionalities required by the generic algorithm are as described in **Chapter 6 – Implementation**. The verification of the functionalities is as follows:

- I. Functionality 1: Pre-processing of audio. Converting the audio to mono requires the average of two channels. The audio was then filtered with the window method using a Kaiser window (refer to section **3.1.2 Filters** and **3.1.3 Applying the filter coefficients**). In playing the audio back, it was clear when listening to the audio that the audio segment has been converted to mono and filtered. This was followed by the downsampling of the audio segment, the audio segment decreased to a 5th of its size and when played back, it sounded the same than after the filter had been applied.
- II. Functionality 2: Landmark discovery. Local maximum frequency peaks were discovered by looking at the values beside the peak in question. These were evenly spread, which created the threshold (refer to **Figure 35**), to determine prominent frequency peaks. Prominent frequency peaks within each other's target range were converted to landmarks. The correct procedure and implementation were followed as unique landmarks were generated and successful results were obtained (for the results refer to section **7.2 Advertisement identification**).

- III. Functionality 3: Landmark to hash. The landmark's value consists of 20 bits, 8 for the starting frequency $F1$, 6 for the frequency difference ΔF and 6 for the time difference Δt . Refer to section 6.3 **Functionality 3: Landmarks to hash** to see that no combination of the 3 variables in question can deliver an identical hash value.
- IV. Functionality 4: Save to hash. **Figure 38** clearly demonstrates how the database is spaced, allowing functionality 5 to match the required hash values instead of comparing the entire database.
- V. Functionality 5: Compare landmarks. As the functionality consists of all the other functionalities it is also verified. Successful results were obtained, thus the compared functionality was successfully implemented.

The above functionalities and how they were implemented were explained mathematically. The results that were obtained satisfied Robert G. Sargent's [45] requirements to ensure that the model and implementations are correct. This being said, the algorithm needs to be validated as an audio fingerprinting algorithm and more specifically a robust landmark audio fingerprinting algorithm.

7.1.2 Validation

For the validation of the two algorithms, three techniques described by Robert G. Sargent [45] are used. First, the algorithm is compared to that of Haitsma and Kalker's algorithm to validate that it is an audio fingerprinting algorithm and then to Avery Wang's algorithm to validate that it is a robust landmark audio fingerprinting algorithm. Next, face validity followed in which it is decided whether the behaviour of the algorithm is logical and reasonable. This was followed by a predictive validation where predicted behaviour is compared to that of the actual behaviour.

7.1.2.1 Comparison to other models

The algorithm was executed on a computer with an Intel® Core™2 Duo CPU @ 2.10 GHz with 4 GB of memory. The same audio segment is used as in the Haitsma and Kalker paper with an additional advertisement. The audio segment is sampled in stereo at 44.1 kHz in 16-bit format. The same songs used in Haitsma and Kalker's paper [2] and a radio advertisement are used. The audio extracts and the advertisement are 30 seconds in length for the tests described in 5.3.2 Tests. The songs are from four different musical genres:

- "Fortuna" by Carl Orff, referred to as Orff
- "Success has made a failure of our home" by Sinead O'Connor, referred to as Sinead
- "Say what you want" by Texas, referred to as Texas
- "A whole lot of Rosie" by AC/DC, referred to as AC/DC and
- An advertisement of CPE Debt Councillors referred to as Advert

The error rating in **Table 7** is defined as the difference between the correct landmarks found and the amount of false positive landmarks found divided by the maximum landmarks generated per audio

$$Error\ rate = \frac{Landmarks - falsepositive}{Total\ landmarks} \quad (31)$$

The abovementioned audio segments are subjected to the following degradations, as shown in **Table 7**.

Table 7 : Error rating for different kinds of signal degradations

Processing	AC/DC	Advert	Orff	Sinead	Texas
Mp3@128Kbps	0.006	0.012	0.003	0.003	0.010
Mp3@32Kbps	0.031	0.013	0.009	0.006	0.026
All-Pass filtering	0.018	0.023	0.005	0.005	0.028
Equalisation	0.014	0.003	0.005	0.001	0.031
Band Pass filter	0.002	0.001	0.000	0.002	0.010
Time scale +4%	0.126	0.454	0.255	0.122	0.633
Time scale -4%	0.242	0.315	0.254	0.082	0.475
Linear Speed +1%	0.531	0.233	0.187	0.167	0.371
Linear Speed -1%	0.292	0.293	0.135	0.153	0.449
Linear Speed +4%	0.326	0.435	DNF	0.437	0.735
Linear Speed -4%	DNF	0.829	0.658	DNF	DNF
Resampling	0.000	0.000	0.000	0.000	0.000

DNF = did not find

For the results of Haitsma and Kalker's paper [2] refer to **Addendum**. Haitsma used the BER (bit error rate) of the audio fingerprints. It is the percentage of bits that did not correspond with the original fingerprint. Haitsma's research delivered that 35% and lower meant that the audio will still be identified. In **Table 7**, the nearest false positive is so far off that it always finds its match unless described as DNF (did not find).

When the results of **Table 7** and **Table 11** are compared as seen as in **Figure 41**, it can be concluded that the results are almost similar, which, validates the algorithm, because the algorithm does what it is designed for. For advertisement purposes it is assumed that the most probable signal degradations will be compression, noise (slight), equalisation and time scale (within 6%), making the generic algorithm a better solution for advertisement identification.

There is little information available on the exact implementation of the Shazam algorithm or the type of computer it was implemented on. Therefore, the only verification method is to compare the generic algorithm results to that of Avery Wang's. White Gaussian noise [37] is added to the same audio as specified with Haitsma and Kalker's algorithm (see **Table 8**).

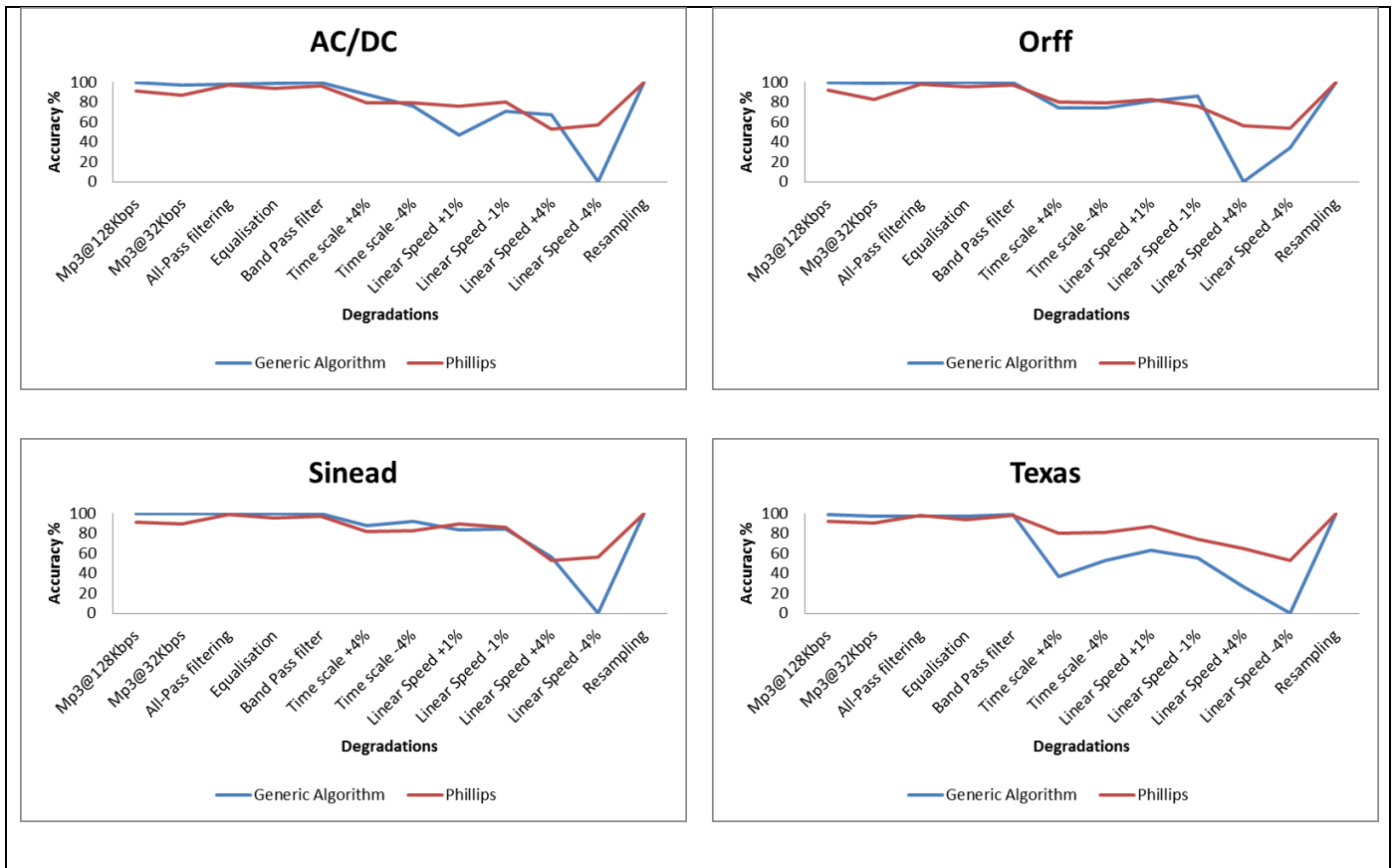


Figure 41 : Generic vs. Phillips

Table 8 : Gaussian white noise addition

Processing	AC/DC	Advert	Orff	Sinead	Texas
(S/N) 15 dB	✓ (517)	✓ (962)	✓ (799)	✓ (1010)	✓ (831)
(S/N) 12 dB	✓ (524)	✓ (908)	✓ (745)	✓ (1002)	✓ (847)
(S/N) 9 dB	✓ (527)	✓ (792)	✓ (682)	✓ (903)	✓ (830)
(S/N) 6 dB	✓ (474)	✓ (680)	✓ (556)	✓ (800)	✓ (795)
(S/N) 3 dB	✓ (400)	✓ (584)	✓ (474)	✓ (671)	✓ (747)
(S/N) 0 dB	✓ (307)	✓ (428)	✓ (369)	✓ (556)	✓ (591)
(N/S) 3 dB	✓ (226)	✓ (350)	✓ (304)	✓ (416)	✓ (480)
(N/S) 6 dB	✓ (142)	✓ (264)	✓ (258)	✓ (333)	✓ (359)
(N/S) 9 dB	✓ (86)	✓ (167)	✓ (127)	✓ (191)	✓ (214)
(N/S) 12 dB	✓ (62)	✓ (119)	✓ (95)	✓ (129)	✓ (144)
(N/S) 15 dB	✓ (42)	✓ (36)	✓ (48)	✓ (48)	✓ (89)

The ✓ symbol indicates a match and the value next to ✓ are the total landmarks identified. The average of all the database advertisement landmarks decreases can be seen in **Figure 42**. In **Figure 43** Avery Wang's results can be seen and the two can be compared [1].

Avery Wang's [1] results looks similar but are not exactly the same. This is due to the fact that our method is not an exact replica of the code, the noise might have been added differently and the test data used are not the same. For Avery Wang's results refer to **Addendum**.

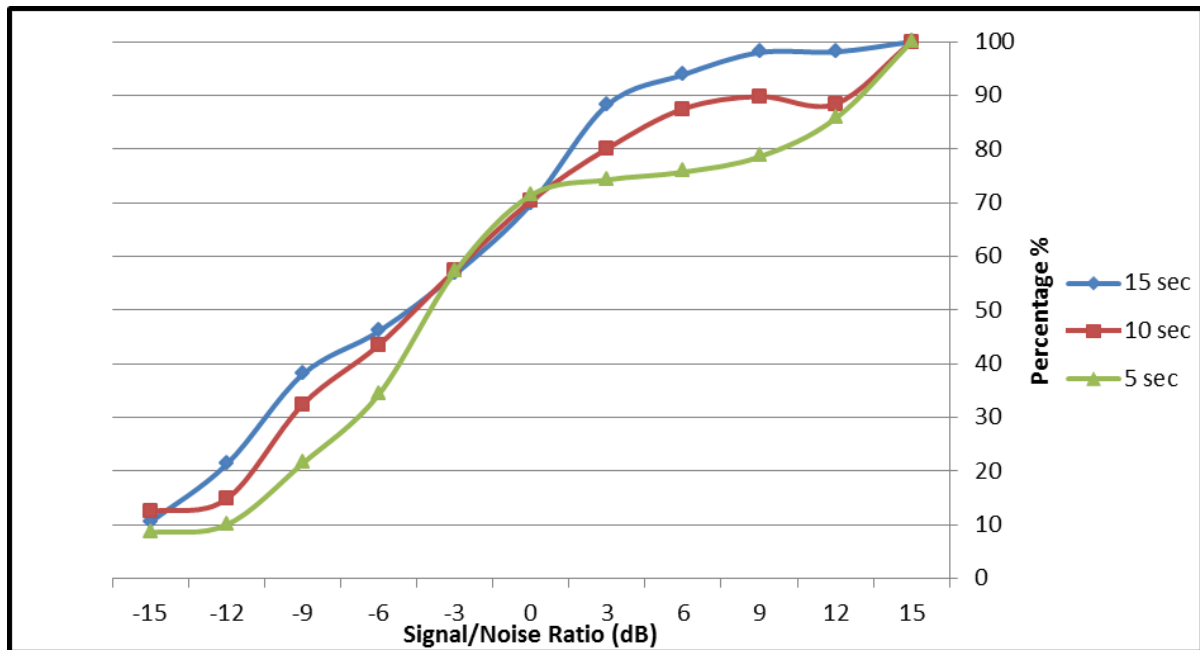


Figure 42 : Average of Gaussian white noise addition

The algorithm has more or less the same decline rate than the Shazam's algorithm. The algorithm is validated by the fact that the results came close to that of Shazam and the results that Avery Wang's paper indicates that their algorithm used spectral peaks and it is known that in this paper that the generic algorithm also used spectral peaks. The algorithm is therefore validated and verified as a robust landmark audio fingerprinting algorithm.

7.1.2.2 Face validity

The generic algorithm behaves desirably, though quite different from the algorithms it was compared to. This makes sense, as it is an isolated algorithm that was only based on a method that Avery Wang [1] used. The results came very close to that of Wang's and the speed (refer to **Table 9**) is in the range 5-500 milliseconds as Wang described [1]. The algorithm is logically validated in the sense that the algorithm was comparable to the other algorithms and it successfully detected advertisements (refer to section 7.2 **Advertisement identification**).

7.1.2.3 Predictive validation

The generic algorithms database was designed to hold 5 000 advertisements. With the optimised parameters, averages of 1 800 landmarks were identified per advertisement. This resulted in a 36% usage of the database, thus the database calculation was predicted successfully. The generic algorithm was predicted to withstand this same type of compression than the other algorithms and it does to an extent (refer to **Table 7** and **Table 11**). The algorithm identified each advertisement, thus the algorithm is as robust as predicted.

7.2 Advertisement identification

The algorithm was implemented for advertisement identification. For advertisement identification a number of 234 radio advertisements with total time amounting to an hour and 56 minutes were added to the database, with the advertisements ranging from 9 to 44 seconds in length.

The algorithm required 5 min and 25 seconds to add the advertisement to the database. A radio signal from OFM radio station with a length of 8 hours, 29 minutes and 59 seconds was provided. The radio signal was divided in 5 minute segments and specially renamed in the following format: Hour_minute_second.wav and Hour_minute_second.mp3 (depending on the format supplied). This allowed the application to identify the exact time the advertisement was played on the radio.

Although the algorithms are faster than real time, analysing every 3 seconds is unnecessarily time consuming. Therefore, the application is hopping in 15 second intervals and analysing the first 3 seconds of these intervals. This time saving technique allows for extra features discussed in **8.2 Recommendations**. With this technique we compare the generic algorithm vs. Haitsma and Kalker's (Phillips) algorithm see **Table 9**.

Table 9 : Advertisements results

Processing	Generic	Phillips
Adding to database	5 min 25s	20 min
Searching 3 seconds sample	116ms	15ms
Search through entire sample	7 min 44s	14 min 20s
Number of advertisements found	113	111
Time faster than real-time	66	36

The generic algorithm found all 113 advertisements in the radio signal provided, The Phillips algorithm only missed 2 (*italic and underlined advertisements in Table 12*), but none found a false positive. It must be noted that the radio audio is more or less noise and compression free. For these results, please refer to **Addendum**. Although Phillips seems faster with only a 3-second worth of database, it has a linear increase in search time as the database size increases. Whereas with the generic algorithm, the searching time is database size independent. The algorithm will therefore be almost 66 times faster than real-time, only slightly depending on the amount of landmarks retrieved. The highest false positive (using wav format) was 8 landmarks and the highest positive reached 112 landmarks. The hypothesis can now be made that a wav format requires more than 19 landmarks to be a match. Whereas with mp3 at 128 kbps it was 5 and 40 landmarks respectively which corresponds with Dan Ellis' finding, suggesting that the mp3's requirement is 9 landmarks [36] for a match.

In this chapter, the generic algorithm functionalities were verified, the algorithm validated and advertisement identification results published. The functionalities were discussed to ensure that the algorithm and implementations were correct. The algorithm was validated by comparing the results to Haitsma and Kalker's and Avery Wang's algorithm, face validity and predictive validation. The advertisement results showed that the generic algorithm was correctly implemented for advertisement identification. With the algorithm validated, verified and successfully implemented for advertisement identification, it is time to draw conclusions and make recommendations.

Chapter 8 – Conclusions and Recommendations

In this chapter, the research question and how the objectives were achieved is reviewed. A short summary about the contribution that the dissertation made is presented. A conclusion is drawn from the results. This is followed by recommendations for future work.

The research question as stated in section **1.4 Research methodology**, is as follows:

Can audio fingerprinting be applied successfully to advertisement monitoring, and if so, which existing audio fingerprinting algorithm is most suitable as a basis for a generic algorithm and how should the original algorithm be changed for this purpose?

The final answer to the research question is yes, because in section **7.2 Advertisement identification** the generic algorithm proved that it is capable of detecting advertisements. The generic algorithm was based on Avery Wang's algorithm, which is also referred to as the robust landmark audio fingerprinting algorithm or method. The generic algorithm was optimised for advertisement identification through the tweaking of the parameters.

8.1 Conclusion

Audio fingerprinting is an accurate, scalable, reliable and robust method of identifying audio segments. It is a good solution to monitor advertisements, rather of people monitoring them.

8.1.1 Achieved objectives

A detailed study of the technical aspects relating to audio fingerprinting was conducted. The technical aspects do not necessarily form part of the practical implementation of the audio fingerprinting algorithm, but technical background knowledge allows for a better accuracy, robustness and speed of the algorithm. The best available methods for the technical aspects were implemented at the time of implementation, considering the cost and effectiveness of the operations.

Different algorithms were studied, that of Haitzma and Kalker's [2], Avery Wang's [1] and the RARE [7] as identified by P.J.O. Doets [5] and based on their database effectiveness (most time consuming operation, when working with a large database). An algorithm was then selected to base the generic algorithm on. Avery Wang's hash table allows for a near constant searching time, independent of database size. This algorithm was therefore selected for optimization as a generic algorithm for the advertisement identification environment.

The generic algorithm was implemented as a practical solution for advertisement identification. The user interface did not only allow for a practical solution to advertisement identification, but also the means to implement the experiments specified so that the generic algorithm could be validated and verified. The generic algorithm was

verified by ensuring the algorithm and its functionalities were correctly implemented. Next, the algorithm was validated by comparing its results against that of Haitsma and Kalker's and Avery Wang's algorithm.

Sufficient results were obtained to validate and verify the generic algorithm, as well to demonstrate a successful advertisement identification of a radio signal.

8.1.2 Summary of contribution

In this dissertation different audio fingerprinting algorithms were compared which led to the conclusion that most algorithms are robust enough to combat the coffee shop problem [31], but have difficulties when large databases are used. The coffee shop problem refers to practical capabilities should be at least so robust that the solution works in an everyday coffee shop.

The comparison made it clear that the more standardised audio fingerprinting algorithm is that of Avery Wang (algorithms based on the robust landmark method), which can successfully implement a hash table with the same speed and accuracy.

With the optimised generic algorithm, it is now possible to identify samples that have fewer available frequencies. The field of audio fingerprinting now has an added application of identifying advertisements.

8.1.3 Interpreting results

When referring to **Table 7**, **Table 11** and **Figure 41**, it is clear that Haitsma and Kalker's algorithm has certain advantages when it comes to signal degradation (linear speed and time change), and the generic algorithm had more of an advantage with mp3 compression. The other degradations are more or less the same. When the algorithm is selected for advertisement identification, it is assumed that the channel will be rather noiseless with minimum compression. As Avery Wang's algorithm is faster for a large database and has better results against slight compression, it was selected to base the generic algorithm on. After testing both algorithms for advertisement identification, the generic algorithm identified two more advertisements than Haitsma and Kalker's algorithm, which proves again that Avery Wang's algorithm is the suitable choice of algorithm.

With regards to advertisement identification, the following aspects were taken into account:

- Database – Avery Wang's hash table is faster than any other algorithm that requires a linear search for the fingerprint. In the generic algorithm's database, not every hash will be filled as some frequency combinations are not contained in today's advertisements, and some hashes will result in large conflicts for this would be common frequency combinations, which most advertisements contain. As the audio that is recognisable by humans has a set structure, it is very unlikely that some of the hash combinations would ever exist in advertisements, since a 20% size increase is recommended to combat the theoretical hashes.

- Parameters – The theoretical environment suggests that the radio station is noise free and every advertisement is played without any other degradation than slight compression. As the radio presenter might speak during the piece of audio that is being identified, not all robustness parameters should be set for maximum speed and lowest robustness.
- Search time – Theoretically it is suggested that the search time is a constant time for every identification. As advertisements differs in length and in number of landmarks and there are hash conflicts of which the most suitable need to be identified, the actual search will never be the same than the theoretical search, but deviates in such a small way that a good estimate can be derived.
- Music in radio signal – Music results in most of the false positives, because as a result of the pre-processing they sometimes cause similar landmarks. The problem was solved by optimising parameters and updating the threshold to 30 hits to be a match.

8.1.4 To the point

The generic algorithm is 66 times faster than real-time and it is proven 100% accurate for the current database size and content. As the landmark technique is studied in more depth, taking in consideration its great speed, more features for this algorithm should be obtained. The algorithm proved that audio is just as identifiable as fingerprints. Advertisement companies and/or radio stations now have a tool for tracking advertisements. This allows for much better content control and enforcement of media laws.

8.2 Recommendations

Following the study of audio fingerprinting for advertisement identification, there will always be a few unstudied aspects and a potential for further study. This is formulated by the author and suggested as recommendations. Here are the recommendations:

8.2.1 Advertisement identification

As the generic algorithm was capable of detecting all advertisements in a more or less noise free and degraded free environment, there are still a few challenges left:

- Clipping or shortening of an advertisement – With the landmark method, it can be seen where the landmark was created in the advertisement. It should then be possible to determine the start and end position of the advertisement in the radio signal. Therefore, when an advertisement is shortened or clipped it should be detectable.
- Quality of advertisements – At the moment there is no way of determining the quality of advertisements played on the radio. If it is possible to map out the entire advertisement in the radio signal, it can then be compared to that of **Table 7**. If the expected amount of landmarks is not found, the application should flag

the radio signal for further investigation. If a proper study of this table is done it might be possible to identify the degradation with the error rating e.g. if an advertisement is played faster (linear speed change) and it receives an error rate between 0.16%-0.5%, the degradation is identified and the company now has proof of this degradation.

- Identify streaming audio – Currently it is a waste of space recording the radio signal and unnecessary labour renaming the audio to correctly identify the time. The application should be adjusted to use the computer clock and identify the advertisements in real-time.
- Parameters investigation – A more in depth understanding about the parameters should be acquired, for different applications.

8.2.2 Video identification

Currently video identification methods are slow, but if adjusted with the landmark algorithm, it should deliver a faster and new algorithm. The new algorithm should be applicable to one of the following applications:

- Advertisement identification – Identifying video advertisements from television signals.
- Copyright protection – If the algorithm is fast enough to scan the internet of various sites for copyright content.
- Media rights – Youtube already has an algorithm implemented to protect users from viewing sensitive content, but this process is slow.
- Object tracking – Creating landmarks of a piece of audio and track this objects in different videos.

8.2.3 Speaker identification

There is a forensic need to identify speakers, presenters, spokesman, narrators etc. Removing the mask feature and pairing the low frequency landmarks could maybe lead to voice prints. Possible voice prints applications:

- Terrorist prevention/tracking – If the algorithm is fast enough, it can monitor cell phone/landline calls for wanted criminals or terrorists.
- Court – If the algorithm is accurate enough to be used in a court of law.

In this chapter a conclusion of the whole dissertation was presented, the generic algorithm proved to be successful in advertisement identification. The effect of this research has been discussed, the objectives achieved and the results interpreted. This chapter ended with recommendations for future work.

Addendum

Table 10 : Kaiser window coefficients

h[0]	-6.17E-05	h[196]	h[34]	0.001407	h[162]	h[68]	-0.00817	h[128]
h[1]	8.95E-07	h[195]	h[35]	0.000139	h[161]	h[69]	-0.00166	h[127]
h[2]	8.57E-05	h[194]	h[36]	-0.00147	h[160]	h[70]	0.007767	h[126]
h[3]	7.09E-05	h[193]	h[37]	-0.00129	h[159]	h[71]	0.00779	h[125]
h[4]	-5.43E-05	h[192]	h[38]	0.000655	h[158]	h[72]	-0.00264	h[124]
h[5]	-0.00014	h[191]	h[39]	0.001949	h[157]	h[73]	-0.01069	h[123]
h[6]	-4.31E-05	h[190]	h[40]	0.000767	h[156]	h[74]	-0.00527	h[122]
h[7]	0.000139	h[189]	h[41]	-0.00158	h[155]	h[75]	0.007868	h[121]
h[8]	0.000167	h[188]	h[42]	-0.00207	h[154]	h[76]	0.01196	h[120]
h[9]	-3.73E-05	h[187]	h[43]	0.000172	h[153]	h[77]	0.000403	h[119]
h[10]	-0.00024	h[186]	h[44]	0.002434	h[152]	h[78]	-0.01315	h[118]
h[11]	-0.00014	h[185]	h[45]	0.001675	h[151]	h[79]	-0.01064	h[117]
h[12]	0.000172	h[184]	h[46]	-0.00141	h[150]	h[80]	0.006656	h[116]
h[13]	0.000311	h[183]	h[47]	-0.00294	h[149]	h[81]	0.017305	h[115]
h[14]	4.20E-05	h[182]	h[48]	-0.00069	h[148]	h[82]	0.005878	h[114]
h[15]	-0.00034	h[181]	h[49]	0.002727	h[147]	h[83]	-0.01535	h[113]
h[16]	-0.00032	h[180]	h[50]	0.002845	h[146]	h[84]	-0.01889	h[112]
h[17]	0.000149	h[179]	h[51]	-0.00084	h[145]	h[85]	0.003059	h[111]
h[18]	0.000493	h[178]	h[52]	-0.00378	h[144]	h[86]	0.024783	h[110]
h[19]	0.000214	h[177]	h[53]	-0.00197	h[143]	h[87]	0.016143	h[109]
h[20]	-0.00041	h[176]	h[54]	0.002664	h[142]	h[88]	-0.01709	h[108]
h[21]	-0.00057	h[175]	h[55]	0.004204	h[141]	h[89]	-0.03384	h[107]
h[22]	2.76E-05	h[174]	h[56]	0.000277	h[140]	h[90]	-0.00627	h[106]
h[23]	0.00068	h[173]	h[57]	-0.00442	h[139]	h[91]	0.038643	h[105]
h[24]	0.0005	h[172]	h[58]	-0.00368	h[138]	h[92]	0.04136	h[104]
h[25]	-0.00039	h[171]	h[59]	0.002053	h[137]	h[93]	-0.01821	h[103]
h[26]	-0.00088	h[170]	h[60]	0.005621	h[136]	h[94]	-0.07898	h[102]
h[27]	-0.00023	h[169]	h[61]	0.002033	h[135]	h[95]	-0.04633	h[101]
h[28]	0.000821	h[168]	h[62]	-0.00464	h[134]	h[96]	0.107805	h[100]
h[29]	0.000905	h[167]	h[63]	-0.00578	h[133]	h[97]	0.296393	h[99]
h[30]	-0.00023	h[166]	h[64]	0.000689	h[132]	h[98]	0.381406	
h[31]	-0.0012	h[165]	h[65]	0.006898	h[131]			
h[32]	-0.00067	h[164]	h[66]	0.004515	h[130]			
h[33]	0.000841	h[163]	h[67]	-0.00416	h[129]			

Table 11 are the results of Haitsma and Kalker's paper.

Table 11 : BER for different kinds of signal degradations [2]

Processing	AC/DC	Orff	Sinead	Texas
Mp3@128Kbps	0.084	0.078	0.085	0.081
Mp3@32Kbps	0.133	0.174	0.106	0.096
All-Pass filtering	0.027	0.019	0.015	0.018
Equalisation	0.062	0.048	0.045	0.066
Band Pass filter	0.038	0.028	0.025	0.024
Time scale +4%	0.206	0.202	0.183	0.200
Time scale -4%	0.203	0.207	0.174	0.190
Linear Speed +1%	0.238	0.172	0.102	0.132
Linear Speed -1%	0.196	0.243	0.142	0.260
Linear Speed +4%	0.472	0.438	0.467	0.355
Linear Speed -4%	0.431	0.464	0.438	0.470
Resampling	0.000	0.000	0.000	0.000

Figure 43 are the results of Avery Wang's paper.

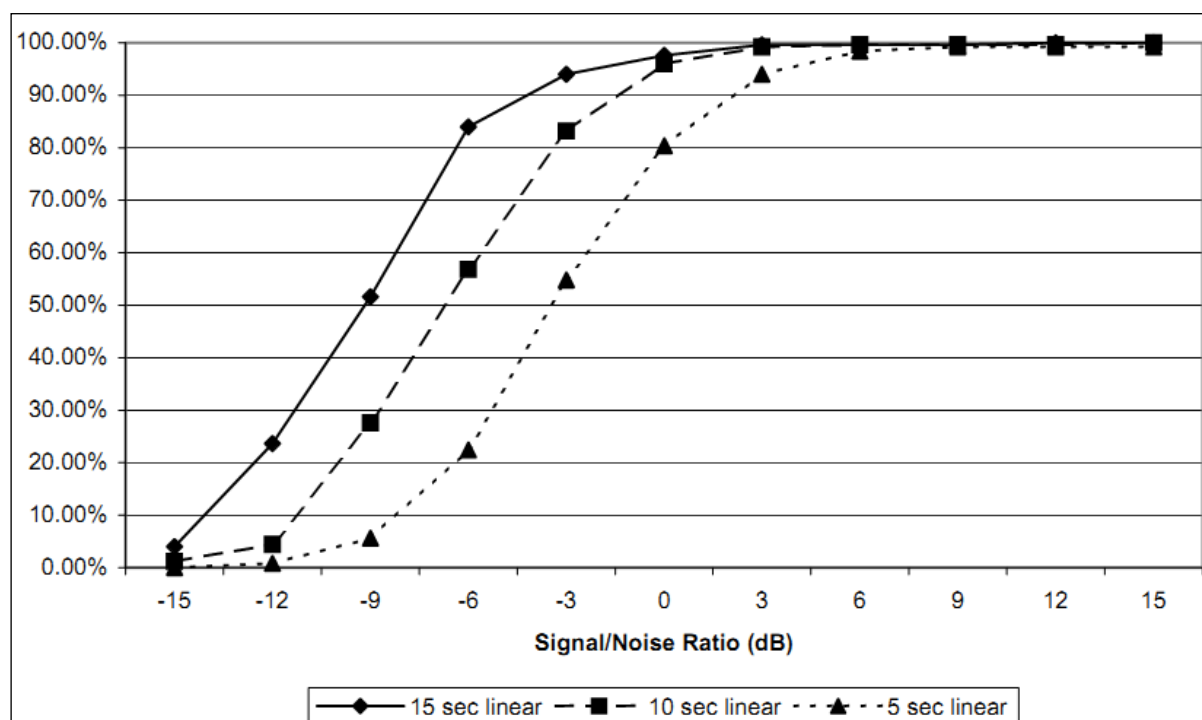


Figure 43 : Recognition rate -- Additive Noise [1]

Table 12 is the advertisement results.

Table 12 : Advertisements identified

<i>Company</i>	<i>Advertisement</i>	<i>Date</i>	<i>time</i>
Vodacom	SamsungCellphone	3/17/2009	15:39:18
HondaWinCentral	Cars	3/17/2009	15:39:48
CPE	DebtCounceler	3/17/2009	15:40:18
BloemSpa	Spa	3/17/2009	15:40:48
NorthCapeMall	Mall	3/17/2009	15:41:18
OFMHitMobile	ShopriteLiquoreShopHeidedal	3/17/2009	15:41:48
PerformingArtCenterFS	PerformingArt	3/17/2009	15:53:45
PGPison	BordProdukte	3/17/2009	15:54:15
OverLandLiquors	Liquors	3/17/2009	15:54:48
ProFitmentCenter	MmotorKlank	3/17/2009	15:55:19
Peugeot	Motor	3/17/2009	15:55:48
PrimePersonal	Personeel	3/17/2009	15:56:18
IntensiveMedicalRescue	AmbulansDienste	3/17/2009	15:56:47
MoorivierMall	Mall	3/17/2009	15:57:18
KovsieNuus	Nuus	3/17/2009	16:22:42
Sars	OmsetBelasting	3/17/2009	16:24:09
AutoPedigree	Cars	3/17/2009	16:24:49
LimaBlinds	Blinds	3/17/2009	16:25:19
FrontierInn	Casino	3/17/2009	16:25:49
TorgaOptical	Specticals	3/17/2009	16:40:19
Checkers	Groceries	3/17/2009	16:40:49
AndragAdricoToyota	Toyota	3/17/2009	16:41:20
Potchefstroom	City	3/17/2009	16:41:50
LimaBlinds	Blinds	3/17/2009	16:42:20
OFMHitMobile	ShopriteLiquoreShopHeidedal	3/17/2009	16:42:49
Purina1	Troeteldiere	3/17/2009	16:53:20
AlfaPharm	Pharmacy	3/17/2009	16:53:55
Midas	MotorOnderdele	3/17/2009	16:54:26
HyperPoolEquipment	SwembadOnderdele	3/17/2009	16:54:56
Elleton	MiningSupplies	3/17/2009	16:55:25
AlfaPharm	Apteek	3/17/2009	16:55:55
Bloemskou	Skou	3/17/2009	16:56:25
Huyndai	Cars	3/17/2009	17:00:34
SantamAgri	Oesversekering	3/17/2009	17:10:03
HondaWinCentral	Cars	3/17/2009	17:24:21
Checkers	Groceries	3/17/2009	17:24:52
BloemfonteinMediClinic	Clinic	3/17/2009	17:25:23
PGPison	BordProdukte	3/17/2009	17:25:52
SarelBotes	Versekering	3/17/2009	17:26:25
Daihatsu	Motors	3/17/2009	17:26:55
TriAngleBearings	Onderdele	3/17/2009	17:27:25
SlapChips	SlapChips	3/17/2009	17:27:55

FrontierInn	Casino	3/17/2009	17:38:42
YellowPages	CommercialInfo	3/17/2009	17:39:13
CentralNissan	Cars	3/17/2009	17:40:13
LiquorLand	Liquor	3/17/2009	17:40:43
Checkers	Groceries	3/17/2009	17:41:13
Windmill	Casino	3/17/2009	17:50:35
ANC	PoliticalParty	3/17/2009	17:51:09
OFMRiverSideHotel	Competition	3/17/2009	17:51:50
Shoprite	Kruideniersware	3/17/2009	17:52:26
VirginiaAuto	Motors	3/17/2009	17:52:56
Mogala	NasionalePark	3/17/2009	17:53:26
AfricanBrickCenter	Bricks	3/17/2009	17:53:55
Bloemskou	Skou	3/17/2009	17:54:25
KFC	BoxMeal	3/17/2009	18:02:52
Telkom	SmartAccess	3/17/2009	18:19:17
KonicaMinolta	BizHub	3/17/2009	18:20:18
Daihatsu	Motors	3/17/2009	18:20:48
CPE	DebtCounceler	3/17/2009	18:21:18
KFC	BoxMeal	3/17/2009	18:25:02
KFC	BoxMeal	3/17/2009	18:42:07
Soveran	Cars	3/17/2009	18:42:37
PGGlass	MotorGlas	3/17/2009	18:56:18
HondaWinCentral	Cars	3/17/2009	18:56:48
SkietWinkel	Vuurwapens	3/17/2009	18:57:18
Ubique	Afslaers	3/17/2009	18:57:48
KonicaMinolta	BizHub	3/17/2009	18:58:17
NicksDieselAndCarServices	MotorDiens	3/17/2009	18:58:48
RockBuildingSupplies	BuildingSupplies	3/17/2009	18:59:18
CaptainDorigo	Food	3/17/2009	19:01:03
Vodacom	SamsungCellphone	3/17/2009	19:23:07
ANC	PoliticalParty	3/17/2009	19:23:41
Vryheidsfront	PolitiekeParty	3/17/2009	19:24:17
Center360	Spilpunte	3/17/2009	19:24:35
Vryheidsfront	PolitiekeParty	3/17/2009	19:34:11
OFMRiverSideHotel	Competition	3/17/2009	19:34:51
AlfaPharm	Apteek	3/17/2009	19:35:27
Landbank	Bank	3/17/2009	19:48:11
SlenderWander	Dieet	3/17/2009	19:48:41
Bloemskou	Skou	3/17/2009	19:49:10
AutoPedigree	Cars	3/17/2009	19:49:40
CaptainDorigo	Food	3/17/2009	19:58:42
DayClub	Club	3/17/2009	20:15:36
<u>Bloemskou</u>	<u>Skou</u>	<u>3/17/2009</u>	<u>20:16:13</u>
Checkers	Kruideniersware	3/17/2009	20:16:36
PierresPlace	Restourant	3/17/2009	20:17:06
WilmedPark	PrivaatHospitaal	3/17/2009	20:17:38

TXAuto	GebruikteVoertuie	3/17/2009	20:18:08
RioResorting	Gambling	3/17/2009	20:18:38
<u>RealResorts</u>	<u>Gaming</u>	<u>3/17/2009</u>	<u>20:18:58</u>
PerformingArtScholerShip	Competition	3/17/2009	20:31:24
HomeMakersFair	HomeImprovement	3/17/2009	20:32:05
AlfaPharm	Pharmacy	3/17/2009	20:32:35
Suidwes	Kooperasie	3/17/2009	20:44:47
OFMRiverSideHotel	Competition	3/17/2009	20:45:24
Shoprite	Kruideniersware	3/17/2009	20:46:00
IntensiveMedicalRescue	AmbulansDienste	3/17/2009	20:46:29
MotlaSanaBuilders	Builders	3/17/2009	20:47:00
TriAngleBearings	Onderdele	3/17/2009	20:47:30
DermaPro	SkoonheidsProdukte	3/17/2009	20:48:00
CaptainDorigo	Food	3/17/2009	20:57:09
KovsieNuus	Nuus	3/17/2009	21:15:48
Checkers	Kruideniersware	3/17/2009	21:17:16
PerformingArtScholerShip	Competition	3/17/2009	21:34:39
PerformingArtScholerShip	Competition	3/17/2009	21:46:06
OFMRiverSideHotel	Competition	3/17/2009	22:21:54
OFMRiverSideHotel	Competition	3/17/2009	23:20:28
DayClub	Club	3/17/2009	23:50:51

Bibliography

- [1] Avery Li-Chun Wang, "An Industrial-Strength Audio Search Algorithm," *ISMIR*, no. 4th, pp. 7-13, October 2003.
- [2] Jaap Haitsma and Antonius Kalker, "A Highly Robust Audio Fingerprinting System," *Internasional Symposium on Music Information Retrieval (ISMIR)*, pp. 107-115, 2002.
- [3] (2011, May) audiohub. [Online]. <http://audiohub.org/get/fa/fa.htm>
- [4] Wes Hatch, "A Quick Review of Audio Fingerprinting," March 2003.
- [5] P.J.O Doets, M. Menor Gisbert, and R.L. Lagendijk, "On the comparison of audio fingerprints for extracting quality parameters of compressed audio," vol. 6072, 2006.
- [6] P Cano, E Batlle, T Kalker, and J Haitsma, "A review of algorithms for audio fingerprinting," *IEEE International Workshop on Multimedia Signal Processing(MMSP)*, pp. 169-173, December 2002.
- [7] Chris Burges, John Platt, Jon Goldstein, and Erin Renshaw. (2010) Microsoft Research. [Online]. research.microsoft.com/en-us/um/people/cburges/ppt/RARE.pdf
- [8] Microsoft. Microsoft. [Online]. <http://www.microsoft.com/net/overview.aspx>
- [9] Buddy Piper. (1981, September) Face the music. [Online]. <http://www.angelfire.com/wa/gsjackpot/facethemusic.html>
- [10] Christopher J.C Burges, John C Platt, and Soumya Jana, "Distortion Discriminant Analysis for Audio," vol. XX, no. Y, 2001.
- [11] Michael Pilhofer and Holly Day, *Music Theory for Dummies.*: For Dummies, 2007, ch. 2, p. 97.
- [12] *The American Heritage® Dictionary of the English Language*, 4th ed.: Houghton Mifflin Company, 2006.
- [13] B.C.J Moore, B.R Glasberg, and T Baer, "A model for the prediction of thresholds, loudness, and partial loudness," *Journal of the Audio Engineering Society*, vol. 45, no. 4, p. 232.
- [14] *Final Environmental Impact Statement for the North Pacific Acoustic Laboratory*, 1st ed. Arlington, VA: Office of Naval Research, 2001.
- [15] S. O. Coutant. (2000, June) Microphones. [Online]. <http://www.coutant.org/shure300/index.html>
- [16] 1911encyclopedia. (2006, October) Heinrich Rudolf Hertz. [Online]. http://www.1911encyclopedia.org/Heinrich_Rudolf_Hertz
- [17] Elert Glenn and Timothy Condon. (2008, October) The Physics Factbook. [Online]. <http://hypertextbook.com/facts/2003/TimCondon.shtml>
- [18] J H et.al McClellan, "A computer program for designing optimal fir linear phase digital filters," *IEEE Transaction on Audio and Electroacoustics*, vol. 6, no. AU-21, pp. 506-526, December 1973.
- [19] Emmanuel C Ifeakor and Barrie W Jervis, *Digital Signal Processing: A Practical Approach*, 2nd ed.: Prentice Hall, 2002.

- [20] Sanjit K Mitra, *Digital Signal Processing : A computer-based approach*, 3rd ed. Santa Barbara, United States of America: McGraw-Hill, 2006.
- [21] J.F Kaiser, "Nonrecursive digital filter design using the Io-sinh window function," in *IEEE International Symposium on Circuits and Systems*, San Francisco CA, April 1974, pp. 20-23.
- [22] Ingo R Titze, *Principles of Voice Production*, 1st ed. Michigan, United States of America: Prentice Hall, 1994.
- [23] Ronald J Baken and Robert F Orlikoff, *Clinical measurement of speech and voice*, 2nd ed., Cengage Learning, Ed.: Singular Thomson Learning, 2000.
- [24] (2011, September) engineeringtoolbox. [Online]. http://www.engineeringtoolbox.com/voice-level-d_938.html
- [25] L Shure and T Krauss. (2009, August) mathworks. [Online]. <http://www.mathworks.com/help/toolbox/signal/ref/fftfilt.html>
- [26] Julius O Smith III. (2010, March) Spectral Audio Signal Processing. [Online]. https://ccrma.stanford.edu/~jos/sasp/Choice_Hop_Size.html
- [27] (2011, June) compgroups. [Online]. <http://comgroups.net/comp.dsp/How-to-get-rid-of-resampling-artifacts>
- [28] SWGFAST. (2010, March) Scientific Working Group on Friction Ridge Analysis, Study and Technology. [Online]. http://www.swgfast.org/documents/glossary/090508_Glossary_2.0.pdf
- [29] Richard O Duda, Peter E Hart, and David G Stork, *Pattern Classification (2nd Edition)*.: Wiley-Interscience, 2000.
- [30] Yan Ke, Derek Hoiem, and Rahul Sukthankar. (2006, April) Computer Vision for Music Identification. pdf.
- [31] Dan P. W Ellis. (2011, April) E4896 Music Signal Processing. [Online]. <http://www.ee.columbia.edu/~dpwe/e4896/lectures/E4896-L13.pdf>
- [32] Pedro Cano, Eloi Batlle, Emilia Gómez, Leandro de C.T.Gomes, and Madeleine Bonnet, "Audio Fingerprinting: Concepts And Applications," *Studies in Computational Intelligence (SCI)*, no. 2, pp. 233-245, 2005.
- [33] James P. Ogle and Daniel P.W. Ellis, "Fingerprinting to identify repeated sound events in long-duration personal audio recordings," 2007.
- [34] Matthew L Miller, Manuel Acevedo Rodriguez, and Ingemar J Coxt, "Audio Fingerprinting: Nearest Neighbour Search in High Dimensional Binary Space," *The Journal of VLSI Signal Processing*, vol. 41, no. 3, pp. 285-291, December 2005.
- [35] Shumeet Baluja and Michele Covell, "Audio fingerprinting: Combining computer vision & data stream processing," *In International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 213-216, 2007.
- [36] Daniel P.W. Ellis. (2009) Robust Landmark-Based Audio Fingerprinting. [Online]. <http://labrosa.ee.columbia.edu/Matlab/fingerprint/>
- [37] Christopher J.C Burges, John C Platt, and Soumya Jana, "Extracting Noise-Robust Features from Audio Data," in *ICASSP*, 2002, pp. 1021-1024.

- [38] K. I Diamantaras and S. Y Kung, *Principal Component Neural Networks: Theory and Applications.*: Wiley, 1996.
- [39] Zoran Cindori. (2010) libZPlay library. [Online]. <http://libzplay.sourceforge.net/WELCOME.html>
- [40] <http://ac3filter.net/projects/tools>. [Online]. <http://ac3filter.net/projects/tools>
- [41] magix. [Online]. <http://www.magix.com/us/free-download/>
- [42] R.A Haddad and A.N Akansu, "A class of fast Gaussian binomial filters for speech and image processing," *Signal Processing, IEEE Transactions on*, vol. 39, no. 3, pp. 723 -727, March 1991.
- [43] Linda G Shapiro and George C Stockman, *Computer Vision*. Washington, United States of America: Prentice Hall, 2001.
- [44] Mark S Nixon and Alberto S Aguado, *Feature extraction and image processing*, 2nd ed.: Academic Press, 2008.
- [45] Robert G. Sargent, "Verification and validation of simulation models," in *Proceedings of the 37th conference on Winter simulation*. Orlando, Florida, United States of America: Winter Simulation Conference, 2005, pp. 130-143.

Appendix A – Conference contributions from this dissertation

IEEE African Winter School on Information Theory and Communications 2010

Mopani Camp, Kruger National Park, South Africa

June 2010

Workshop paper acceptance: *Comparison of Algorithms for Audio Fingerprinting*

SATNAC (Southern Africa Telecommunication Networks and Applications Conference)

Spier Estate, Stellenbosch, South Africa

September 2010

Work in progress paper acceptance: *Comparison of Algorithms for Audio Fingerprinting*

SATNAC (Southern Africa Telecommunication Networks and Applications Conference)

East London convention centre, East London, South Africa

September 2011

Full Paper acceptance: *The Study and Implementation of Shazam's Audio Fingerprinting Algorithm for Advertisement Identification*

IEEE African Winter School on Information Theory and Communications 2010

Comparison of Algorithms for Audio Fingerprinting

Heinrich A. van Nieuwenhuizen, Willim C. Venter and Leenta M.J. Grobler
School of Electrical, Electronic and Computer Engineering
North-West University, Potchefstroom Campus, South Africa
Email: {20252188; willie.venter; leenta.grobler}@nwu.ac.za
Telephone: (027) 18 299-4058 Fax: (027) 18 299-1977

Abstract—The practical implementation and application of audio fingerprinting to recognize specific audio signal are becoming more and more popular these days. Different research groups are working on implementations of audio fingerprinting, but they seldom, if ever compare their algorithms to those of other people working in the field. We can therefore not make a fair judgments of which of the available algorithms are suitable for certain applications. In this paper, we implement and compare two different audio fingerprinting algorithms in terms of accuracy, reliability, robustness, versatility and scalability.

Keywords: *Audio Fingerprinting; Automatic Music Reconigition; Content-based Audio Identification; Perceptual Hashing; Robust Matching;*

I. INTRODUCTION

I. INTRODUCTION

Fingerprinting systems is not a new concept; it has been around for more than a hundred years. In 1893 Sir Francis Galton was the first to “prove” that no two fingerprints of human beings were alike [2]. This notion was taken further by using any unique feature to identify an object; this includes the iris and even ears. People also realized the potential of constructing fingerprints of audio signals to identify and compare them. This principle is called audio fingerprinting. Audio fingerprints make use of short audio segments usually between 3-30 seconds (depending on the algorithm) to create an audio fingerprint. This audio fingerprint is compared to a database of known audio fingerprints to identify the original audio source. The audio fingerprint of the segments does not necessarily have to be of high quality to be a match. Distortions and interference of the original signal makes matching of the fingerprints less reliable, but (to a certain extent) it will still be recognizable. The distortions and interferences can be compared to a smudged or partial human fingerprint.

Haitsma and Kalker use an audio fingerprinting technique where a fingerprint of length 370ms is generated every 11.6ms. Non-semantic features are used to generate the fingerprints. The overlap of fingerprints in their technique ensures that their audio fingerprints vary slowly over time [2]. The company Shazam uses the peaks of the spectrogram of an audio segment to generate audio fingerprints [4].

The fingerprints of audio segments to be identified are

stored as a hash code in a database as a 32 bit vector. Typical databases are extremely large, consisting of at least one million songs and each song has at least 10,000 unique fingerprints [3]. Different techniques are used to make the database search effective.

II. SUMMARY

In this paper 2 different fingerprinting algorithms are compared, that of Haitsma and Kalker [2] and Avery Wang’s Shazam’s implementation [4]. The algorithms are compared according to the following criteria: accuracy, reliability, robustness, versatility and scalability, in order to determine which technique is more suited for a specific application.

There are several applications for audio fingerprinting algorithms. According to Wes Hatch [5] the biggest benefactor would be the broadcast monitoring industry. Other applications would be playlist generation, royalty collection, program verification and advertisement verification.

III. BIBLIOGRAPHY

- [1] S. Baluja and M. Covell, "Audio Fingerprinting: Combining Computer Vision & Data Stream Processing," In International Conference on Acoustics, Speech, and Signal Processing (ICASSP), pp. 213-216, 2007.
- [2] J. Haitsma and A. Kalker, "A Highly Robust Audio Fingerprinting System," *International Symposium on Music Information Retrieval (ISMIR)*, pp. 107-115, 2002.
- [3] M. L. Miller, M. A. Rodriguez, and I. J. Coxt, "Audio Fingerprinting: Nearest Neighbor Search in High Dimensional Binary Space," *The Journal of VLSI Signal Processing*, vol. 41, no. 3, pp. 285-291, Dec. 2005.
- [4] A. L.-C. Wang, "An Industrial-Strength Audio Search Algorithm," *ISMIR*, 2003.
- [5] W. Hatch, "A Quick Review of Audio Fingerprinting," Mar. 2003

Heinrich van Nieuwenhuizen received his B.Eng degree in 2009 and is currently pursuing his M.Eng at the North West University, Potchefstroom campus. His research interests include software design, audio fingerprinting and implementation and comparison of audio fingerprinting algorithms for industrial use

Comparison of Algorithms for Audio Fingerprinting

Heinrich A. van Nieuwenhuizen, Willim C. Venter and Leenta M.J. Grobler

School of Electrical, Electronic and Computer Engineering
North-West University, Potchefstroom Campus, South Africa
Email: {20252188; willie.venter; leenta.grobler}@nwu.ac.za
Telephone: (027) 18 299-4058 Fax: (027) 18 299-1977

Abstract—The practical implementation and application of audio fingerprinting to recognize specific audio segments are becoming more and more popular these days. Different research groups are working on implementations of audio fingerprinting, but they seldom, compare their algorithms to those of other people working in the field. A fair judgment can therefore not be made of which of the available algorithms is suitable for certain applications. In this paper, two audio fingerprinting algorithms are tested that of Avery Wang's and Haitsma and Kalker's in terms of accuracy, speed, versatility and scalability

Keywords: Audio Fingerprinting; Automatic Music Recognition; Content-based Audio Identification; Perceptual Hashing; Robust Matching;

I. INTRODUCTION

Fingerprinting systems are not a new concept; it has been around for more than a hundred years. In 1893 Sir Francis Galton was the first to "prove" that no two fingerprints of human beings are alike [1]. This notion was taken further by using any unique feature to identify an object; this includes the iris and even ears. People also realized the potential of constructing fingerprints of audio signals to identify and compare them. This principle is called audio fingerprinting.

Audio fingerprints make use of short audio segments usually between 3-30 seconds in length (depending on the algorithm) to create an audio fingerprint. This audio fingerprint is compared to a database of known audio fingerprints to identify the original audio source. The audio fingerprints of the segments do not necessarily have to be of high quality to be a match. Distortions and interference of the original signal makes matching of the fingerprints less reliable, but (to a certain extent) it will still be recognizable. The distortions and interferences can be compared to a smudged or partial human fingerprint.

In this paper a brief description about audio fingerprinting, a generic code of Avery Wang's Shazam algorithm and that of Haitsma and Kalker's algorithm are discussed [2].

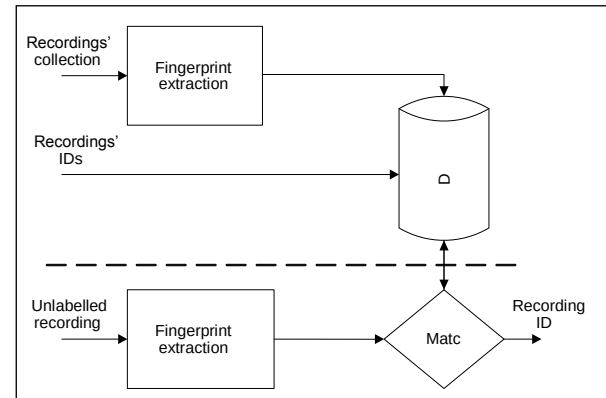


Figure 1: Content-based audio identification framework [6]

In section II, the three main groups of audio fingerprinting techniques are briefly discussed, after which the operation of Avery Wang's Shazam [2] and Haitsma and Kalker [1] algorithm are briefly discussed. In section III the operation of the algorithms are discussed. In Section IV the validation and verification is presented. The paper is ended with a conclusion and future work in section V.

II. DIFFERENT AUDIO FINGERPRINTING TECHNIQUES

There are several applications for audio fingerprinting algorithms. According to Wes Hatch [3] the biggest benefactor would be the broadcast monitoring industry. Other applications would be playlist generation, royalty collection, program verification and advertisement verification.

According to P.J.O Doets, M. Menor Gisbert and R.L. Lagendijk there are three groups [4] which audio fingerprinting can be classified as:

Group 1: Systems that use features based on multiple subbands, namely Philips' Robust Hash algorithm, reported to be very robust [1] against distortions. Phillips uses Haitsma and Kalker's algorithm.

Group 2: Systems that use features based on a single band such as the spectral domain, namely Avery Wang's Shazam and Fraunhofer's AudioID algorithms.

Group 3: Systems using a combination of subbands or frames, which is optimized through training, namely

Microsoft's Robust Audio Recognition Engine (RARE) that uses Hidden Markov Models (HMMs). [5]

III. OPERATION

The Haitsma and Kalker's algorithm propose that a fingerprint extraction scheme should be based on a general streaming approach, taking an audio signal framing it into windows of 370ms length for every 11.6ms giving it an overlapping factor of 31/32 [1].

After which the FFT of every frame is computed filter though band division stored into 32-bit sub Fingerprints.

See **figure 2** below

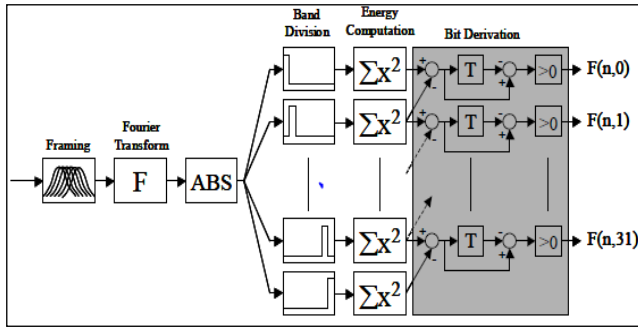


Figure 2: Haitsma and Kalker's algorithm [1]

An unidentified sample of an audio signal should be 3-30 seconds in length for their algorithm to identify a match.

Avery Wang claims for a database of 20 thousand tracks implemented on a PC, the search time is 5 to 500 milliseconds [2]. As the code is not available, generic code for Matlab was produced by Dan Ellis [5]. Robert Macrae of C4DM Queen Mary University London altered the code for use in the windows environment.

The latter algorithm proposed to make use of a spectrogram. The spectrogram is the squared magnitude of the STFT (Short-time Fourier Transform).

$$\text{spectrogram}(t, \omega) = |STFT(t, \omega)|^2 \quad (1)$$

Usually the spectrogram is divided into small sizes (typically 512 points) which are called windows or frames. This is the shared basis of group 2. The differences between the fingerprint algorithms in the group typically involve how much the frames overlap, and how the fingerprint is defined in the frame and the storing and searching of the fingerprints. Avery Wang Shazam algorithm uses the energy peaks in the frame and form spectral pair landmarks. The local maxima within a defined section are grouped into pairs [4].

The hash values are computed and compared the entry with the most hits is returned as the match. (Typically more than 9 spectral peaks are considered a match [7].)

IV. VALIDATION AND VERIFICATION

The two algorithms are implemented. Haitsma and Kalker's results are compared to that of their own "A Highly Robust Audio Fingerprinting System" [1] and Avery Wang's compared to his own "An Industrial-Strength Audio Search Algorithm" [2] and that of Dan Ellis "Robust Landmark-Based Audio Fingerprinting" [5]. Verifying that the algorithms are correctly implemented.

The two algorithms are then compared to each other in terms of accuracy, speed, versatility and scalability. The algorithms are further tested by subjecting the data to noise and compression and the algorithms parameters are tweaked for better results. Comparing the algorithms in the latter terms but with different data ranging from classical music to pop and advertisements.

V. CONCLUSION AND FUTURE WORK

The following were observed in the study.

Increasing and decreasing both algorithms frame size will decrease speed but increase the accuracy respectfully.

Defining more peaks in the Shazam's algorithm's frame (normally 5) would also result in better accuracy but decrease speed. Increasing the overlapping regions in both algorithms will increase robustness but decrease speed.

The nearest false positive is on average 5 or 6, which strengthens James .P Ogle and Daniel P.W. Ellis theory that 9 peaks[7] are needed to be a match.

In the future the following should be tried

Translating the algorithms to a faster programing environment. Further research should be done on different techniques to access the database quicker.

More application uses should be investigated e.g. gunshots, engine noise etc.

VI. BIBLIOGRAPHY

- [1] J. Haitsma and A. Kalker, "A Highly Robust Audio Fingerprinting System," *International Symposium on Music Information Retrieval (ISMIR)*, pp. 107-115, 2002.
- [2] A. L.-C. Wang, "An Industrial-Strength Audio Search Algorithm," *ISMIR*, 2003.
- [3] W. Hatch, "A Quick Review of Audio Fingerprinting," Mar. 2003.
- [4] P. J. O. Doets, M. M. Gisbert, and R. L. Lagendijk, "On the comparison of audio fingerprints for extracting quality parameters of compressed audio," vol. 6072, 2006.
- [5] D. P. W. Ellis. (2009) Robust Landmark-Based Audio Fingerprinting. <http://labrosa.ee.columbia.edu/Matlab/fingerprint/>
- [6] P. Cano, E. Batlle, E. Gómez, L. de C.T.Gomes, and M. Bonnet, "Audio Fingerprinting: Concepts And Applications," *Studies in Computational Intelligence (SCI)*, no. 2, pp. 233-245, 2005.
- [7] J. P. Ogle and D. P. W. Ellis, "Fingerprinting to identify repeated sound events in long-duration personal audio recordings," 2007.

Heinrich van Nieuwenhuizen received his B.Eng degree in 2009 and is currently pursuing his M.Eng at the North West University, Potchefstroom campus. His research interests include software design, audio fingerprinting and implementation and comparison of audio fingerprinting algorithms for industrial use.

SATNAC (Southern Africa Telecommunication Networks and Applications Conference) 2011

The Study and Implementation of Shazam's Audio Fingerprinting Algorithm for Advertisement Identification

Heinrich A. van Nieuwenhuizen, Willim C. Venter and Leenta M.J. Grobler
School of Electrical, Electronic and Computer Engineering
North-West University, Potchefstroom Campus, South Africa
Email: {20252188; willie.venter; leenta.grobler}@nwu.ac.za
Telephone: (027) 18 299-1961 Fax: (027) 18 299-1977

Abstract- The recognition of a person by his/her fingerprint is not a new concept, but the recognition of a piece of audio by its audio sample, also known as its audio fingerprint, is. Different research groups have delivered different working implementations of audio fingerprinting for music, but not for advertisement identification. A fair judgment can therefore not be made whether the available algorithms are suitable for advertisement identification. In this paper, the audio fingerprinting algorithm of Avery Wang's is implemented and studied in terms of accuracy, speed, versatility and scalability

Keywords: Audio Fingerprinting; Automatic Music Recognition; Content-based Audio Identification; Perceptual Hashing; Robust Matching;

I. INTRODUCTION

Fingerprinting systems are not a new concept; they have been around for more than a hundred years. In 1893 Sir Francis Galton was the first to "prove" that no two fingerprints of human beings are alike [1]. This notion was taken further by using any unique feature to identify an object; this includes the iris and even ears. People also realized the potential of constructing fingerprints of audio signals to identify and compare them; a principle known as audio fingerprinting.

A question which can be raised is why audio fingerprinting? Is there not an easier way, for example cross-correlation? The answer is not quite that simple, although if any simple mathematical comparison equations were used on a normalized and an identical WAVE file, it would probably work. When the same song on radio or on a CD is heard, they might sound similar, but the truth is that they are not the same mathematically - especially when noise is added or adjustments are made to the audio. The solution for a reliable, robust and accurate solution is extracting unique features using audio fingerprinting.

There are several applications for audio fingerprinting algorithms. According to Wes Hatch [3], the biggest benefactor would be the broadcast monitoring industry. Other applications would be playlist generation, royalty collection, program verification and advertisement verification.

Hatch's research inspired the advertisement identification technique for this paper.

Audio fingerprints are generated from short audio segments usually between 3-30 seconds in length (depending on the algorithm). This audio fingerprint is compared to a database of known audio fingerprints to identify the original audio source as seen in Figure 1.

The audio fingerprints of the segments do not necessarily have to be of high quality to be a match. Distortions and interference of the original signal makes matching of the fingerprints less reliable, but to a certain extent, it will still be recognizable. The distortions and interferences can be compared to a smudged or partial human fingerprint.

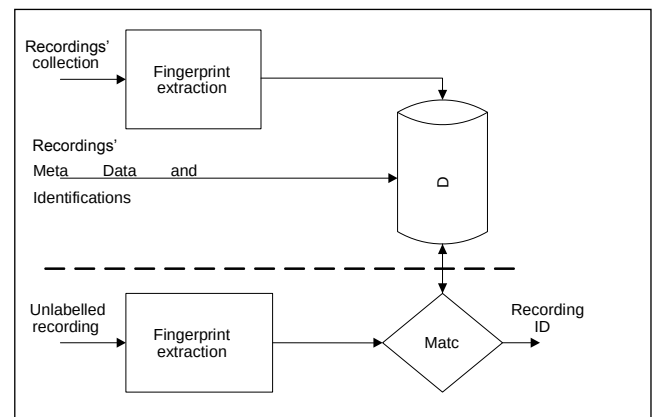


Figure 1: Content-based audio identification framework [7]

When you hear an advertisement on the radio, you take in the information and continue to listen to the next song or advertisement. You do not actually notice whether the advertisement has been clipped (shortened) or played faster than the initial recording.

This leaves the advertiser with a dilemma: even if they were to appoint personnel to verify that their advertisement was played correctly, human error is always possible. It is also too time consuming for a human to sit and listen and verify each advertisement [3].

Identifying these advertisements with audio fingerprinting is an exciting new solution. By analyzing the audio of a radio signal, all the advertisements can be identified.

There is however the question of accuracy, reliability and the speed at which the algorithm detects the advertisements.

Another problem is that the algorithms were designed for music purposes.

It is shown in Figure 2 that voice does not have the same frequency response as music, which makes identifying advertisements (which typically consist of 80% voice data) quite difficult.

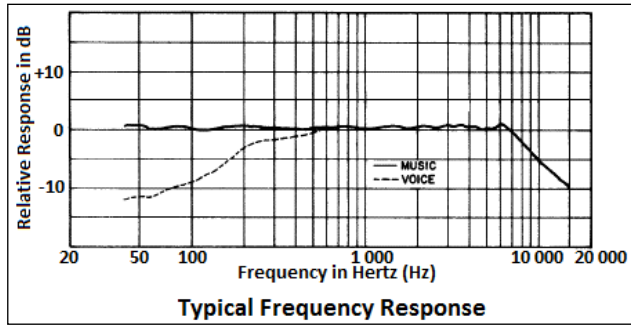


Figure 2: Typical Frequency Response [9]

In this paper a brief description of audio fingerprinting and an adaptive code of Avery Wang's Shazam algorithm are discussed.

In section II, the three main groups of audio fingerprinting techniques are briefly reviewed, after which the operation of Avery Wang's Shazam [2] algorithm is discussed in section III. After evaluating the performance of the algorithm, we describe the practical implementation of advertisement identification in section IV. In section V the results are published while the validation and verification is presented in section VI. The paper ends with a conclusion and potential future work is highlighted in section VII.

II. DIFFERENT AUDIO FINGERPRINTING TECHNIQUES

According to P.J.O Doets, M. Menor Gisbert and R.L. Lagendijk there are three groups [4] into which audio fingerprinting can be categorized:

Group 1: Systems that use features based on multiple subbands, namely Philips' Robust Hash algorithm, reported to be very robust against distortions [1]. Phillips uses Haitsma & Kalker's algorithm.

Group 2: Systems that use features based on a single band such as the spectral domain, namely Avery Wang's Shazam and Fraunhofer's AudioID algorithms.

Group 3: Systems using a combination of subbands or frames, which are optimized through training, namely Microsoft's Robust Audio Recognition Engine (RARE) which uses Hidden Markov Models (HMMs) [5].

For this paper we are only interested in Group 2, as the commonly known algorithm, Avery Wang's Shazam, which falls in this group, was chosen for this study.

III. OPERATION

All audio data used in the advertisement operation will be sampled from radio advertisements heard on your everyday radio station. Both Avery Wang's Shazam and Haitsma & Kalker's [2] algorithms were applied in the following scenario, but Avery Wang's algorithm was deemed more suitable for today's radio advertisements and was therefore studied in greater detail.

Avery Wang claims that for a database of 20 000 tracks, implemented on a PC, the search time is between 5 to 500 milliseconds [2]. As the code is not available, adaptive code for Matlab™ was produced by Dan Ellis [6]. Robert Macrae of C4DM Queen Mary University, London, altered the code for use in the Windows environment and the authors in turn altered and implemented the code for use in advertisement identification. The code was reproduced in VB.NET.

The proposed algorithm makes use of a spectrogram. The spectrogram is the squared magnitude of the STFT (Short-time Fourier Transform).

$$\text{spectrogram}(t, \omega) = |STFT(t, \omega)|^2 \quad (1)$$

Usually the spectrogram is divided into small fragments (typically 512 points) which are called *windows* or *frames*.

This is the shared basis of Group 2. The differences between the fingerprint algorithms in the group typically involve how much the frames overlap, how the fingerprint is defined in the frame and the storing and searching of the fingerprints.

Avery Wang's Shazam algorithm uses the energy peaks in the frame and form spectral pair landmarks. They chose spectral peaks for their robustness against noise and approximate linear superposability [2]. The local maxima within a defined section are grouped into pairs [8].

A. Properties investigated

In this study the following was observed.

Decreasing the frame size from the common value of 512 to 256 or 128 will increase the accuracy but decrease speed, as there will be more frames - meaning more peaks.

Defining more peaks in a frame (normally 5) would also result in better accuracy but decreased speed. Overlapping regions will increase robustness but decrease speed.

Depending on the specific application, these parameters can be tweaked accordingly. The generic code is based on Shazam's concept, so logically it is optimized for use in cell phone applications (which is typically subjected to noise and where speed is not a huge concern, but robustness and accuracy is).

For an application which is not subjected to noise and which requires real time analysis, the frame size will be increased, peaks will decrease and there would be no overlapping.

IV. ADVERTISEMENT IDENTIFICATION

After scientifically implementing and analyzing the algorithm, a clear next step would be practical application. As the adaptive Shazam code was a slightly better candidate with a more practical hash table, it is chosen for the practical application.

A sample containing 234 radio advertisements with total time amounting to 1 hour and 56 minutes was used, with the advertisements ranging from 9 to 44 seconds.

A sample with a length of 8 hours, 29 minutes and 59 seconds containing radio data was used and divided into 5 minute segments. They were renamed to the following format:

Hour_minute_second.wav

or

Hour_minute_second.mp3

This allowed the application to identify the exact time that the advertisements were played on the radio.

Even with the algorithm analyzing a sample every 3 seconds, the process was too time consuming. A faster solution was analyzing a 3 second audio segment at 15 seconds intervals.

V. RESULTS

Shazam's results were conducted on an Intel® Core2 Duo™ processor running at 2.1 GHz with access to 4 GB of memory. Phillips' results were derived from Haitsma & Kalker's paper [1].

All files, before processing, were sampled at 44.1 kHz in stereo at 16bps (bits per second), in accordance with Haitsma & Kalker's paper [1]. This insures accurate comparison between algorithms.

Table 1 : Advertisements results

Processing	Shazam	Phillips
Adding to database	5min 25s	20 min
Searching 3 seconds sample	116ms	15ms
Search through entire sample	30 min 56s	14 min 20s
Number of advertisements found	113	111
Time faster than real-time	16	36*

*see explanation below

It can be seen in Table 1 that Haitsma & Kalker's algorithm might have the advantage for this database size, but the algorithm displays a linear time increase with respect to database size, whereas Avery Wang's algorithm remains relatively constant.

When 300 minutes worth of audio (approximately 60 songs) is inserted into the database, Avery Wang's algorithm searches through a 5 minute unknown audio segment in 19.8 seconds and Haitsma & Kalker in 20.696 seconds.

The search time for a 5 minute sample of unknown audio is determined by the database size as shown in Figure 3.

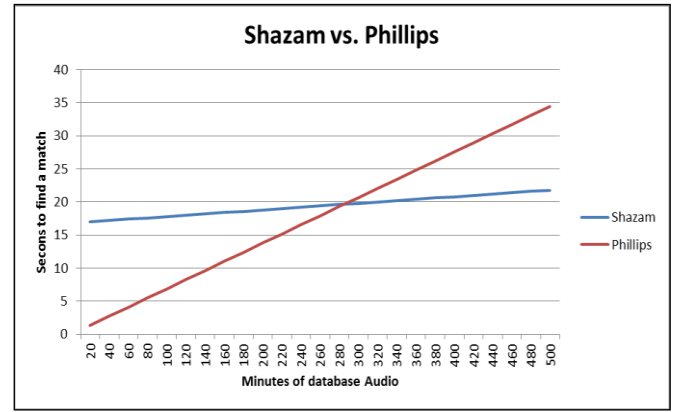


Figure 3: Shazam vs. Phillips

The adapted Shazam algorithm found all 113 advertisements in the radio audio provided, while Haitsma & Kalker's algorithm missed two. Neither of them found a false positive with accordance to their thresholds. It should be noted that the radio audio is virtually noise free and uncompressed.

The largest false positive found when using the sample in wav format was 8 landmarks and the largest positive was 112, while the mp3 sample at 128kbps found 5 and 40 respectively. This supports the decision to use a threshold of 19 and 9 for the wav and mp3 formats respectively and this corresponds with Dan Ellis' findings [6] suggesting the threshold for mp3 is 9 landmarks.

The unique feature which the Shazam algorithm exhibits is its ability to group spectral peaks (Shazam's audio fingerprint), which is of particular use in advertisement identification, as this allows multiple fingerprints to be detected or excluded on the same piece of audio.

This is very helpful to produce an identification match even when a radio presenter talks whilst the advertisement is playing.

VI. VALIDATION AND VERIFICATION

To compare the authenticity of the adaptive Shazam algorithm, the search time was compared to that of Avery Wang's paper [2].

It was not exactly the same but this can be attributed to many different reasons, including the fact that the code was not entirely identical and the same computer was not used. The adaptive code was also compared to that of Dan Ellis' adaptive Matlab code which is also derived of the Shazam's algorithm and the results were also very similar.

As Avery Wang's algorithm is not publically available, the verification is only approximate, but as much as possible was derived from theory and the results prove that this is a functional algorithm. Even if the adaptive algorithm does not entirely match that of Avery Wang's algorithm, it can still be indisputably classified as a Group 2 algorithm.

A. Conclusion

The following was observed in the study.

The algorithm was implemented and to verify its accuracy it was compared to that of Avery Wang's "An Industrial-Strength Audio Search Algorithm" [2] and Dan Ellis' "Robust Landmark-Based Audio Fingerprinting" [6]. It was verified that the algorithms are correctly implemented.

The adaptive code based on Wang's algorithm is a clear option for advertisement identification. It was seen that decreasing the algorithm frame size, will decrease speed but increase the accuracy respectfully.

Defining more peaks in the Shazam's algorithm's frame (normally 5) would also result in better accuracy, but decreased speed. Increasing the overlapping regions in the algorithm will increase robustness but decrease speed.

The nearest false positive found with mp3 compression at 128kbps averages 5 or 6, which strengthens James P Ogle and Daniel P.W. Ellis' theory that 9 peaks [8] are needed for a match. It was found that the nearest false positive when using the wav format was 8 and, as all matches had 26 peaks or higher, a safe threshold of 19 peaks was determined.

Overall, impressive results for the algorithm were obtained, analyzing radio signals 16 times faster than real-time, which in turn allow more data to be analyzed using larger databases. This will make the algorithm more lucrative for advertisement companies.

B. Future work

The following additional work can be done in the future:

Different algorithms should be compared to each other with respect to audio and advertisement identification, including robustness.

More application uses for audio fingerprinting should be investigated, e.g. gunshots, engine noise etc.

The prospects of using audio fingerprinting algorithms in cases where advertisements are read should be investigated.

The possibility of video identification using audio fingerprinting techniques (video fingerprinting), with the use of Avery Wang's Shazam algorithm should be explored.

The algorithm's use in voice identification using specific thresholds should be investigated.

Coding algorithm should be looked at for increase speed and larger database sizes

- [1] Haitsma, Jaap and Kalker, Antonius, "A Highly Robust Audio Fingerprinting System." International Symposium on Music Information Retrieval (ISMIR), Eindhoven : s.n., 2002, pp. 107-115.
- [2] Wang, Avery Li-Chun., "An Industrial-Strength Audio Search Algorithm." ISMIR, London : Shazam Entertainment, Ltd., 2003.
- [3] Hatch, Wes., "A Quick Review of Audio Fingerprinting." March 2003.
- [4] Doets, P.J.O, Gisbert, M. Menor and Legendijk, R.L., "On the comparison of audio fingerprints for extracting quality parameters of compressed audio." [ed.] Edward J. Delp III and Ping Wah Wong. Delft : Security, Steganography, and Watermarking of Multimedia Contents VIII, 2006, Vol. 6072.
- [5] Burges, Christopher J.C, Platt, John C and Jana, Soumya., "Distortion Discriminant Analysis for Audio." s.l. : IEEE TRANSACTIONS ON SPEECH AND AUDIO PROCESSING, ZZ, Issue Y, Vol. XX.
- [6] Ellis, Daniel P.W., "Robust Landmark-Based Audio Fingerprinting." [Online] 2009. <http://labrosa.ee.columbia.edu/Matlab/fingerprint/>.
- [7] Cano, Pedro, et al., "Audio Fingerprinting: Concepts And Applications." Studies in Computational Intelligence (SCI), 2005, Issue 2, pp. 233-245.
- [8] Ogle, James P. and Ellis, Daniel P.W., "Fingerprinting to identify repeated sound events in long-duration personal audio recordings." New York : s.n., 2007.
- [9] <http://www.coutant.org/shure300/index.html>

This work was completed at the Telkom Centre of Excellence

At the NWU, and is funded by the HTBO THRIP project

Heinrich van Nieuwenhuizen received his B.Eng degree in 2009 and is currently pursuing his M.Eng at the North West University, Potchefstroom campus. His research interests include software design, audio fingerprinting and implementation and comparison of audio fingerprinting algorithms for industrial use.