Kunnskap for en bedre verden

Bacheloroppgave

IE303612

MetaTracks: Acoustic fingerprinting using computer-vision techniques

802, 842, 810

Totalt antall sider inkludert forsiden: 168

(Vedlegg: 3)

Innlevert Ålesund, 26/05-2016

Obligatorisk egenerklæring/gruppeerklæring

Den enkelte student er selv ansvarlig for å sette seg inn i hva som er lovlige hjelpemidler, retningslinjer for bruk av disse og regler om kildebruk. Erklæringen skal bevisstgjøre studentene på deres ansvar og hvilke konsekvenser fusk kan medføre. **Manglende erklæring fritar ikke studentene fra sitt ansvar**.

Du/dere fyller ut erklæringen ved å klikke i ruten til høyre for den enkelte del 1-6:		
1.	**Jeg/vi erklærer herved at min/vår besvarelse er mitt/vårt eget arbeid, og at jeg/vi ikke har brukt andre kilder eller har mottatt annen hjelp enn det som er nevnt i besvarelsen.**	☒
2.	**Jeg/vi erklærer videre at denne besvarelsen:** • ikke har vært brukt til annen eksamen ved annen avdeling/universitet/høgskole innenlands eller utenlands. • ikke refererer til andres arbeid uten at det er oppgitt. • ikke refererer til eget tidligere arbeid uten at det er oppgitt. • har alle referansene oppgitt i litteraturlisten. • ikke er en kopi, duplikat eller avskrift av andres arbeid eller besvarelse.	☒
3.	Jeg/vi er kjent med at brudd på ovennevnte er å betrakte som fusk og kan medføre annullering av eksamen og utestengelse fra universiteter og høgskoler i Norge, jf. Universitets- og høgskoleloven §§4-7 og 4-8 og Forskrift om eksamen.	☒
4.	Jeg/vi er kjent med at alle innleverte oppgaver kan bli plagiatkontrollert i Ephorus, se Retningslinjer for elektronisk innlevering og publisering av studiepoenggivende studentoppgaver	☒
5.	Jeg/vi er kjent med at høgskolen vil behandle alle saker hvor det forligger mistanke om fusk etter NTNUs studieforskrift.	☒
6.	Jeg/vi har satt oss inn i regler og retningslinjer i bruk av kilder og referanser på biblioteket sine nettsider	☒

Publiseringsavtale

Studiepoeng: 20

Veileder: Kjell Inge Tomren

Jeg/vi gir herved NTNU i Ålesund en vederlagsfri rett til å gjøre oppgaven tilgjengelig for elektronisk publisering:	☒ja	☐nei
Er oppgaven båndlagt (konfidensiell)? **(**Båndleggingsavtale må fylles ut)	☐ja	☒nei
- Hvis ja: **Kan oppgaven publiseres når båndleggingsperioden er over?**	☐ja	☐nei
Er oppgaven unntatt offentlighet? (inneholder taushetsbelagt informasjon. Jfr. Offl. §13/Fvl. §13)	☐ja	☒nei

Dato: 26.05-2016

METATRACKS: ACOUSTIC FINGERPRINTING

A system for real-time media identification using computer-vision techniques

Delivered for the accreditation of the degree

Bachelor of Science
Computer Engineering

The report is delivered by

Student number	Name
120249	Kristoffer L. Nesvik
120170	Glenn S. Skjong
120515	Kristian Støylen

Under the supervision of
Kjell Inge Tomren

Faculty of Engineering and Natural Sciences
Norwegian University of Technology and Science
Larsgården 2 - 6009 Ålesund - Norway

Spring semester 2016

Preface

This bachelor project has been conducted for MetaTracks AS, to elucidate the feasibility of a real-time acoustic fingerprinting system. The project was picked because it seemed like the most challenging and rewarding project of the alternatives presented to us. When starting this project, we had no prior knowledge of acoustic fingerprinting or signal processing, theory which is fundamental for the development of such a system.

During the process of creating this system we have acquired a lot of new knowledge. We are thankful for the opportunity to have worked on such an exciting project which has given us a lot of new experiences and insights.

We would like to give a special thanks to our employer Christian A. Strømmen at MetaTracks, for allowing us to carry out this project, and for his good cooperation. We would also direct a thanks to our supervisor Kjell Inge Tomren and Hans Georg Schatuun at NTNU Ålesund for their good input on theoretical and practical matters.

[illegible]
Kristoffer L. Nesvik

Glenn S. Skjong
Glenn S. Skjong

Kristian Støylen
Kristian Støylen

Abstract

Acoustic fingerprinting is a modern technique used for audio recognition, where short excerpts of audio files are used as a compact "signature" for the underlying media content.

In this thesis, we have focused on the design, development and implementation of such a service; an application for mobile devices, capable of querying a database with a large collection of acoustic fingerprints to identify multimedia content in real-time, mainly movies and TV-series.

Research regarding real-time acoustic fingerprinting is limited, and redefining the scope of the study was done several times throughout the project.

Scientific papers on near-real time acoustic fingerprinting, adapted to a real time system, make up the most of the literature used in this thesis, and we base most of our production work on these.

In the information gathering phase of the project, we selectively picked well-reputed papers from several sources, including ScienceDirect and IEEE. The information gathered proved reliable, and have been used in several near-real time acoustic fingerprinting systems previously.

Our finished framework is developed to be robust, and the methods employed are following industry standards. As of now, the finished library come in two flavors; one written for the cross-platform Mono framework, and one written for the Apple Foundation framework. To continue the development of the platform would surely create a good basis for a new research or bachelor project, in an interesting field with a lot of promise.

Contents

List of Figures

List of Tables

Terms

Acoustic fingerprint: A compact, content-based signature that summarizes an audio recording.[9],

Spectrogram: A visual representation of the spectral density of frequencies in a signal, as they vary typically over time.[26]

Wavelet: A brief oscillation around the x-axis. Wavelets can be used to divide signals or other functions into different scale components.[26]

Robust hash: Hashed, semi-unique values, where similar input values generate similar hashes.[12]

Perceptual hash: Hashes generated where similar multimedia input give similar hashes. Related to robust hashes. Typically used in copyright infringement services and digital forensics.[11]

Abbreviations

DLL	Dynamic Link Library
DHWT	Discrete Wavelet Transform
FFT	Fast Fourier Transform
STFT	Short-Time Fourier Transform
API	Application Programming Interface
HAS	Human Auditory System
AWS	Amazon Web Services
SQL	Structured Query Language
GPL	GNU General Public License
LGPL	GNU Lesser General Public License
C#	(C Sharp) programming language
PCM	Pulse Code Modulation
WAV	Waveform Audio File Format
AIFF	Audio Interchange File Format
ISO	International Organization for Standardization
MPEG	Moving Picture Experts Group
LSH	Locality-Sensitive Hashing

Formula

Fast Fourier Transform

$$c_k = \frac{1}{N} \sum_{j=0}^{j=N-1} x_j \omega^{jk}. \tag{1}$$

Jaccard Similarity Coefficient

$$JS(A, B) = \frac{|A \cap B|}{|A \cup B|} \tag{2}$$

N-term Hann Window

$$w[n] = 0.5 + 0.5cos\frac{2\pi n}{N - 1} \tag{3}$$

1 Introduction

Chapter contents

1.1 MetaTracks

The start-up company MetaTracks has been our acting employer for the entirety of this project development. MetaTracks is partially financed by Innovation Norway, and partially by external investors, aiming to create an innovative, web-enabled service which deliver content synchronized with digital media such as movies and TV-series.

MetaTracks have several ongoing development projects, and we as the project group have worked on one of these sub-projects, which is related to acoustic fingerprinting and real-time media recognition. Details surrounding the problem is further described in Section 1.3

1.2 Background

Acoustic fingerprinting in commercial applications is a relatively new concept. One of the most prominent music content identification services, 'Shazam', launched for smart-phones only recently in 2008.[24] Google-owned video-sharing giant 'YouTube' launched their copyright management service 'Content ID' for full in 2012, which utilize digital fingerprinting and watermarking technology to identify copyrighted content in their ever expanding collection.[29]

A robust fingerprinting system targeted at movies and series, without the need for the content meta-data or embedded watermarks in the actual content, is naturally sought after. The use-cases are increasingly many: Television broadcasters can use it for targeted advertisements, movie studios can use it to identify illegitimate copies of their content, and game- and entertainment creators can use it to strengthen their content with for example second-screen technology.

1.3 Problem description

Our project assignment as proposed by MetaTracks was to develop a mobile application framework with an accompanying API (Application Programming Interface), which take advantage of acoustic fingerprinting to identify which movie or series the user of the application is currently watching.

In addition to identifying which movie or series is currently being watched by the user, the application must be able to identify **where** the user is in the current playback, while also taking into account the possibility that the user may have paused, skipped in or stopped the playback entirely.

This type of project is strictly a R&D (Research and Development) project, where the feasibility and/or the final product of the project is unknown.
Exploration of different technologies and methods, repeated testing and even changing the entire project structure and functionality several times is typical in these types of projects.

1.4 Scope

Defining the scope for our project was done in collaboration with the project owner. The main high-level goal for the project was to develop a framework for identifying user position in a multimedia playback. Several questions would have to be addressed throughout the project development for the product to be feasible:

- How should we store very large (in the billions) sets of acoustic fingerprints in an effective manner?
- What would be the most efficient way of generating these acoustic fingerprints? Are there any existing methods with reliable performance benchmarking which could be adapted?
- How do we facilitate for reasonable query times over very large data sets in a real-time system?
- How should we process these very large data sets on devices with limited resources e.g mobile devices and thin clients?
- Which processing tasks should be handled server-side, and which should be handled client-side?
- How do we reduce noise and distortion that occur typical in microphone recordings?

1.5 Thesis structure

In the coming chapters of the thesis, we will address the development questions outlined in Section 1.4 in more detail, and how they were solved technically with regards to the problem description in 1.3

Chapter 2 will introduce theory which is relevant to the product development, which include the background for signal and audio processing, acoustic fingerprinting and related mathematical functions in Section 2.1. Previous research on related fields is explored in section 2.2.

Chapter 3 present the software (3.1), hardware (3.2), frameworks (3.1.3) and platforms (3.1.4) we have used at different stages in the development process. Legal concerns relating to relevant patents and existing systems are covered in Section 3.4.

In chapter 4 we will present the design (4.1), system architecture (4.2) and development process (4.3) of our final product.

Chapter 5 and 6 will cover our thoughts, reflections and the future of the platform, as well as conclusions and final recommendations for the project.

2 Theory

Chapter contents

2.1 Concepts and terminology

2.1.1 Human Auditory System

A typical human can hear sounds at frequencies between 20 Hz and 20 kHz. As will be discussed later, this is vital information relating to perceptually hashing multimedia signals. [5]

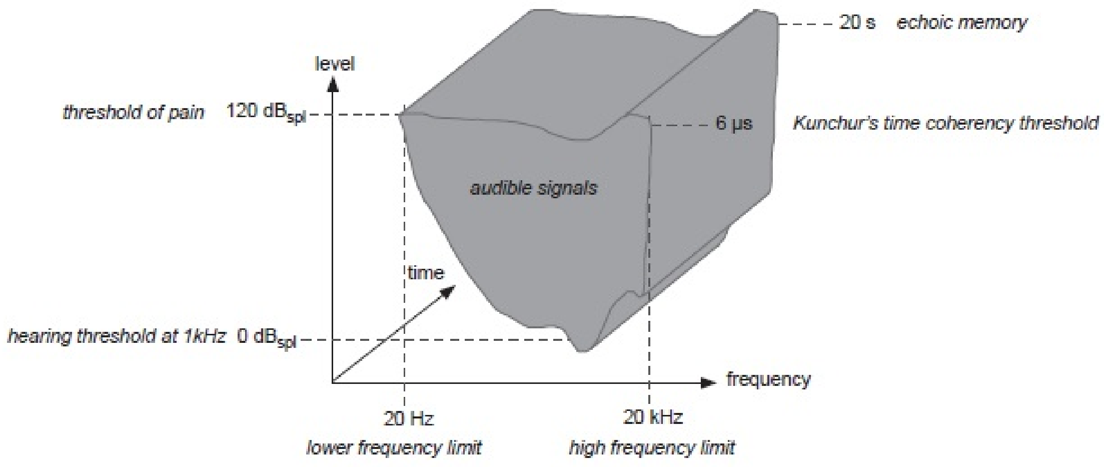


Figure 2.1: Audible signals within the frequencies and time limits of hearing.

Figure 2.1 shows graph of the restrictions of a humans hearing, being limited by the thresholds for sound pressure and frequencies. The grey area shows the region within which frequencies humans can hear audible sounds as well as how long a human can remember detailed audio signals.

2.1.2 Acoustic fingerprinting

Acoustic fingerprinting is a technique where short excerpts of audio files are used as a signature for the actual media content. Typically, the signatures are stored as hashed values in a central database.

One of the most important traits of acoustic fingerprinting is the ability to link *unlabeled* audio playback to their corresponding metadata, either in real-time or near real-time.

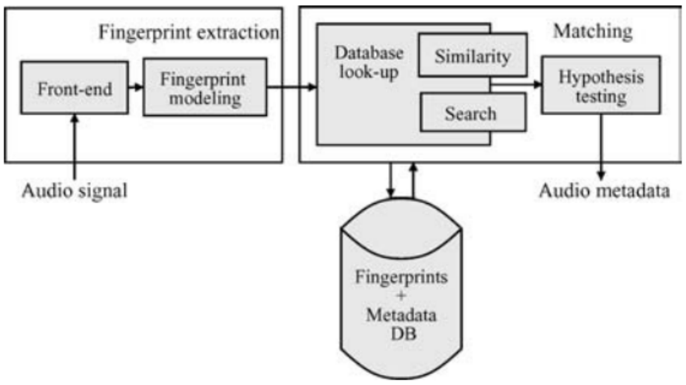


Figure 2.2: A typical audio identification framework.

In figure 2.2 we can see the process of transforming an unlabeled audio signal to it's corresponding metadata through acoustic fingerprint matching, as it is typically implemented by fingerprinting services.

The fingerprints and metadata in the database are compact, quickly queryable and and through robust hashing have a very large discrimination power over other, semi-unique fingerprints.[9]

The hashing techniques which are employed could at first appear trivial, but acoustic fingerprints can not be hashed by traditional cryptographic means, as you would for example a password or other sensitive data. A form of "forgiving" perceptual hash[16][11] must be implemented, so that content which appear similar to the HAS will have similar hashes associated to them.

2.1.3 Acoustic fingerprints versus content watermarking

The following table is taken from Kudrle[15], and show the most prominent differences between an acoustic fingerprint and the traditional way of detecting multimedia similarities which is content watermarking.

	Acoustic fingerprint	Digital watermark
Content remains unchanged	Yes	No
Semi-unique	Yes u	Not necessarily
Can be removed (illegimate)	No	Yes
Used retrospectively	Yes	No
Survives manipulation	Yes	Limited
Compatible with all devices	Yes	No
Prints stored seperately	Yes	No
Proprietary	No	Yes

Table 2.1: Technology comparison of fingerprinting and digital watermarking.

When a movie studio release a movie they often embed watermarks in the playback to identify copyright infringement. This is known as steganography, which effectively change the content to obscure these watermarks. Consequentially, this technique has already been cracked with both hardware and software.[6]

Acoustic fingerprints however, can never be removed from the original media. This makes sense, given that no information is required to be embedded in the original media, as opposed to digital watermarking. As so, no software or hardware can remove the fingerprint without distorting the actual *playback* of the media as it is being played.

Acoustic fingerprinting can be used retrospectively, meaning that even after the release of the movie, across different encoding techniques and territorial differences. Watermarks can not be easily changed after the initial release.

The robustness of acoustic fingerprints makes it persistent to changes, distortion and manipulation, and is compatible with all devices that have recording capabilities.

The fingerprints are usually stored in a separate database for added security, and does not require any agreements with the owners of the intellectual property, this would however be preferable for legal and standardization reasons.[15]

2.1.4 Signals

A signal can be defined as: *The portion of any time-variant signal that falls in the audible range of the human auditory system, capable of invoking a hearing sensation. The signal can be acoustic, electronic or digital.*[5]

Humans perceive signals as variations in pressure at the eardrum and these pressure changes are perceived as sounds, these are analog signals. Analog signals are signals that can take any value from a continuum of values and are defined at every instant of time.[26, 5]

The problem with analog signals is that they can't be processed by computers due to their nature of being indefinite and defined at every instant of time and number of levels. To be able to process signals using computers they have to be sampled and converted to digital signals, usually through an Analog-to-digital converter (ADC). Digital signals can then be processed since they are now defined at only a finite number of levels and points in time.[26]

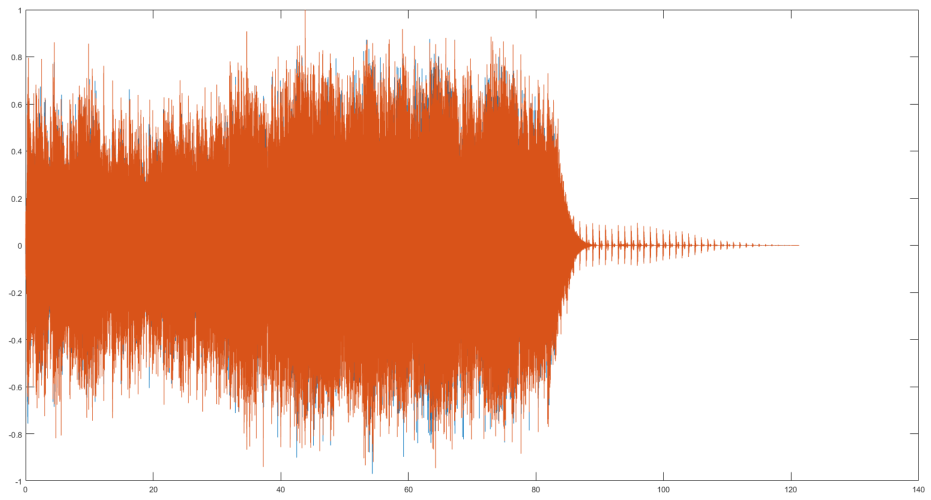


Figure 2.3: A digital signal plot of a song sampled at 44100 Hz.

In the above figure (2.3), we can see the amplitude plot of a digital signal as it's frequency varies over time. The signal is a song sampled at a rate of 44.1 kHz.

2.1.5 Audio File Formats, encoding & compression

An audio file format is a container format for storing audio data on a computer system. There are several different formats of audio and audio codecs, but they can be divided into three distinct, basic groups: uncompressed audio file formats, lossless compression audio formats and lossy compression audio file formats.[3]

The most widely used and known uncompressed audio file format is Pulse-code Modulation (PCM). Windows and Mac use WAV and AIFF respectively as their standard PCM formats. WAV and AIFF are flexible file formats, designed to store more or less any combination of sampling rates or bit-rates. This makes it a suitable file format for storing and archiving original audio recordings, also known as master records.[3]

Format	Compression	Sample rate (max)	Bit rate (max)	Storage requirement
WAV	None	192000	9216	Very high
MP3	Lossy	48000	320	Medium
AAC	Lossy	48000	320	Medium
AIFF	None	192000	9216	Very high
FLAC	Lossless	655350	1411	High

Table 2.2: Comparison of the most used compression methods.

In table 2.2 we have listed the most common audio file formats. The three most commonly used file formats are "Wave" files (.wav), MPEG Layer-3 files (.mp3) and the Advanced Audio Coding format (AAC), which are described below.

- **WAV** - Standard audio file format used mainly on Windows systems. Commonly used for storing uncompressed, CD-quality sound files, which means that they can be large in size - around 10 MB per minute of music. It is less known that Wave files can also be encoded with a variety of codecs to reduce the file size.[4]

- **MP3** - The MPEG Layer-3 format is the most popular format for downloading and storing music. By eliminating portions of the audio file that are essentially inaudible to the Human Auditory System, mp3 files are compressed roughly one-tenth the size of an equivalent PCM file, while maintaining a good perceivable audio quality.[4]

- **AAC** - Advanced Audio Coding (AAC) is an audio coding standard for lossy digital audio compression. ACC has been standardized by ISO (International

Organization for Standardization) and IEC as part of the MPEG-2 and MPEG-4 specifications.[2]

2.1.6 Nyquist-Shannon Sampling Theorem

The Nyquist-Shannon Sampling Theorem (NSST) establishes that a signal can be recreated in full, as long as the target sampling rate is chosen so that it is sampled two times the maximum frequency of the signal.

The minimum sampling rate following the NSST is defined as the Nyquist sampling rate. For example, a signal containing frequencies up to 25000 Hz must be sampled at a minimum of 50000 times per second. In this case, 50000 is the Nyquist sampling rate regardless of the sampling rate applied being higher than 50000 times per second.

Following the NSST we can also define what is known as the Nyquist frequency and the Nyquist range. The Nyquist frequency is half the sampling rate of the system. In the same 25000 times per second system as mentioned earlier, the Nyquist frequency is 12500 Hz.

The Nyquist range is defined as the range of frequencies between zero and the Nyquist frequency, in the same system it spans from 0 Hz up to and including 12500 Hz.[26, 5]

2.1.7 Sample-rate conversion

Sample-rate conversion, informally called re-sampling, is a process which converts the sample rate of an audio signal to another sampling rate.

Audio signals can be resampled for a number of reasons, such as reducing the file size for space efficient storage on other media such as mastered DVD's, or reducing the processor usage when processing the samples. Reducing the sampling rate effectively distorts the original audio signal, the information will however remain audible by the HAS down to a sampling rate of about 5000 Hz.[25, 26]

2.1.8 Fast Fourier Transformation

The Fast Fourier Transform is defined as the following by Vegte[26]: *The Fast Fourier Transform (FFT) is a mathematical function that takes the description of a signal or filter from the time domain and transforms it, typically into the frequency domain.* The FFT provides a practical means of identifying a signal's spectrum.

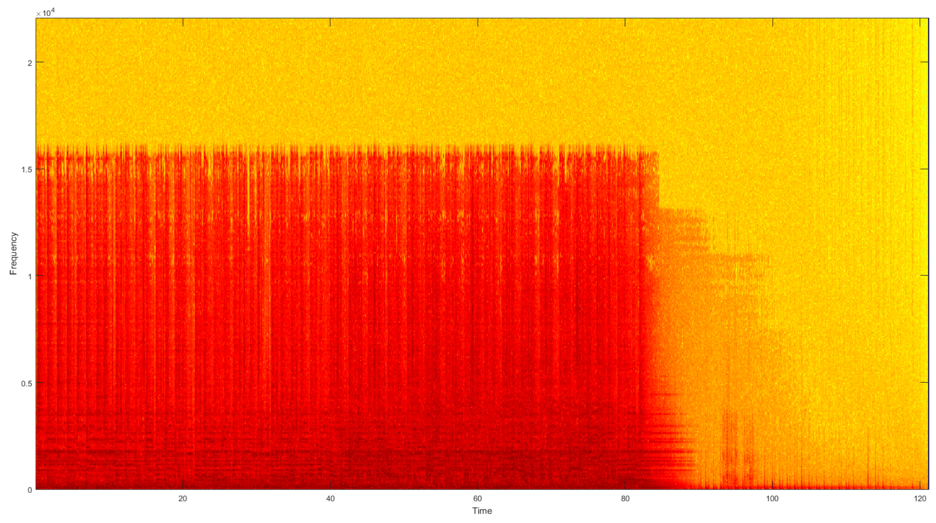


Figure 2.4: Spectrogram of a song sampled at 44.1 kHz.

In the above figure (2.4), we can see the Time-Frequency spectrogram plot of a digital signal generated in MATLAB.

Signal spectrums such as the one shown above act as the entry point for many digital signal processing tasks, such as image and radio processing and differential equations, to name a few.

The FFT, as it's name suggest, is a faster implementation of the traditional Discrete Fourier Transform (DFT). The FFT effectively reduces the computing complexity of DFT from $\mathcal{O}(n^2)$ to $\mathcal{O}(n \log n)$, where n is the size of the data.

2.1.9 Haar Wavelet

Wavelets are functions that must satisfy certain set requirements. A wavelet should always integrate to zero, briefly oscillating the x-axis. There exists many variants of wavelets and wavelet families, the oldest and simplest of all wavelets being the ”Haar Wavelet”. The Haar Wavelet is a step function taking values 1 and -1, on $[0, \frac{1}{2})\ and\ [\frac{1}{2}, 1)$, respectively. [27] The graph of the Haar Wavelet is given in Figure 2.5

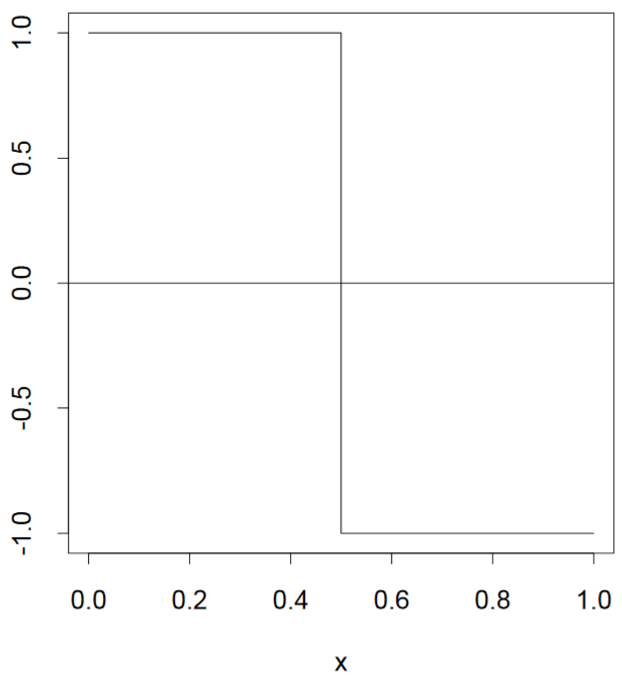


Figure 2.5: Graph showing a Haar Wavelet.

The Haar wavelet is the simplest of all wavelets, as it is not continuous and therefore not differentiable. This is an advantage in acoustic fingerprinting scenarios, where input signals are unpredictable and sudden.[27]

2.1.10 Discrete Wavelet Transform

A Discrete Wavelet transform (DWT) makes it possible to efficiently downsize an image, while still retaining the original data. The file size will effectively be just a fraction of the original file size, depending on the compression ratio. This makes the image more suitable for storage and quick transmission.

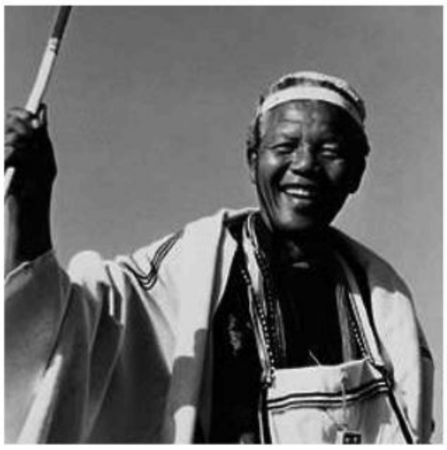


Figure 2.6: An original grayscale image which will undergo a DWT.

Even though the size of files undergone a DWT is small, it can be losslessly approximated back to it's original form from e.g., a 200:1 compression ratio to the original 1:1 compression ratio.

A DWT can use either "averaging and differencing" or "normalization". Normalization is by far the superior approach, as it can approximate to the original photo at a much better compression ratio and detail, over averaging and differencing.[17]

The original picture in Figure 2.6 will in the following chapter be shown as it has undergone a DWT using normalization.

Figure 2.7 shows the original picture in Figure 2.6, undergone several Haar wavelet transforms using normalization. The standard averaging and differencing techniques involves repeated processing of certain pairs (a,b) to yield new pairs $(\frac{a+b}{2}, \frac{a-b}{2})$. In normalization on the other hand, we process each such pair (a,b) to yield $(\frac{a+b}{\sqrt{2}}, \frac{a-b}{\sqrt{2}})$ instead. [17]

This will yield the results shown in figure 2.7, where the compression ratios are 200:1, 100:1, 50:1, 25:1, 10:1 and 2:1 respectively.

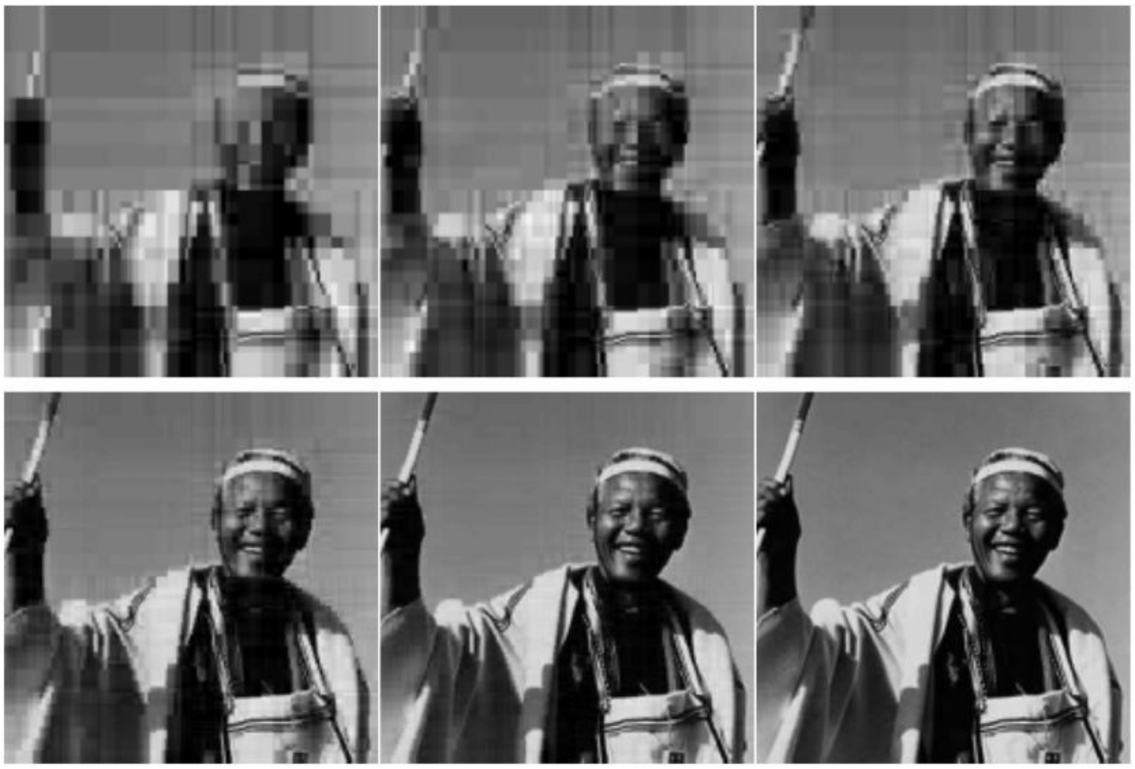


Figure 2.7: The original grayscale picture gone though several DWT's at different compression ratios.

Note that the approximation of the last picture which is compressed with a 2:1 ratio is so similar to the original picture in 2.6 that it's almost indistinguishable, while the actual file size is substantially smaller.

2.1.11 Jaccard Similarity Coefficient

The Jaccard Similarity Coefficient is a statistic used for comparing the similarity and diversity of a sample of sets. The Jaccard Coefficient measures similarity between finite sample sets, and is defined as the size of the intersection divided by the size of the union of the sample sets.[14]

The Jaccard Similarity Coefficient is defined as follows:

$$JS(A,B) = \frac{|A \cap B|}{|A \cup B|} \tag{2.1}$$

Considering two sets A = {0, 1, 2, 5, 6} and B = {0, 2, 3, 5, 7, 9}. We can find the similarity using the following function:

$$JS(A,B) = \tfrac{|A \cap B|}{|A \cup B|} = \tfrac{|\{0,2,5\}|}{|\{0,2,3,5,7,9\}|} = \tfrac{3}{8} = 0.375$$

More notation, given a set A, the cardinality of A denoted |A| counts how many elements are in A. The intersection between two sets A and B is denoted $A \cap B$ and reveals all items which are in both sets. The union between two sets A and B is denoted $A \cup B$ and reveals all items which are in either set. [18]

2.1.12 Robust hashing (Min-Hash and LSH)

Min-Hash is a hashing algorithm used to find the similarity between sets of data. It is commonly used in finding similarities in corpora (large collections of written works).[10]

Min-Hash measures the similarity between sets of data using the Jaccard Coefficient. However, when calculating the similarity between documents the document can't simply be broken into individual words like they are a set of numbers.[19, 20]

A document would be broken down into something called shingles. Each shingle containing a set number of words, and a document is broken down into **total words - shingle length + 1 number of shingles**. This means that if a document contained the single sentence "The quick brown fox jumps over the lazy dog", it would be broken down into the following five word long shingles[10]:

1. The quick brown fox jumps
2. quick brown fox jumps over
3. brown fox jumps over the
4. fox jumps over the lazy
5. jumps over the lazy dog

Resulting in a set looking like the following:
Set A = {The quick brown fox jumps, quick brown fox jumps over, brown fox jumps over the, fox jumps over the lazy, jumps over the lazy dog}

These sets of shingles can then be compared for similarity using the Jaccard Coefficient. There is now a way to compare two documents for similarity, but it is not an efficient process. Each pair would have to be compared individually, which is not a scalable process. Instead of comparing each document individually, the random shingles from two documents can be compared. For example, a document that is 10000 words long, would be broken down into 9996 shingles, randomly selecting a subset of 200 shingles to represent the entire document. A second document could be 20000 words long, being broken down into 19996 shingles, randomly selecting a subset of 200 shingles from that as well. To adjust for the workload, the accuracy is reduced. [20, 10]

LSH (Locality-sensitive hashing) focus on pairs of signatures that are *likely* to be similar. In other words, LSH hashes the items several times, in such a way that similar items are more likely to be hashed to the same "bins" than dissimilar items are.

In difference to cryptographic hashing, LSH attempts to **maximize** the rate of collision for items which appear similar. For the case of acoustic fingerprinting, this would be collisions between audio which appear similar to the HAS.[20]

Figure 2.8 shows the probability of a Min-Hashed set becoming a candidate for a perceptual LSH bin, as a function of the Jaccard similarity described in Section 2.1.11.

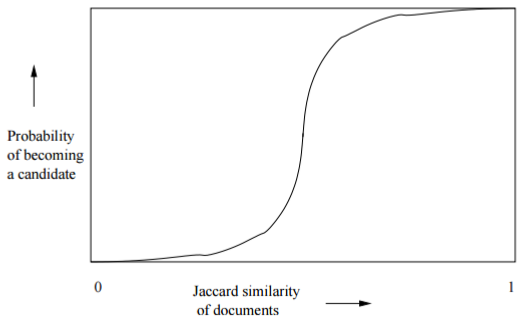


Figure 2.8: Candidature probability as a function of the Jaccard similarity of the hash.

Following the theory surrounding Min-Hash and LSH, we use the proposed definition of a finalized robust hash as follows:

A robust audio hash is a function that associates to every basic time-unit of audio content a short semi-unique bit-sequence that is continuous with respect to content similarity as perceived by the HAS.[12]

2.1.13 Binary Search

Binary search is an efficient searching algorithm, requiring a sorted, indexed list to work.

The binary search works by comparing the element to be located to the middle term of the list. The list is then split into two smaller sublists of the same size, or where one of these smaller lists has one fewer term than the other. The search continues by restricting the search to the appropriate sublist based on the comparison of the element to be located and the middle term.[23]

An example for how this works could be:
To search for 19 in the list

$$1, 2, 5, 9, 12, 15, 19, 21, 25, 26$$

First, we split the list which has 9 terms into two smaller lists with five terms each

$$1, 2, 5, 9, 12 - 15, 19, 21, 25, 26$$

We then compare 19 and the largest term in the first list. Because 19 is bigger than 12, we can conclude 19 is in the second list. Next, split this list, which has five terms, into the two smaller lists of two and three terms respectively.

$$15, 19 - 21, 25, 26$$

Compare 19 with the largest term in first list, and we have found 19.

2.1.14 Perceptual hashing

Perceptual hashing is another robust hashing method that is used in various different fingerprinting applications.

Perceptual hashing, unlike conventional hashing used in cryptography, represents a unique binary string or code that uniquely identifies a segment of audio content (such as music) similar to personal fingerprints used to identify human beings.[11]

2.1.15 Hann Window

The Hann function is a window function, typically used in digital signal processing to select a subset of a series of samples in order to perform a Fourier transform or other calculations.[13]

The Hann function is applied to each overlapping frame in a signal, effectively minimizing frame discontinuities. The N-term Hann windows is defined by the following equation:

$$w[n] = 0.5 + 0.5cos\frac{2\pi n}{N-1} \tag{2.2}$$

2.1.16 Time Complexity

Time complexity is concerned on how fast or slow an algorithm performs its task. The complexity is defined as a numerical function T(n), time versus input. Big O notation is a symbolism used to describe the asymptotic behaviour of a function, meaning how fast or slow it grows or declines.[8, 1]

An example of a time function T(n), written with Big O notation to express an algorithm's runtime complexity, looks like this:

T(n) = O(n2)

2.2 Previous Research

The most thorough proposal for a robust, perceptual audio fingerprinting system is presented in Haitsma, Kalker & Ostveen[12], and the method presented here would eventually form the basis for most acoustic fingerprinting services that are used today. Industry leaders in acoustic fingerprinting services, such as the music recognition service 'Shazam' which use a Landmark-based implementation[28] base its core algorithm on findings and proposals by Haitsma, Kalker & Ostveen. The algorithm used in MetaTracks is loosely based on this proposal, as well.

The main inspiration for the MetaTracks fingerprinting system, however, comes from Bajula & Covell[7], where several disciplines are used to create what is described as a 'novel' method for audio identification. Combining computer-vision and data stream processing, compact, robust and perceptually hashed fingerprints are created to facilitate efficient matching of a real-time audio input, as opposed to a near real-time solution.

Mathieu & Geoffroy[21] present a complete system, 'AudioPrint', which identify advertisement trends in radio broadcasts.

Related research on **real-time** audio recognition is scarce, and we did not succeed in finding any proof-of-concept on how it could be carried out in practice. Mufin GmbH (nee magix gmbH) deliver a similar product.

2.3 Summary

To fulfill the requirements and scope of this project we have employed a lot of theoretical research and mathematical methods, we have included most of the concepts and terminology that we have used to create the finished product in this theory chapter. We have described and explained each of the steps in the process uses to create a fingerprint and comparing it against the stored fingerprints, in our database.

These concepts allows us to take a signal, either from a pure digital source or in real-time and create acoustic fingerprints. In creating these fingerprints we use re-sampling, Hann Window, FFT, DHWT and Min-Hash, following that we use the concept of Jaccard similarity and LSH to compare the fingerprints we create client-side, to the ones stored in the central database.

3 Method and materials

Chapter contents

3.1 Software

3.1.1 Integrated Development Environments

For the duration of the project several Integrated Development Environments (IDE's) have been used to ease and improve the software development process. With built-in source code management and version control features, professional IDE's facilitate productivity and collaboration within the project group.

3.1.1.1 Microsoft Visual Studio

Microsoft Visual Studio is an IDE with a large collection of tools and services to help create applications for the Microsoft platform. Several programming languages are supported, such as Visual Basic, Visual C++, Visual C#. With a large set of plugins available, more languages, frameworks and platforms can be supported natively. Visual Studio has for the most part of the project been our IDE of choice.

3.1.1.2 XCode

Xcode is Apple's integrated Development Environment containing a suite of software development tools used for development for OS X, iOS, WatchOS and tvOS. Xcode can compile C, C++, Objective C++, Objective C and Swift, all supported languages for the mentioned platforms. We have used Xcode because it is the sole compiler for Swift.

3.1.1.3 Xamarin Studio

Xamarin Studio is a standalone IDE developed by the Microsoft-owned company 'Xamarin'. In our project, Xamarin Studio was mainly used to properly deploy our .NET application to Apple-products, using the Mono framework.

3.1.1.4 MathWorks' MATLAB

MATLAB (Matrix Laboratory) is a C-based computing suite, focusing on advanced matrix manipulation and scientific computing. With a robust signal processing toolbox, MATLAB helped us throughout the development to learn about and analyze audio signals.

3.1.2 Collaboration tools

To efficiently collaborate on software development, effective version control and project management is important. Several tools were used during the development process, which is described below.

3.1.2.1 GitHub (Private repository)

GitHub is an online Git repository with integrated revision control and source code management. GitHub is the de-facto standard for source code management, both in professional and hobbyist software development projects, and also our version control platform of choice.

3.1.2.2 Atlassian JIRA

Atlassian JIRA is a professional project management platform which include bug-tracking and issue tracking, facilitating for proper usage of Scrum.

3.1.2.3 ShareLaTeX

ShareLaTeX is an online tool where project groups can collaborate on documents written in the LaTeX markup language. Widely used for publication of scientific documents, and also the language of choice to format this thesis.

3.1.3 Frameworks

Frameworks and API's (Application Programming Interfaces) played a vital role in our project, as its not preferrable to recreate for example a digital signal processing suite, when one is readily available and can be licensed.

3.1.3.1 .NET

.NET is an open-source framework developed by Microsoft providing a comprehensive programming model for building a wide range of applications, from mobile and desktop to web. Programs written for .NET Framework execute in a software environment known as Common Language Runtime (CLR), an application virtual machine that provides services such as security, memory management, and exception handling.

3.1.3.2 BASS by Un4seen Development

BASS is an industry-grade cross-platform, multi-language audio library. BASS' purpose is to streamline the process of capturing and processing (signal processing) audio streams. With support for a multitude of platforms, BASS turned out to be the only library which met all the demands for our project. BASS is free for non-commercial use, but can be licensed for commercial use, described in Appendix C.

3.1.3.3 BASS.NET

BASS.NET is a .NET language wrapper for the BASS library mentioned above. BASS.NET also provide libraries for Android, iOS and ARM Linux platforms, amongst others. Just as BASS, BASS.NET must also be licensed before commercial usage.

3.1.3.4 Xamarin.iOS

Xamarin.iOS is a platform which makes it possible to create native iOS applications with the flexible C# programming language. Was used earlier on in the project to allow our C# code to run seamlessly on iOS devices, however later on it was revealed that while Xamarin.iOS will run fine on iOS devices as a standalone application, it would not be able to be integrated with our employers platform as Xamarin.iOS projects will only be able to go hand in hand with other Xamarin or C# projects. Because of this we had to rewrite the code base from C# to Swift, so discarding Xamarin.iOS from the final product.

3.1.3.5 NAudio

NAudio is an open source .NET library for audio processing. While the library was thoroughly used in the early development phases, it had some limitations in regards to support and digital signal processing, and was mostly replaced by BASS and BASS.NET in the final product.

3.1.3.6 Exocortex.DSP

Exocortex.DSP is an open source C# library that provides fully featured single and double precision complex number data types, complex number math libraries, 1D, 2D, and 3D complex and real symmetric fast fourier transforms, and highly accurate statistical routines. This library allows us to use FFT methods with a specified window.

3.1.4 Deployment platforms

To deploy our Web API, we used several platforms that support on-the-fly deployment of Visual Studio Web API projects.

3.1.4.1 Amazon Web Services

Amazon Web Services (AWS) is a modern collection of 'cloud computing' services, offering flexible pay-as-you-go storage, databases and hosting among a multitude of other services.

3.1.4.2 Amazon Relational Database Service

One of the most important services offered by AWS for our project, Amazon RDS is a high-performance relational database web service running 'in the cloud'. During the project work we quickly saw the need for several databases, mainly for storing acoustic fingerprints, storing Web API credentials and more.

3.1.4.3 Amazon DynamoDB

Amazon DynamoDB is a proprietary NoSQL database service also offered through AWS. The main difference between DynamoDB and RDS is the market model, where developers pay for throughput rather than storage. DynamoDB was used for a large portion of our development project, but in the end we migrated the service to Amazon RDS due to speed limitations and the need for a relational database.

3.1.4.4 MySQL

MySQL is an open-source relational SQL-based database management system. MySQL is often used in large web applications, business and e-commerce.

3.2 Hardware

MetaTracks provided us, for the duration of the project, a MacBook Pro (2015) and iPad Pro (2015) for testing and deployment purposes. It was important that our application ran properly on a wide range of Apple devices, which hold a large share of handheld devices on the market.

3.2.1 MacBook Pro 2015

Platform: OSX 10.11 (upgradeable)
Chipset: Intel
CPU: Dual-core 2.9GHz
GPU: Iris Graphics 6100
Main Memory: 8GB RAM

3.2.2 Apple iPhone 6S

Platform: iOS 9 (upgradeable)
Chipset: Apple A9
CPU: Dual-core 1.84 GHz
GPU: PowerVR GT7600
Main memory: 2GB RAM

3.2.3 Apple iPad Pro (2015)

Platform: iOS 9 (upgradeable)
Chipset: Apple A9X
CPU: Dual-core 2.26 GHz
GPU: PowerVR Series 7
Main memory: 4GB RAM

3.3 Acoustic Fingerprinting Framework

A shared library is used for storing all the methods and classes related to acoustic fingerprinting. This is done because multiple projects need to use the same fingerprinting methods and instead of rewriting code multiple times, we just use a library to share between them all. The algorithm used in the acoustic fingerprint generation is created by Google researchers, Shumeet Baluja and Michele Covell - we had to interpret this research and write it in source code.

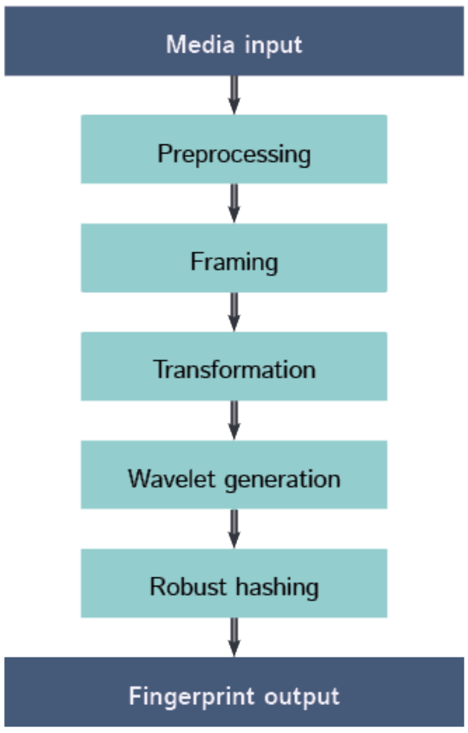


Figure 3.1: Conceptual overview of the acoustic fingerprinting process.

As seen in figure 3.1, the transformation from a media input to a finished acoustic fingerprint consist of five phases: Preprocessing or decoding, framing, transformation, wavelet generation and robust hashing. The individual workings of these phases are further explained in the coming sections.

3.3.1 Decoding and Resampling

The entry point of the acoustic fingerprinting algorithm is the digital decoding and downsampling of audio files into a processable PCM stream.

To achieve this, the third-party ”BASS” audio library is used. In practice, this process involves six distinct steps:

1. BASS is initialized, using the default device driver.
2. A compressed audio file is staged in BASS.
3. A file stream is initialized, which is holding the original signal data.
4. A mixing stream is initialized, which will hold the downsampled signal data.
5. The original signal data is decoded, downsampled to 5512 Hertz, while also having it's channel data averaged (Surround channels to mono).
6. The resulting sample data is returned, which is in effect an array of 32-bit floating point values.

In Figure 3.2 we can see the audio signal before and after this process. As is apparent, the differences are small. When audio is downsampled, the signal is brought from one frequency range to another, without removing the distinguishable audio or manipulating the timeframe.[26]

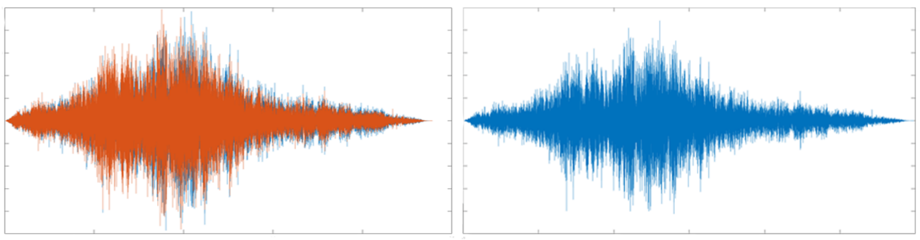


Figure 3.2: Downsampled audio. Left: Original audio signal. Right: Downsampled audio signal. *Figure made in MATLAB.*

3.3.2 Spectrogram

Having a processable PCM stream, we can initialize the creation of the spectrogram. The spectrogram is contained with overlapping "frames" created by a Hann Function, each frame being run through an FFT function provided by the Exocortex DSP library.

Overlapping frames are implemented to account for the spectrogram slowly changing over time. This will be important at later stages, providing matching robustness to position uncertainty. When the spectrograms has been generated, we are left with a two-dimensional array of values containing the two axes of the individual spectrograms, higher values means higher intensity.

Figure 3.3 shows a spectrogram generated by our acoustic fingerprinting library. On the y-axis of the plot we have the signal frequency. On the x-axis, the time is shown. The light gray to white color range shown on the z-axis refer to the intensity at a given time-frequency point.

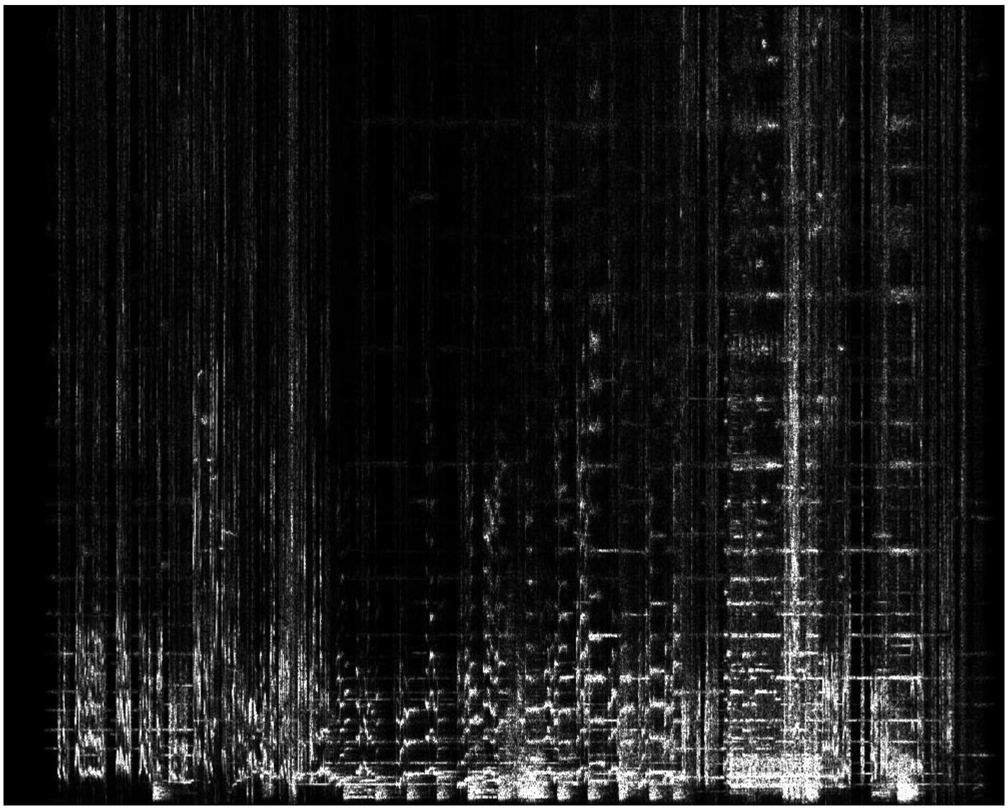


Figure 3.3: Visualization of a frequency spectrogram, generated by the Acoustic Fingerprinting Library.

3.3.3 Wavelets

The generated spectrogram arrays will in turn be split into 32 logarithmically spaced frequency bins, where the frequencies range from 318-2000 Hz. In effect, this will generate wavelets with a length of 190 milliseconds, having 32 bins with the values from each frequency range from the full spectrogram. This processing compresses the image greatly, making it easier to store in a central database while also making it more robust to noise and distortion.

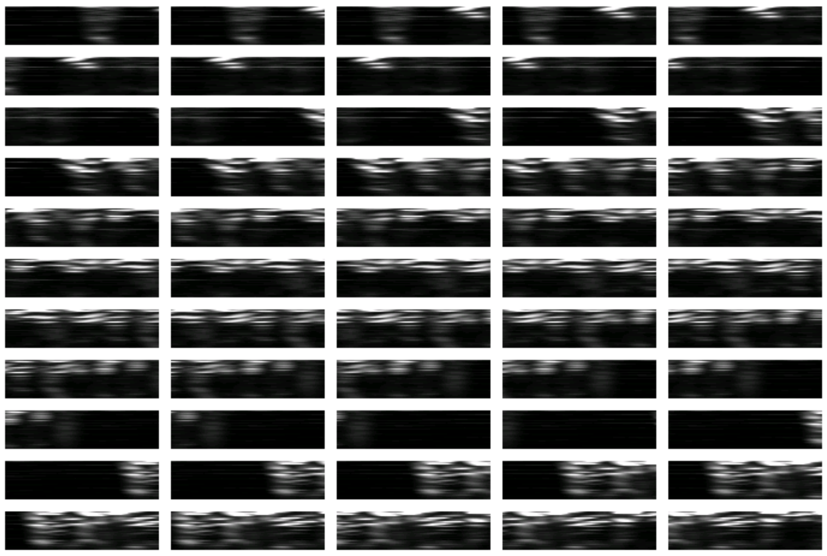


Figure 3.4: Visualization of wavelets, generated by the Acoustic Fingerprinting Library

In figure 3.4 we can see the wavelets that are generated from different high-intensity parts of the spectrogram. This also illustrates how each image changes slightly over time thanks to the overlapping used in the spectrogram. Each individual image will in turn sent through a Haar Wavelet Transform, compressing them further while still retaining the most important, high-intensity features of the wavelet images.

3.3.4 Fingerprinting

For the creation of actual acoustic fingerprints, we extract the top 200 energy points from our generated wavelets, saving them as 8192 binary values in an acoustic signature, which is shown in Table 3.1. An energy point is calculated using absolute values in a descending order, so we will have a mix between negative and positive numbers. If one of the 200 top energy points is a negative value, it will translate to ”false”, while a positive value will translate to ”true”. The 8192 resulting Booleans make up the final acoustic fingerprint, after this we will begin hashing.

0	true
1	false
2	true
3	true
4	false
5	true
6	false
.....	
8190	false
8191	false

Table 3.1: A sample acoustic fingerprint signature.

The signatures, with their accompanying time stamps, will be added to a Fingerprint object. A large list containing all the Fingerprint objects are created, making up the different fingerprints for the entire multimedia file. The different fingerprint objects will contain values comparable to the data shown in table 3.2. A visual representation of the finalized acoustic fingerprints is shown in figure 3.5.

SignatureNumber	14
Signature	{bool[8192]}
Timestamp	1.792

Table 3.2: Sample information on an acoustic fingerprint.

Figure 3.5: Visualization of acoustic fingerprints, generated by the Acoustic Fingerprinting Library.

3.3.5 Hashing

The final step of the process is to robustly hash the fingerprints. If we were to use cryptographic hashing methods, they would give completely different results with even the slightest change in bits, which is not preferable. A robust hashing method gives similar fingerprints similar hashes, is needed. This robust hashing will be done through the use of Min Hash and LSH grouping.

Following this we will take the Min Hash values acquired and group them together using LSH grouping. This is done to reduce the amount of hashes we get per fingerprint. A collection of hashes is formatted as in table 3.3.

0	87931
1	63800
2	10635
3	78529
4	52508
5	88291
6	11127
.....	
31	27700
32	4138

Table 3.3: A sample of acoustic fingerprint hashes.

For the LSH grouping, 33:3 key/value pairs are created. In effect, this means that a single fingerprint will have 33 different hashes representing it, this allows us to use jaccard similarity to see how similar they are. These are the hashes that will be used in the retrieval process.

3.3.6 Acoustic fingerprint retrieval

In Figure 3.6 we can see the retrieval process of acoustic fingerprints stored in the central database.

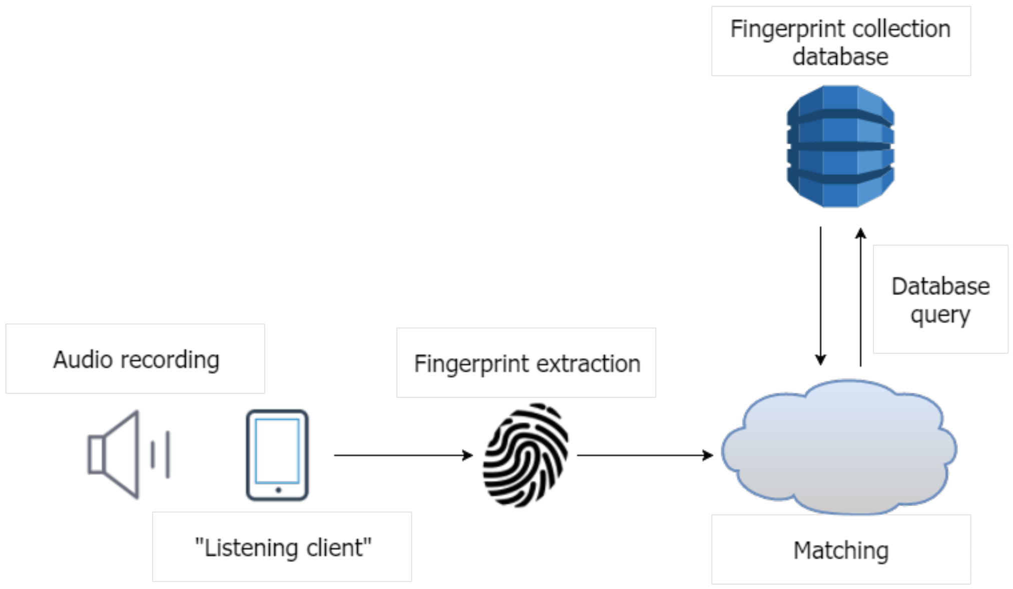


Figure 3.6: Overview of the acoustic fingerprint retrieval and comparison process.

Two arrays of data, one containing the acoustic fingerprints and one containing the related timestamps, are downloaded to the mobile ”listening” client. The timestamps are retrieved by a single HTTP request from the client to the Web API. The Web API will query the fingerprint collection database, and return the fingerprints to the client.

Once recording is initialized on the mobile device,, the downloaded list of fingerprints will be compared to audio recorded using the embedded microphone on the device. The comparison process will compare three seconds of buffered microphone audio for each iteration.

3.3.7 Fingerprint matching

The fingerprint matching process include two distinct parts: one is a full search, and the other part is a partial search. The first time a comparison happens the full search will be run to find out rouglhy where the user is located in the movie. After we have an estimated timestamp, we can perform partial, expanding searches from then on for quicker and more efficient searching.

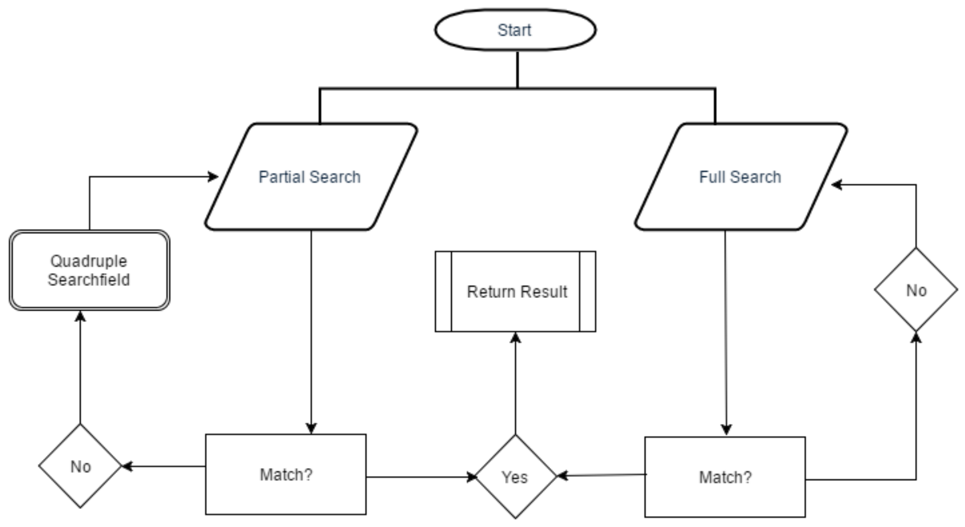


Figure 3.7: Overview of the recognition process.

If the partial search is being used, future searches will be in a specified search-field of 15 seconds around the time stamp of the last matched fingerprint. For example, a match at 3 minutes and 15 seconds will result in further searches between 3 minutes and 3 minutes and 30 seconds. If a match is not found in that search-field, the search window will be doubled for each consecutive "miss". This is done so look-up times will be significantly faster than if we were to keep searching through the entire movie every three seconds, and to reduce chances for missed fingerprints or false positives.

The actual search is carried out by iterating through every fingerprint of both lists, using a binary search to quickly compare how many equal hashes the fingerprints has. One equal hash translated to one "vote" as shown in Table 3.3.7.

List 1	List 2
81256	14612
78212	81256
6271	46222
85946	78212
14612	94359
42976	27648
56127	5162

Table 3.4: Example of an acoustic fingerprinting voting scenario.

If two fingerprints has four or more votes (out of 33 possible), it is considered a potential match. More votes means a stronger match. If two or more fingerprints have four or more votes, the fingerprint with the highest amount of votes will be selected to be returned and shown on iOS device.

The full acoustic fingerprinting matching has a complexity of $\mathcal{O}(n^2 \log n)$. $\mathcal{O}(n^2)$ for the initial search, and $\mathcal{O}(\log n)$ for the binary search of hash bins.

3.3.8 Noise filtering

The entire process of creating fingerprints is about compressing the audio signal as much as possible while still keeping the relevant information needed for audio to be unique. Doing this will make the fingerprinting process resistant to noise and bad quality.

We are not using any noise filtering during the recording process, only in the fingerprinting process itself. In the entire fingerprinting process we use four different methods to compress audio while still keeping the most relevant pieces of information present.

These are reducing sample rate of audio down to 5512 samples per second, using a overlap during the spectrogram generation so the information is slowly being changed over time, perform the haar wavelet transform to remove most of the unnecessary information in the audio signal while still keeping the most important parts, and finally we extract only the 200 most energetic points for each fingerprint.

This process helps our fingerprinting algorithm to be resistant to noise or bad quality audio.

3.4 Legal

Several of the systems and methods for acoustic fingerprinting have been previously patented, and part of the initial research consisted of mapping the different patents that could have a relevance to, and potentially halt or terminate the development. Especially for a start-up company, a potential copyright infringement case could threaten the life or profitability of the company should it escalate, so naturally this was a scenario we wanted to avoid.

Some of the most relevant patents include:

Patent title	Office	Identification
Audio fngerprinting system and method	US	7013301
Audio fingerprint for content identification	US	8949872
System and method for recognizing audio pieces via audio fingerprinting	US	7487180
Device, method, and medium for generating audio fingerprint and retrieving audio data	US	8380518
Apparatus and method for generating an audio fingerprint and using a two-stage query	US	8886531
Automatic identification of sound recordings	US	7881931
Robust and invariant audio pattern matching	US	20090265174

Table 3.5: Patents relevant to MetaTracks' acoustic fingerprinting algorithm.

4 Results

Chapter contents

4.1 Design

The final product created by the project group consists of five discinct parts or subsystems: namely the 'Acoustic Fingerprinting Library', the 'Database Population Application', an iOS application for mobile devices, a Windows application for desktop, as well as a Web API. All of these are further discussed in the following sections.

4.1.1 Acoustic Fingerprinting Library

The Acoustic Fingerprinting Library is the dynamic shared library of our implementation, currently in use by both the Database Population Application and the iOS Application. This library contain the algorithm for acoustic fingerprint generation, as well as the routine for recognition and matching of input fingerprints. The library can act as the back-end for any front-end application, be it mobile or desktop.

To function properly, the Acoustic Fingerprinting Library requires a device driver with audio recording capabilities. The actual front-end implementation and access control to this device driver is irrelevant to the framework, as long as the delivered payload is a 32-bit floating point stream.

The library do have some subtle implementation differences depending on the target platform, to account for different orders of operation on for example Windows and OS X systems. This is further described in the system documentation (Appendix C).

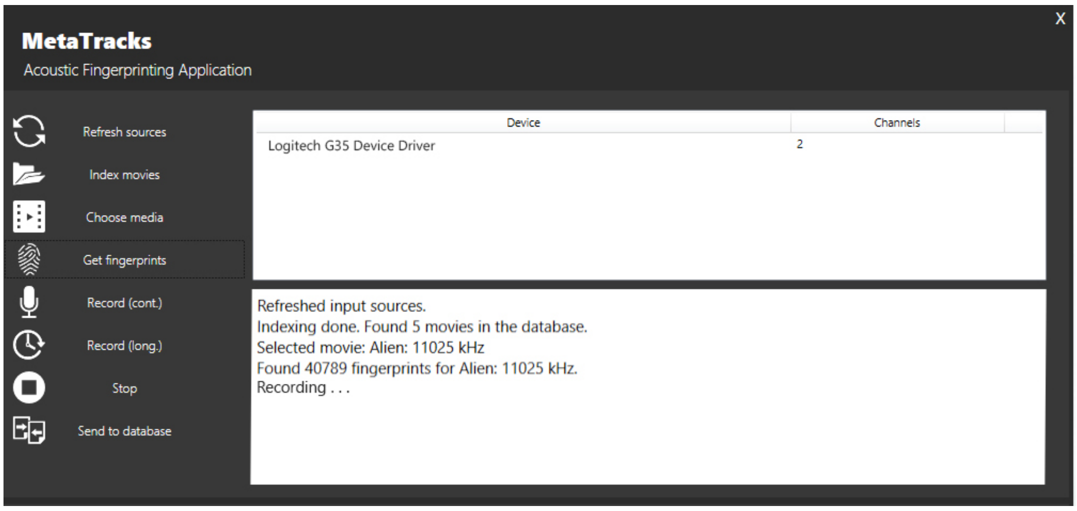
4.1.2 Windows Application

The entry point of the Windows application is the user of the application selecting an input source from the provided list of connected device drivers (typically the on-board microphone).

A HTTP GET is sent to the Web API, returning the movies that are currently present in the acoustic fingerprinting database. Once a movie has been picked, the user can retrieve the acoustic fingerprints corresponding to the movie by issuing a second HTTP request.

When the fingerprint callback is finished, the recording can be initiated by the user-initiating the input stream from the device driver.

As the input stream is running and recording, a background thread is continuously comparing excerpts of the input stream with the fingerprints collected from the database. When the fingerprint matching subroutine finds a match, the matching fingerprint is returned to the user in the form of a timestamp.



Device	Channels
Logitech G35 Device Driver	2

```
Refreshed input sources.
Indexing done. Found 5 movies in the database.
Selected movie: Alien: 11025 kHz
Found 40789 fingerprints for Alien: 11025 kHz.
Recording . . .
```

Figure 4.1: Finished Windows application for media recognition.

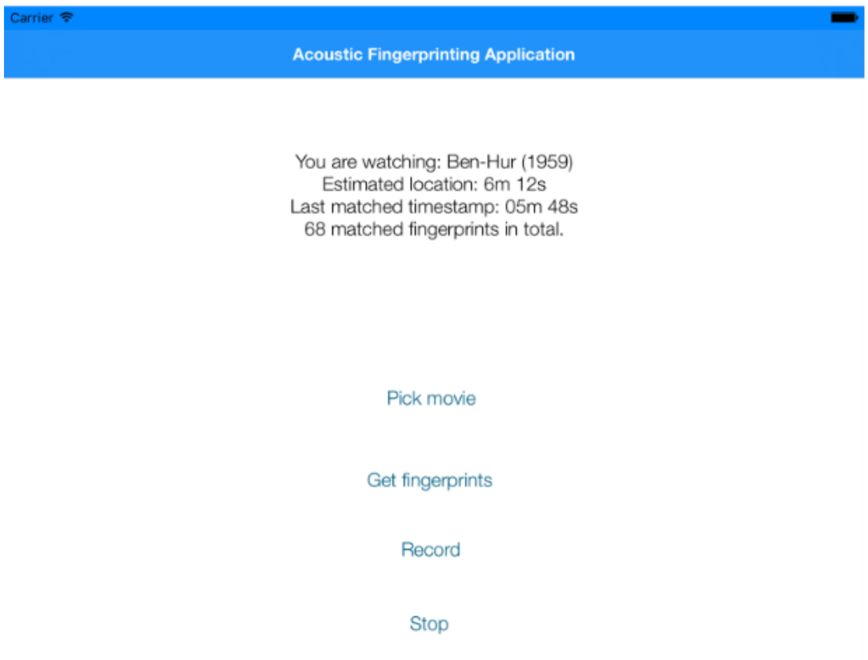
Figure 4.1 shows the Window Application user interface, with the uppermost inner window showing the available device drivers, and the window below showing the information which is being returned to the user of the application. The user controls are present on the left-hand side.

4.1.3 iOS Application

The Windows application and the iOS application are closely related, as they have the same high-level functionality, presented on different platforms. iOS-enabled devices run on an ARM processor, while the Windows application will typically only be run under the x86-64 processor architecture. Different processor architectures have different requirements when it comes to efficiency and throughput, while the iOS application also have different requirements with regards to network connectivity, storage and main memory.

On start-up, the acoustic fingerprinting library will query the Web API for available movies through a HTTP request. As the user interface is loaded, the user is instantly able to pick a movie and retrieve the associated fingerprints.

The recording and recognition routine can be started as soon as the fingerprints are received, given that the user gives the microphone permission to record.



You are watching: Ben-Hur (1959)
Estimated location: 6m 12s
Last matched timestamp: 05m 48s
68 matched fingerprints in total.

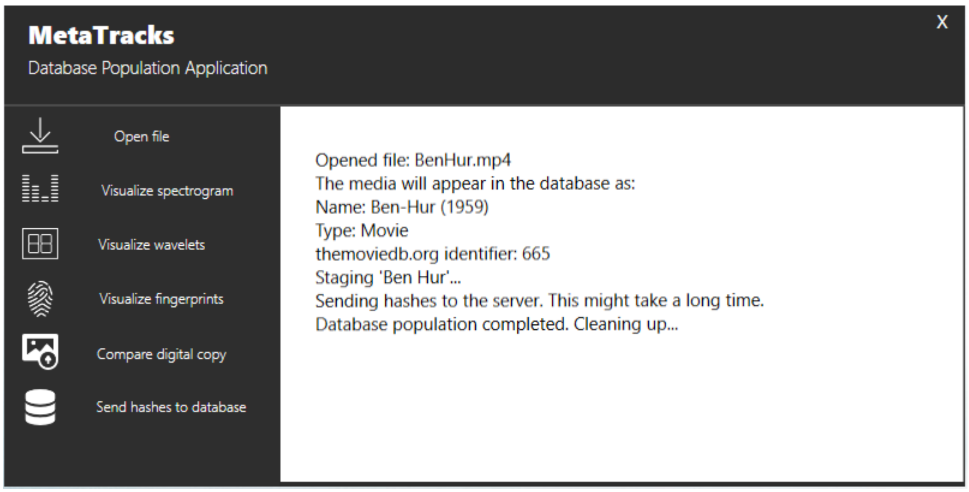
Figure 4.2: Finished iOS application for media recognition.

Figure 4.2 shows the iOS application during runtime. The user of the application is watching a movie, having the last matched timestamp and minute, as well as an estimated position being calculated and returned.

4.1.4 Database Population Application

The Database Population Application is a back-end application used for fingerprinting and hashing movies and TV-series, and inserting the hashed data into our central MySQL database.

The application consists of six distinct parts, where two are required for database population and four of them are for back-end integrity checking of the media. The controls, as shown on the left-hand side in 4.3 have the following functions:
1. Open file. The media file is selected by the user and staged in the application, preparing it for further processing and manipulation.
2. Spectrogram visualization is optional, and provides a visual representation of the media files' spectrogram for integrity checking or analysis of the signal.
3. Wavelet visualization is optional, and provides a visual representation of the spectrogram's individual wavelets generated by image processing techniques and tools.
4. Fingerprint visualization provides a visual representation of the processed spectrogram' and wavelets' fingerprints, the fingerprints that will later be hashed and inserted into the database.
5. The application also provides a means to **compare the integrity** of the preprocessed, purely digital file with a copy already in the database.
6. Finally, staged files can be completely processed and have it's hashes inserted into the database, making it ready for production immediately.



Opened file: BenHur.mp4
The media will appear in the database as:
Name: Ben-Hur (1959)
Type: Movie
themoviedb.org identifier: 665
Staging 'Ben Hur'...
Sending hashes to the server. This might take a long time.
Database population completed. Cleaning up...

Figure 4.3: Back-end Windows application for populating the database with acoustic fingerprints.

4.1.5 Web API

The Web API establishes the connection between the front-end applications and the central acoustic fingerprint database. The Web API will handle incoming HTTP requests from the front-end applications, fetching data from the database based on the incoming HTTP header. A typical HTTP header arriving from the front-end applications could look like this:

```
GET /Fingerprints/GetAllFingerprintsSQL?inputTitle="Fight\%20Club"
```

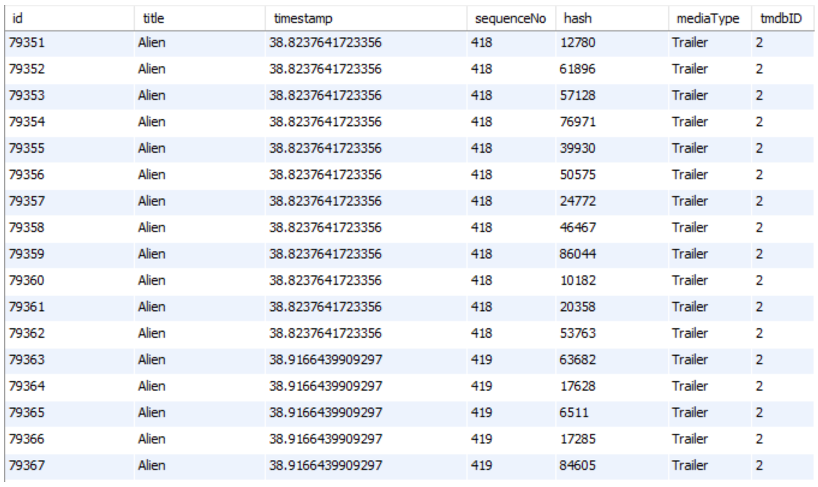
Based on the arriving HTTP header example shown above, the Web API will carry out a SQL query like the following:

```
SELECT * FROM system_users.fingerprintTable WHERE title = 'Fight Club';
```

The data returned from the query will be serialized into a JSON format, which is returned to the front-end client, deserialized client-side and converted into an appropriate, usable object.

4.1.6 Relational database

A table in the central acoustic fingerprint database is shown in Figure 4.4.



id	title	timestamp	sequenceNo	hash	mediaType	tmdbID
79351	Alien	38.8237641723356	418	12780	Trailer	2
79352	Alien	38.8237641723356	418	61896	Trailer	2
79353	Alien	38.8237641723356	418	57128	Trailer	2
79354	Alien	38.8237641723356	418	76971	Trailer	2
79355	Alien	38.8237641723356	418	39930	Trailer	2
79356	Alien	38.8237641723356	418	50575	Trailer	2
79357	Alien	38.8237641723356	418	24772	Trailer	2
79358	Alien	38.8237641723356	418	46467	Trailer	2
79359	Alien	38.8237641723356	418	86044	Trailer	2
79360	Alien	38.8237641723356	418	10182	Trailer	2
79361	Alien	38.8237641723356	418	20358	Trailer	2
79362	Alien	38.8237641723356	418	53763	Trailer	2
79363	Alien	38.9166439909297	419	63682	Trailer	2
79364	Alien	38.9166439909297	419	17628	Trailer	2
79365	Alien	38.9166439909297	419	6511	Trailer	2
79366	Alien	38.9166439909297	419	17285	Trailer	2
79367	Alien	38.9166439909297	419	84605	Trailer	2

Figure 4.4: Acoustic fingerprint database table accessed through the MySQL Workbench software.

The table contain a number of fields, making up the metadata required as per the problem description, including:

(Primary key) id: An unique ID, which is automatically incremented server-side when an acoustic fingerprint is inserted.

title: The title of the fingerprinted media.

timestamp: The timestamps associated with the acoustic fingerprint.

sequenceNo: The sequence in which the hash is contained. This value is incremented every 33 items.

hash: The actual hashed audio fingerprint.

mediaType: Type of media, e.g 'Movie', 'Series', 'Trailer', 'Song'.

tmdbid: An unique ID which corresponds to the media ID from 'The Movie Database'.

4.2 Architecture

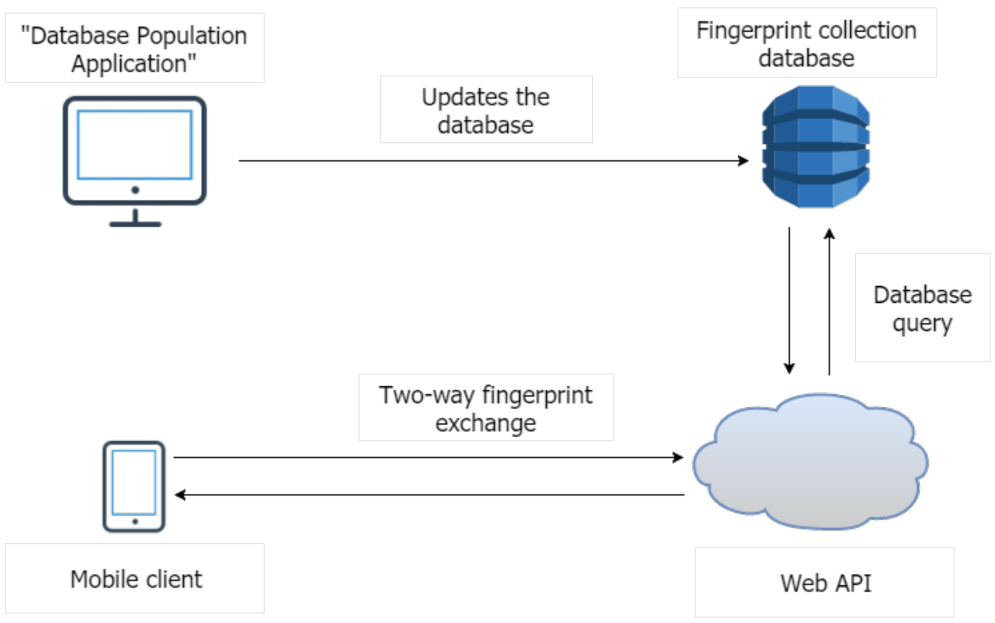

"Database Population Application"
Updates the database
Fingerprint collection database
Database query
Two-way fingerprint exchange
Mobile client
Web API

Figure 4.5: Diagram representation of the system architecture.

4.2.1 Client

The mobile client handles the fingerprint creation process, mainly because if the server were to handle that it would require the client to upload microphone audio to the server which we did not want to do because of privacy concerns. So we decided it would be best for the client to create the fingerprints by itself.

We also decided on letting the client handle the recognition process because if the server were to handle this, it would generate a lot of network traffic if we were to have a lot of users. While handling the recognition process on the client side will reduce performance slightly for an individual user, the benefits out weight the negatives.

4.2.2 Server

On the server-side, it is mostly about communication between device and database. The client sends a request for all available movies currently in the database to the server → server queries the database → sends queried data back to client, all in a single go. The process is the same for downloading the fingerprints from a movie to the client.

4.2.3 Comparing different services

This will show some of the similarities and differences our application has with various other services in terms of specific features used in the fingerprinting creation process.

	Philips	Shazam	AcoustID	MetaTracks
Sample Rate (Hz)	5000	44100	8000	5512
Frequency Range (Hz)	<2000	<20000	<4000	<2000
Window (samples)	2048	32768	512	2048
Overlap	9/10	63/64	1/2	63/64

Table 4.1: Benchmarking of different media recognition services.

Other similar services focus on using audio fingerprinting to recognize music and snippets of sound at near real-time. Our system is focused on recognizing audio from movies and TV-series in real-time. This means our system has to use slightly different specifications as shown in Table 4.2.3.

Along with the different media types, other popular services, as well as services mentioned in Table 4.2.3 focus on finding out which song is currently playing, where the user is in the song is not important. With our service however it's the opposite, we pick a predefined movie from the database and use audio fingerprinting to recognize where the user currently is.

4.3 Process

4.3.1 Agile methodologies

For the duration of the project we decided on using the agile development methodology ”Scrum”, the de-facto standard in agile development. After using this methodology for several other school-related projects, we have gained knowledge and experience related to this methodology. It is a fact that both large and small ICT-projects benefit from using an agile development methodology. This is mainly because it takes into account continuous change throughout the projects development, which is of utmost importance to projects where the requirements specification and technologies is likely to change throughout the project life-cycle.

Scrum is a flexible, agile methodology for handling the development of software. In agile methodologies like Scrum, incremental deliveries and a short planning horizon (typically two to four weeks) is recommended. Project starting and endings are clearly defined.

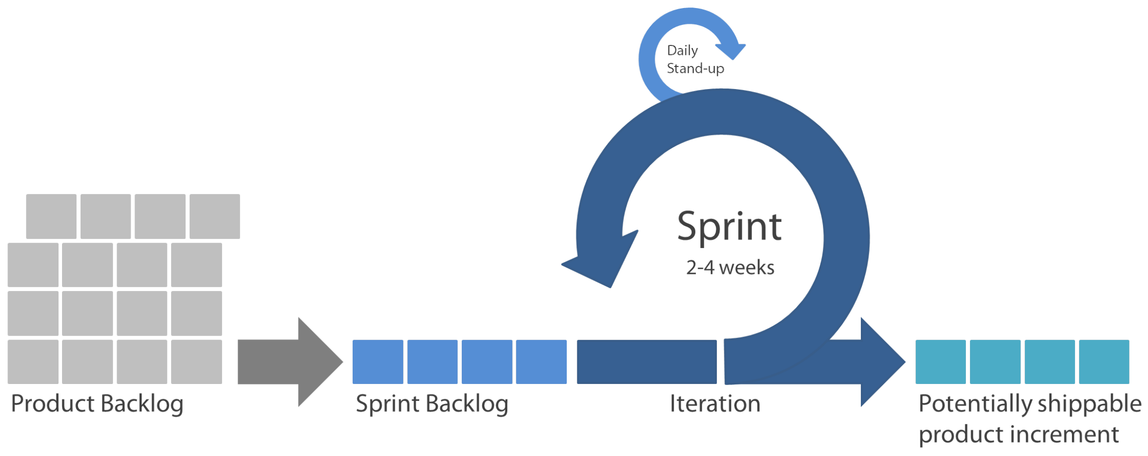


Figure 4.6: Diagram showing the Scrum process.

In figure 4.6 we can see how Scrum works. The project group prioritize the stories from the backlog and put them in the sprint backlog for the current sprint. The stories are worked on through a two to four week long sprint, ending in a new iteration of the product. During every working day of the sprint, daily stand-up meetings are held where each Scrum team member explains what has been done and what was not finished, explaining the reasons for not finishing a story, if that would be the case.

5 Discussion

Chapter contents

5.1 Communication

Throughout the development process, we had multiple meetings with our employer. These supervisory meetings were mostly concerned about showing our development progress, and deciding on what we should focus on next. The meetings also included discussions about third-party library licensing and relevant patents.

5.2 Development and platform

Our finished product could be integrated as a library into a larger platform in the future. The library will have to work on both client computers and mobile devices devices. For populating the database, the Database Population Application will import movies, create fingerprints and upload hashes to database.

The system was developed to work for multiple platforms from the start, as we have one Dynamic Link Library used for most of the shared methods needed, i.e methods for fingerprinting and recognition. This is to reduce code duplication.

Besides this we have one back-end application for populating the database with acoustic fingerprints, as well as one consumer-facing applications for iOS devices. We use an ASP.NET Web API for communication between the user and database, mainly for added security.

5.3 Experimentation

When we first started this project we were at a loss, we did not know what theory was needed to implement this project. Therefore we did a lot of experiments with different programming languages, libraries and theoretical background, before we settled on C# with Xamarin Studios to push the product to several platforms. Two of the most important environments and languages we did some experimenting with is mentioned below, in subsections (5.3.1) and (5.3.2).

In this section we discuss the various experiments we did to find out where our application works great and where it is lacking in performance. We experimented with three movies and five TV-series, as well as a number of different trailers and songs. The experimentation is done with microphone audio with the media file playing on a different device nearby. We also did some experimentation with a digital comparison with different audio filters and effects to see how many equal fingerprints are matched.

5.3.1 MATLAB

MATLAB was the first application and language we tried, mostly because this is what is usually used in signal processing. Here we learned a lot about how signals behave, and how to process them to get the results we wanted. We played with the idea of creating the application with MATLAB or at least use its MCR (Matlab Runtime Compiler) environment to perform certain tasks, including sampling rate conversion, FFT and Haar wavelet transformations.

During the experimentation phase, we understood that using MATLAB or even some of it for the application was inefficient, considering the fact that, we wanted to store all the basic functions of the application in one standalone library that could be used by different platforms. Using MATLAB for this purpose would not have worked well on different platforms, and would in the end require the user to have MATLAB installed on their platform of choice.

5.3.2 Python

Following the MATLAB experimentation described in the previous section, we decided on exploring a solution where Python code is embedded into the .NET framework. Using IronPython, an open source wrapper designed for this specific purpose, we attempted to import scientific computing libraries into .NET.

While this solution proved somewhat effective with native code, lacking support for the scientific Python libraires SciPy and NumPy forced us to drop this solution and focus on creating a fully native implementation from scratch written in C#.

5.3.3 Movies and TV-series

Our experimentation using either movies and TV-series did not go as well as we had initially hoped for. Several matches throughout the entire movie were found, but only rarely do we find the correct match over a false positive. This is mainly because audio in movies often overlap with other parts in the same movie, causing the recognition to give matches in the wrong places.

5.3.4 Trailers

Movie trailers works well most of the time. From our experimentation, a trailer typically range from 95 to 60 percent successful matches. The reason it works much better on trailers as opposed to movies is because trailers (and songs, for that matter) have much more vibrant and definitive audio as opposed to movies.

5.3.5 Songs

With songs we get decent matches. It does have issues with getting the correct time stamp during for example choruses in digital files, naturally.

5.3.6 Experimentation data

We did some experimentation with different filters and distortion. This was done by having two equal sound files and adding filters and distortions on one of them to see how much is still equal after the fingerprinting process. This helps us see what type of filters and distortion our fingerprinting process is resistant to and what distortion effects it has problems with. We used Audacity to add and distort filters and effects to the sound file.

Noise Type	Factor	Matched percentage
White noise	50%	14.51%
White noise	20%	59.10%
Echo	1s offset	68.01%
High Pass Filter	1000 Hz	∼100%
Low Pass Filter	1000 Hz	∼100%
Temp Change	20%	59.4%
Tone (Sine)	400 Hz, 0.8 Amp	∼100%
Tone (Square)	400 Hz, 0.5 Amp	90.14%
Pitch	+ 50%	2.3%
Pitch	- 50%	0.597%

Table 5.1: Acoustic fingerprinting matching accuracy with added digital noise and distortion.

From the results of our experimentation, we got some interesting results. Our fingerprinting process is very noise resistant, it started having problems when 50% of the audio was only white noise. It also handles Echo extremely well, giving us a 68% equal match. Tempo changing was also something it handled surprisingly well, with a 20% increased tempo of the sound file, it still got 59.4% correct matches.

What it had big problems with was pitch changes. Only receiving 2.3% and 0.597% equal matches. This could mean if the media playing has pitched audio, our algorithm might have some problems being able to recognize where the user is in the video.

5.4 Future work

In the future there are ways to improve our algorithm that we did not have the time to explore. Some ways to improve it would be to explore possibilities with further compressing the audio to be more lenient with lower quality audio and noise.

5.4.1 Different Hashing

Another possibility is to use different methods to hash our fingerprints, for example the usage of perceptual hashing could be beneficial as it is also a way to do robust hashing. Neural Hashing is an interesting new technology that could also be used.

5.4.2 Different Implementation

If we want to improve the algorithm without changing the hashing method or compressing audio further, we would have to look into going away from our current Wavelet-based approach and find a different way to create our fingerprints.

5.4.2.1 Landmarks

We could think about using a Landmarks based implementation, which is what Shazam uses for their audio fingerprinting.

The basic operation of this scheme is that each audio track is analyzed to find prominent onsets concentrated in frequency, since these onsets are most likely to be preserved in noise and distortion. These onsets are formed into pairs, parameterized by the frequencies of the peaks and the time inbetween them. These values are quantized to give a relatively large number of distinct landmark hashes. Parameters are tuned to give around 20-50 landmarks per second.[22]

5.4.2.2 Speech Recognition

A completely different possible way to create this type of project could be to use speech recognition software, instead of any fingerprints, to find where you are in the movie. The way this could work is by listening to what is said in the movie, the application will search through the transcripts of the entire movie and look for the best match and return the time stamp of the match.

The benefits of this is that it would be much faster than the creation of fingerprints and the recognition would also be much quicker as it only looks through plain text transcripts to find where you are.

However the downside with an approach like this would be if someone is talking in the room with the movie playing might mess up the searching. Also this requires something to be clearly said in the movie to recognise it, it will only be able to recognise where you are if the application is able to clearly hear what is said.

5.5 Deviations from the original plan

5.5.1 Migration from C# to Swift

Early on in the project it was agreed by us and the employer the project was to be written in C# with Xamarin.iOS to run on iOS devices, but in April it was revealed Xamarin.iOS is actually incompatible with our employers platform as Xamarin only allows integration with other Xamarin and C# projects. So we had to rewrite our entire code base from C# to Swift.

In the future this is something we should learn from this as rewriting an entire application to a programming language none of us knew is something that set us back by multiple weeks. We should have discussed in more detail how our project would be integrated as it is vital to know early on if there is any compatibility issues that could come up.

5.5.2 Selection of Media

Originally we wanted to make the algorithm search for all movies and just automatically find what movie were being watched, and then find the time stamp of the movie, however this would require a lot of computing resources that we do not have access to.

Because of this we decided on letting the user select the movie being watched first, so the user selects a movie he/she wants to watch and then all fingerprints from that movie is downloaded onto the user's device (takes up roughly 7MB) and it will start the recognition process trying to figure out how far the user is in the movie.

5.5.3 Network Traffic / Client-side Recognition

One issue we could see during the development of the project was that doing the recognition on the server would take up a lot of resources and network bandwidth because the process for each user would be to upload a 3 second snippet of fingerprints every 3rd second and require a recognition to be done, following this the server would send back results. This creates a lot of unnecessary traffic for the server to handle and it will not scale very well with a lot of users considering this process is something that will happen very often for the entirety of a movie.

Our solution to this problem was to move the recognition from server-side to client-side. This means the user selects the movie he/she wants to watch and all fingerprints from that movie gets downloaded to the users device and the users device will handle both fingerprinting and recognition. As a collection of fingerprints for an entire movie is roughly ∼7 MB this will not be a problem for most users.

The downside with this approach is that the performance on a mobile device is not that high. So the recognition does take longer than it would have if we had decided to let the recognition happen server-side. But we think the trade-off is worth it considering the reduction of network traffic.

6 Conclusion and recommendations

Chapter contents

6.1 Conclusion

In this thesis we have presented the implementation of a Acoustic Fingerprinting system using Computer Vision technology. The system is able to recognize audio from different media using small recorded audio snippets from a mobile device and give real-time near accurate time stamps on where the user is in the current media playback. The accuracy varies depending on the audio being played from the media, but in general, if the audio is vibrant it will create unique fingerprints and give acceptable matches.

One concern early on was how much disk space fingerprints from an entire movie would use after we uploaded it to the central fingerprint database, but in the end our fingerprints do not take up a lot of space. An entire movie with 2-3 million fingerprints takes up only about 7 MB of disk space, and a TV-series episode takes up roughly 3 MB of disk space. (Both with regular runtimes, 2 hours and 20 minutes respectively)

We also had to try be efficient with the creation of our acoustic fingerprints. There is an apparent trade-off when generating quality fingerprints. If we desire, we could generate a fingerprint much quicker than we currently do by not having any compression, but doing that would mean the fingerprints would not take into consideration different quality of sound or any noise or distortion. We decided to use fingerprints with a lot of compression, so if the sound is similar to the human ear, the fingerprints should be relatively similar by perceptual hashing.

Recognition and matching of acoustic fingerprints is a routine that could take a lot of time, as it needs to iterate through millions of fingerprints to recognize which fingerprints that can be considered potential matches. So instead of iterating through every element in the data set, we decided to use a Binary Search implementation which greatly reduces any searching to be done in a collection of millions of hashes to search through. In addition, we implemented two different variants of the binary search, covering different fingerprint ranges.

In order to do this we need to greatly compress the audio while still keeping the most important information in the audio clear. We decided to reduce sample rate, perform haar wavelet transform and extract the most energetic points in the audio while discarding the rest.

Early on we were thinking on letting client-side handle creation of fingerprints and let the server-side handle recognition, but later on we switched and moved Recognition to client-side. Currently the server-side only handles sending and receiving data from database to client while client handles the fingerprints.

Since the collection of fingerprints for an entire movie is such a low size, any modern mobile device can handle having those fingerprints in memory, as it also greatly improves performance to use memory instead of local storage.

6.2 Recommendations

The following recommendations are offered by the project group, for further research into the aspiring field of acoustic fingerprinting, include some of the project group's personal views on the field.

1. As the acoustic fingerprint research field expands over time and becomes more accessible, acoustic fingerprinting databases for movies and TV-series could be open-sourced, much like the 'MusicBrainz' database for music. This would effectively reduce the proprietary 'hold' which movie studios have over the content. This would benefit the open-source community, and with their cooperation also benefit the movie studios personally.

2. In terms of computational power, real-time fingerprint comparisons are very costly. Efficient sorting and searching algorithms would have to be implemented to account for this cost, over different platforms and differing orders of operation on these platforms.

3. As the database of acoustic fingerprints grow large enough (containing billions of fingerprints), a more permanent storage solution would be required. Where we have used a traditional relational database, a non-relational database could take it's place, where query times are known to be quicker. For our project, this was purely a monetary cost issue, with Amazon DynamoDB (NoSQL) charge a fee per throughput unit.

Bibliography

[1] Victor S. Adamchik. *Algorithmic Complexity.* URL: https://www.cs.cmu.edu/~adamchik/15-121/lectures/Algorithmic%20Complexity/complexity.html.

[2] *Advanced Audio Coding.* URL: https://en.wikipedia.org/wiki/Advanced_Audio_Coding.

[3] *Audio and sound file extension list.* URL: http://www.file-extensions.org/filetype/extension/name/audio-and-sound-files.

[4] *Audio file formats.* URL: http://www.nch.com.au/acm/formats.html.

[5] *Audio quality in networked systems.* URL: http://download.yamaha.com/api/asset/file/?language=en&site=countrysite-master.prod.wsys.yamaha.com&asset_id=60732.

[6] Melinos Averkiou. *Digital Watermarking.* URL: https://www.cl.cam.ac.uk/teaching/0910/R08/work/essay-ma485-watermarking.pdf.

[7] Shumeet Baluja and Michele Covell. "Waveprint: Efficient wavelet-based audio fingerprinting". In: *Pattern recognition* 41.11 (2008), pp. 3467–3480.

[8] *Big O notation.* URL: http://web.mit.edu/16.070/www/lecture/big_o.pdf.

[9] Pedro Cano et al. "A review of audio fingerprinting". In: *Journal of VLSI signal processing systems for signal, image and video technology* 41.3 (2005), pp. 271–284.

[10] Matthew Casperson. *Minhash for dummies.* URL: http://matthewcasperson.blogspot.no/2013/11/minhash-for-dummies.html.

[11] Lahouari Ghouti and Ahmed Bouridane. "A robust perceptual audio hashing using balanced multiwavelets". In: *Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on.* Vol. 5. IEEE. 2006, pp. V–V.

[12] Jaap Haitsma, Ton Kalker, and Job Oostveen. “Robust audio hashing for content identification”. In: *International Workshop on Content-Based Multimedia Indexing*. Vol. 4. Citeseer. 2001, pp. 117–124.

[13] F. J. Harris. *On the use of windows for harmonic analysis with the discrete Fourier transform*. Vol. 1. Institute of Electrical and Electronics Engineers, 1978.

[14] *Jaccard Index*. URL: `https://en.wikipedia.org/wiki/Jaccard_index`.

[15] Sara Kudrle. “Media Fingerprinting: Managing Content, Security and Quality”. In: *Miranda Technologies, Grass Valley, California, presented on Apr* 12 (2010).

[16] M Kivanç Mıhçak and Ramarathnam Venkatesan. “A perceptual audio hashing algorithm: a tool for robust audio identification and information hiding”. In: *Information Hiding*. Springer. 2001, pp. 51–65.

[17] Colm Mulcahy. “Image compression using the Haar wavelet transform”. In: ().

[18] Jeff M. Phillips. *Jaccard Similarity and Shingling*. URL: `https://www.cs.utah.edu/~jeffp/teaching/cs5955/L4-Jaccard+Shingle.pdf`.

[19] Jeff M. Phillips. *Minhash*. URL: `https://www.cs.utah.edu/~jeffp/teaching/cs5955/L5-Minhash.pdf`.

[20] Anand Rajaraman et al. *Mining of massive datasets*. Vol. 1. Cambridge University Press Cambridge, 2012.

[21] Mathieu Ramona and Geoffroy Peeters. “AudioPrint: An efficient audio fingerprint system based on a novel cost-less synchronization scheme”. In: *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE. 2013, pp. 818–822.

[22] *Robust Landmark-Based Audio Fingerprinting*. URL: `http://labrosa.ee.columbia.edu/matlab/fingerprint/`.

[23] Kenneth H. Rosen. *Discrete Mathematics and Its Applications*. Vol. 7. McGraw-Hill Education, 2011.

[24] *Shazam on iPhone could change music discovery*. URL: `http://www.cnet.com/news/shazam-on-iphone-could-change-music-discovery`.

[25] Julius O. Smith III. *Digital Audio Resampling Home Page*. URL: `https://ccrma.stanford.edu/~jos/resample/`.

[26] Joyce Van De Vegte. *Fundamentals of Digital Signal Processing*. Prentice Hall PTR, 2002.

[27] Brani Vidakovic and Peter Mueller. “WAVELETS FOR KIDS”. In: ().

[28] Avery Wang et al. “An Industrial Strength Audio Search Algorithm.” In: *ISMIR*. 2003, pp. 7–13.

[29] *YouTube, Content ID*. URL: `https://en.wikipedia.org/wiki/YouTube#Content_ID`.

Appendix A:
Preliminary Report

Preliminary report - MetaTracks

The following report is delivered by

Name	Student number
Kristoffer L. Nesvik	120249
Glenn S. Skjong	120170
Kristian Støylen	120515

For evaluation in the course
IE303612 Bachelor Thesis - Automation and computer engineering

Supervisor: Kjell Inge Tomren
Employer: Christian A. Strømmen

NTNU

Department for engineering and natural sciences
NORWEGIAN UNIVERSITY OF SCIENCE AND TECHNOLOGY
Larsgården 2 - 6009 Ålesund - Norway
Spring semester 2016

Contents

1 Abstract

When choosing our final assignment, we set several criteria that had to be fulfilled in order to get the most value out of the assignment. We wanted to utilize all the knowledge gained through several subjects and assignments while being enrolled as computer engineering students, including system development, several programming languages, development methodologies and information security.

Our application was accepted by "MetaTracks", a start-up company from the Ålesund area. MetaTracks' main goal is to create an innovative web-based service for content synchronized with digital media such as movies and TV series. Working with a start-up company will hopefully give us many valuable experiences we would not otherwise get when working for a traditional "large corporation". These days, start-up companies are becoming increasingly popular, both in Norway and abroad.

MetaTracks was started and is run by Christian Strømmen, which will act as our employer for the course of the project. The project is financed by Innovation Norway and other external investors.

2 Terms

Acoustic (audio) fingerprint: A digital summary of audio signals. Acoustic fingerprints are usually generated by transforming and hashing *spectrograms*. There are no set rules or standards for *how* these fingerprints should be generated, and generation methods differ greatly.[1]

Spectrogram: A visual representation of the time-frequency spectrum in several types of signals. The intensity of the signal is also represented in a spectrogram.[4]

Fingerprint database: A centralized collection of acoustic fingerprints. The fingerprints are usually stored with a focus on quick access.

Scrum: An agile framework for software development of complex systems. Scrum focuses on incremental and interative work and delivery through the projects life-cycle.[12]

Multimedia: Defined as content using a combination of *content forms*, this could be a combination of audio, video, images, animation, and interactivity.[3]

Life cycle: The ”system development life cycle” is a set of steps required to deliver a stable, finished software product. The cycle typically include five main phases: planning, analysis, design, implementation and maintenance.[15]

Signal processing: Signal processing revolve around representing signals in a digital format. Once digitally represented, a signal can be analyzed, modified, or have its digital information retrieved.[8]

3 Project organization

In this chapter of the report we present how we have planned the organization of the project group. Each group member will have different roles, with specific tasks and responsibilities. In the end of the chapter we present the supervising group for the project, which consist of the employer and the supervisor.

3.1 Project group

Student number	**Name**
120249	Kristoffer L. Nesvik
120170	Glenn S. Skjong
120515	Kristian Støylen

Table 3.1: The student numbers and names for all the group members delivering the assignment for evaluation in the course IE303612.

3.1.1 Roles in the project group

Kristoffer L. Nesvik: Project leader
Glenn S. Skjong: Scrum leader
Kristian Støylen: Secretary

In addition to the roles defined above, each group member will be responsible for decisions relating to the project. This includes, but are not limited to; the planning, development and documentation of the product.

3.1.2 Project leader tasks

Area of responsibility:

- Enforcing group norms and rules, as described in chapter 4.3.
- Delegating responsibility and tasks to the project group members, for example specific tasks related to project development.
- Create a working environment with good morale and motivation.

Project leader specific tasks:

- Overall responsibility for making sure that partial goals and deliverables are finished on time and is delivered as a whole.
- Tasked with supervising the software development, making sure that good routines and standards related to software development is followed throughout the project.
- The project leader is participating in the planning, development and documentation work on the same level as the rest of the project group.

3.1.3 Secretary tasks

Area of responsibility:

- Overall responsibility for documentation and other written documentation, both internal documents for the project group and documents meant for the supervising group and/or the University.
- Responsible for logging project work and project advances.

Secretary specific tasks:

- Write summaries and documentation from meetings between the supervising group and the project group.
- Quality assurance of written documentation.
- The secretary is participating in the planning, development and documentation work on the same level as the rest of the project group.

3.1.4 Scrum leader tasks

Area of responsibility:

- Responsible for the agile development framework ”Scrum”, assuring that this framework is used in accordance with the methodology standards.
- Planning ”sprints” and evaluating the ”Product Backlog.”, ref. chapter 5.3.

Scrum leader specific tasks:

- Responsible for assuring that the Scrum-team works as planned to meet the required amount of work described in the sprint log.
- The Scrum leader is participating in the planning, development and documentation work on the same level as the rest of the project group.

3.2 Supervising group

The supervising group consists of:

Project supervisor Kjell Inge Tomren at NTNU

Employer and product owner: Christian A. Strømmen at MetaTracks

4 Agreements

This chapter describes the agreed upon contract between the employer and the project group, in addition to group norms and rules that will be followed for the duration of the project.

4.1 Contract of employment

- The project group are to deliver continuous reports describing the ongoing process and milestones in the project to the employer. The reports should be delivered preferably every two weeks.
- Questions the project group might have relating to the products upcoming development should be directed at the employer.

4.2 Workspace and resources

The project group will mainly be working at the facilities of NTNU Ålesund. Professors and other specialists will be available at the University for consultation.

Notables resources the project group will have access to for the duration of the project is Atlassian's "JIRA", an advanced tool for planning and execution of development projects. A dedicated service for server- and database hosting will also be required for development and testing of the product, which will be provided by the University.

Resources related to testing, for example testing of specific variants and platforms, will be made available by the employer.

Most of the details surrounding the project will be withheld from public disclosure, where a non-disclosure agreement will be signed between the project group and the employer in the event that such information is presented to the project group.

Meetings related to the development of the project between the project group and the supervising group will be held preferably every two weeks.

4.3 Group norms and rules of cooperation

Group norms and rules of cooperation that will be the foundation for the project work includes:

- Project group meetings shall begin and end on time.
- All group members will take part in decisions relating to the project development.
- Cooperation and transparency shall be promoted during the project.
- A positive attitude towards the project and enforcing good learning behaviors shall make up the foundation of the work.

Perspectives and attitude towards the profession:

- Continuous ”maintenance” of knowledge and abilities through continuous professional and hobbyist work.
- A profound focus on information security and secure operations of software systems, in accordance with applicable laws and regulations.

5 Project description

MetaTracks is a start-up company, aiming to create an innovative web-based service for content synchronized with digital media, such as movies and TV-series.

The project is financed both by Innovation Norway and other external investors.

5.1 Topics, goals and intentions

The project group will be working on a sub-project for MetaTracks. The project includes the development of a software solution that can detect which film or episode a user is watching. The software must also be able to identify where in the film or episode the user is at any given time. To make this possible, the employer means to use acoustic fingerprints sourced from the audio in the multimedia content presented to the end user.

The project consists of producing a system that can generate these acoustic fingerprints from movies and TV, as well as recognizing the audio from the content the user is currently watching to match this audio with fingerprints in a central database.

As a requirement set by the employer, an application for the Apple iPhone is required. As this application can be developed in either Objective-C or Swift programming languages, it can communicate directly with an ASP.NET server developed by the project group. The iPhone requires a standalone application as the native web browser Safari does not support direct microphone access.[5]

In addition to the iPhone application, the project group aims to develop a web application, runnable in traditional web browsers from a personal computer. Supported web browsers would include Google Chrome, Mozilla Firefox and Microsoft Edge, as these support microphone access natively. access.[5]

The project group plans on developing a back-end ASP.NET application and develop a front-end application for web browsers initially. Once the back-end server is complete, and the front-end application can communicate with the server effectively, the Apple iPhone application can be developed. Web applications can easily be tested and debugged, and parts of the code can be reused for the iPhone application once the web application is complete.

It must be possible to generate fingerprints from a ”pure” digital soundtrack in a multimedia file, as well as from a microphone in a mobile client such as a smart-phone. The fingerprints generated from a mobile client has to take into account any noise or distortion present on the channel When crowd-sourcing fingerprints, the system might need several applicable fingerprints before finally accepting them.

As required by the employer and requirements specifications, the fingerprints are to cover a duration not exceeding one second. The goal would be to generate acceptable fingerprints spanning 250 milliseconds. In effect, this means that close to 300 fingerprints would be required per minute of playback. A movie with a run-time of two hours would require up to 36 000 fingerprints. Efficient storage of fingerprints, taking into account the accommodation of the large amount of fingerprints, is also a problem that will be addressed by the project group.

5.1.1 Project challenges

Some of the challenges posed in the project include:

- Generating fingerprints from soundtracks in ”pure” multimedia files.
- Generating fingerprints from a microphone on a mobile client, such as a smart-phone.
- When a mobile client is used, how do we compare that fingerprint with a ”pure” fingerprint from a digital source?
- How many identical or acceptable fingerprints is required from the user before we can accept the fingerprint and be sure of the users location?
- Comparing the fingerprints from the user with the fingerprints in a centralized database.
- Continuous checks must be carried out to identify where the user is in a multimedia playback.
- When carrying out these checks, it must be possible to identify if the user has paused the playback or skipped either forward or backwards in the playback.
- Good resource utilization is paramount, both for generating the fingerprints and matching them to existing fingerprints. The service will first and foremost run in the background on mobile devices, where resources are limited.

5.2 Requirements specification

Purpose of the requirements specification

The following requirement specifications purpose is to give an overall description of the development of the product; including specifications and purpose of the actual product, requirements for the product and specific *functional* requirements for the product.

Purpose of the product

The product that the project group will develop will be integrated in the main "MetaTracks" product. This integration must be achieved in an effective and elegant manner.

Function of the product

- Effectively generating digital fingerprints from multimedia content.
- Utilizing these fingerprints to create an innovative web service with returning users.

Specific functional requirements

- Generating fingerprints must be possible on clients with limited resources, such as mobile clients.
- The system must be able to generate acceptable fingerprints in environments with potential sources of noise and distortion.
- The product must function on several platforms.
- Documentation with relevant test data must be included to decide on the feasibility of the product.

Programming languages and development

The software will likely be written in, or utilize, one or more of the following technologies:

- C#: Programming language developed by Microsoft for their .NET initiative.

- .NET: Software framework developed by Microsoft.
- ASP.NET: Web application framework, part of the .NET framework.
- Swift: Programming language designed for producing applications for Apple products.
- Objective-C: Similar to Swift, developed for Apple systems.
- HTML5: Markup-language for structuring websites.
- CSS: Stylesheet language for designing websites.
- JavaScript and WebRTC: Programming language and extension for adding functionality to websites.
- MATLAB: Technical computing language, used in the project for testing algorithm implementations.

Virtualization and development server

On our virtual server hosted by the University we will run the following services. The goal is to emulate the actual services that will run on the production server once the product is complete.

- Amazon DynamoDB: A flexible NoSQL database for storing the acoustic fingerprints. Focus on low-latency access, which is required when aiming for a 250 ms fingerprint duration.
- Windows Server 2012 R2: For hosting the ASP.NET-based application.
- A secure FTPS server or similar service for pushing changes to the development server.

The list of services is subject to change through the project life-cycle.

Solutions

As explained earlier in this document, audio is subject to noise and distortion from several sources. To address this problem in an effective manner, there are several solutions that might be worth exploring:

- Signal filtering techniques can be used, such as Elliptical, Butterworth and Chebyshev-filtering techniques.[13]
- Utilizing open-source software designed for these tasks, as explained in the next section.

Licensing

If the project uses libraries and modules from project with an open source license, these should be licensed under the LGPL license, MS-PL license or other similar licenses.[2]
Examples of relevant libraries that we can use in our project includes:

- NAduio
- CSCore

Both of these libraries are licensed under the MS-PL license.

Relevant patents to consider

There are several patents covering the generation and identification of acoustic fingerprints in digital media.
As the project group is developing the software solution, it is important to consider possible patents to avoid possible patent infringement from patent holders.
Some of the relevant patents include:

- US 7013301, granted to "AmpliFIND": Covering a similar service strictly for *music*, utilizing Fast Fourier Transforms and Singular Value Decomposition on matrices.[6]
- US 8949872, granted to "Yahoo! Inc.": Covering a similar service strictly for *television*, utilizing segmentation of the given multimedia and creating vectors based on these generated segments[14]
- US 7487180, granted to Holm, Frode & Hicken, Wendell: Covering the system implementation of the US 7013301 grant.[7]
- US 8380518, granted to "Samsung Electronics Co. Ltd": Covering a system for recognizing user "moods" in several music applications.[9]
- US 8886531, granted to Vogel, Brian K: Covering a technique for identifying possible duplicate entries in larger collections of audio.[16]
- US 7881931, granted to "Gracenote Inc.": Covering a system for equalizing different audio tracks through acoustic fingerprinting.[18]
- US 7873521, granted to "Nippon Telegraph And Telephone Corporation": A wide patent covering several audio recognition inventions, systems and signal processing mechanisms.[10]

This is by no means an exhaustive list, and will be expanded as the project planning continues. The list only include relevant *grants*, not pending applications.

5.3 Plan of action

For the duration of the project we have decided on using the agile development methodology "Scrum", the de-facto standard in agile development. After using this methodology for several other school-related projects, we have gained knowledge and experience related to this methodology. It is a fact that both large and small ICT-projects benefit from using an agile development methodology. This is mainly because it takes into account continuous change throughout the projects development, which is of utmost importance to projects where the requirements specification and technologies is likely to change throughout the project life-cycle.

Scrum is a flexible, agile methodology for handling the development of software. In agile methodologies like Scrum, incremental deliveries and a short planning horizon (typically two to four weeks) is recommended. Project starting and endings are clearly defined.[12]

The methodology recommends the project group to carry out short, daily project meetings. These meetings, called "scrum-meetings", usually span from ten to fifteen minutes. The meetings have a clearly defined structure, and the goal is to answer three important questions[12]:

- What has been done since the last meeting?
- What will be done until the next meeting?
- What obstacles was encountered, hindering the group member(s) from effectively implementing the planned functionality?

An efficient Scrum-process include three different roles[12]:

- *The product owner*, responsible for prioritizing the product queue.
- *The Scrum-master*, responsible for leading the team and acting as a gatekeeper, making sure the "Scrum team" is not disturbed by tasks not included in the ongoing sprint.
- *The Scrum-team*, responsible for carrying out the actual programming, writing documentation and delivering the actual product at the end of each sprint.

5.4 Information gathering

Several current systems and solutions utilize technology similar to the technology the project group is tasked with developing;

- MusicBrainZ (open-source)
- Shazam[17]
- SoundHound
- DejaVu (open-source)

Most of these systems focus solely on music, but similiar technology could be implemented partially, for example to identify opening themes or soundtracks in a film or series.

Further information the project group might need during the project will primarily be gathered through specialists from NTNU Ålesund.

5.5 Risk analysis and assessment

After a preliminary discussion of technologies and possible solutions, we deem the project completion likely, considering that some requirements are met.

- Effective communication between the project group and the employer.
- Proper planning and assessments, streamlining the development.
- Proper execution of development, with programming standards and documentation standards being followed for all applicable languages.
- Thorough testing and analysis of the solution.

In addition to success factors, there are several elements that might pose a risk to the project planning, execution or testing:

- Poor planning and communication, effectively threatening the project even prior to development.
- Non-standardized or poor programming habits would effectively make the project unmaintainable.
- Deficient testing could possibly ship an incomplete product to the users, threatening the profitability and the goal of the entire project.

5.6 Main activities in the project work

The activities presented in the table below will, according to plan, begin on 01/02/2016. A gantt-diagram representation and breakdown chart is included in the hand-in folder.

#	Activity	Responsibility	Cost	Timespan
A1	**Project planning and development environment setup**	Project group	N/A	**3.5 days**
A1.1	Preliminary report finished	Project group	N/A	N/A
A1.2	Configuration of development server	Project group	N/A	2 days
A1.3	Setup and configuration of development environment	Project group	N/A	1 days
A1.4	Collection of royalty-free multimedia material	Project group	N/A	0.5 days
A2	**First draft of the system**	Project group	N/A	**6 weeks**
A2.1	Generate library (database) with hashed media files	Project group	N/A	2 days
A2.2	Recognize audio from digital input (Web app)	Project group	N/A	2 weeks
A2.3	Recognize audio from microphone input (Wep app)	Project group	N/A	2 weeks
A2.4	Recognize audio from microphone (iPhone app)	Project group	N/A	2 weeks
A3	**Working system prototype**	Project group	N/A	**6-7 weeks**
A3.1	Complete standalone web application	Project group	N/A	2 weeks
A3.2	Complete standalone iPhone application	Project group	N/A	2 weeks
A3.3	Working pausing, jumping, cross-platform resume.	Project group	N/A	2 weeks
A4	**Test cases, upgrading and debugging**	Project group	N/A	**5 weeks**
A4.1	Data collection from users and test cases	Project group	N/A	1 week
A4.2	Analyzing the collected data	Project group	N/A	1 weeks
A4.3	Working system prototype	Project group	N/A	3 weeks
A5	**Deliverables: system documentation, reports**	Project group	N/A	**1 week**

Table 5.1: Main activities and partial activities

The project will not come with a cost extending the non-capital expenses related to work-hours put in by the project group.

5.7 Planning and project management

5.7.1 Primary plan

The main task of the project group is to test the *feasibility* of the system. In accordance with the milestones in table 5.1, the project group will eventually deliver an argued decision whether the project is feasible or not- including a functioning system prototype.

5.7.2 Project management tools

- Atlassian JIRA
- Microsoft Project Professional 2016

5.7.3 Development tools

- ShareLaTeX: An online tool for collaboration of scientific documents writte in the LaTeX markup language.
- GitHub: Web-based collaboration tool for software development.
- Several Integrated Development Environments (IDE), including Visual Studio, MonoDevelop and ReSharper.

5.7.4 Internal control and evaluation

Internal control and evaluation of the software solution development will happen together with the employer.

5.8 Decisions and the decision process

Based on the projects nature where continuous testing together with trial and error is central, the entire project is likely to change drastically during the projects life-cycle.

Limiting the scope of the project will happen in collaboration with the employer.

As the project will be subject to continuous change, we will utilize an agile development methodology which takes this into account, as explained in chapter 5.3. Using updated development standards and best-practice methods is of utmost importance to cope with these changes, and will create a foundation for the project development.

6 Documentation

This chapter describes the documents and reports that will be worked out and written by the project group for the duration of the project.

6.1 Reports and technical documents

The following documentation and reports shall be worked out by the project group over the course of the project[11]:

- Development routines.
- Acceptance in accordance with laws and regulations.
- Distribution and copying of materials.
- Justificable storage of materials.
- System maintenance documentation following the project.
- Full system documentation.
- Test and lab data.

7 Planned meetings and reports

Several meetings have to be held for the duration of the project. This is mainly to give the supervising group updates related to how the project is coming along. This chapter lists the planned meetings between the project group and the supervising group, as well as required periodic reporting.

7.1 Meetings

Meetings between the supervising group and the project group will preferably take place at the residence of InnoTown or at the University.

7.1.1 Planned meetings

28.01.2016:	Meeting focusing on patents and upcoming project work.
15.02.2016:	Development meeting with supervising group.
29.02.2016:	Development meeting with supervising group.
14.03.2016:	Development meeting with supervising group.
28.03.2016:	Development meeting with supervising group.
11.04.2016:	Development meeting with supervising group.
25.04.2016:	Development meeting with supervising group.
09.05.2016:	Development meeting with supervising group.
23.05.2016:	Final planned meeting with supervising group.

All meeting dates are tenative and will be subject to change.

7.1.2 Project group meetings

Project meetings for the project group will find place every weekday, mainly from 08:00 to 15:30. The project meetings will include a "Daily Scrum", as well as development, planning and documentation work.

7.2 Periodic reporting

Periodic reports including documentation will be shared between the project group and the supervising group. Several periodic reports are also delivered to NTNU Ålesund for subject evaluation.

8 Project deviations

Handling deviations includes steps that must be carried out if project work is halted, or developed content is not in accordance with plans or visions for the product. These steps includes, but are not limited to:

- Deviating content must immediately be reworked to satisfy project requirements.
- Steps must be taken to ensure that such deviation does not occur again.
- Responsibility for continuous overseeing of potential deviations reside with the project group leader.

9 Required equipment

At different stages in the project, the project group might need equipment which is not typically available to the project group. Equipment for carrying out system tests will be required, mainly Apple products such as the Apple iPhone and possibly the Apple Mac(Book). These clients' behaviors can be difficult to emulate through software.

Bibliography

[1] Kalker & Heitsma Cano Batlle. *A Review of Algorithms for Audio Fingerprinting*. Tech. rep. Universitat Pompeau Fabra & Philips Research Eindhoven, 2002.

[2] Free Software Foundation. *GNU Lesser General Public License*. 2007. URL: `http://www.gnu.org/licenses/lgpl-3.0.en.html`.

[3] Wikipedia Foundation. *Multimedia*. 2015. URL: `https://en.wikipedia.org/wiki/Multimedia`.

[4] Wikipedia Foundation. *Spectrogram*. 2015. URL: `https://en.wikipedia.org/wiki/Spectrogram`.

[5] *getUserMedia supported browsers*.

[6] F. Holm and W.T. Hicken. *Audio fingerprinting system and method*. US Patent 7,013,301. 2006. URL: `https://www.google.no/patents/US7013301`.

[7] F. Holm and W.T. Hicken. *System and method for recognizing audio pieces via audio fingerprinting*. US Patent 7,487,180. 2009. URL: `http://www.google.com/patents/US7487180`.

[8] Emmanuel C Ifeachor and Barrie W Jervis. *Digital signal processing: a practical approach*. Pearson Education, 2002.

[9] H. Kim et al. *Device, method, and medium for generating audio fingerprint and retrieving audio data*. US Patent 8,380,518. 2013. URL: `https://www.google.ch/patents/US8380518`.

[10] T. Kurozumi, H. Nagano, and K. Kashino. *Sound signal detection system, sound signal detection server, image signal search apparatus, image signal search method, image signal search program and medium, signal search apparatus, signal search method and signal search program and medium*. US Patent 7,873,521. 2011. URL: `http://www.google.com/patents/US7873521`.

[11] Nils Olsson. *Praktisk rapportskriving*. 1st ed. 2014.

[12] Johansen Langlo Rolstadås Olsson. *Praktisk Prosjektledelse (fra idé til gevinst)*. 1st ed. 2014.

[13] Singh Singla. *Paper on Frequency based audio Noise Reduction using Butter Worth, Chebyshev & Elliptical Filters*. Tech. rep. Guru Kashi University, 2015. URL: `ttp://www.ijritcc.org/download/browse/Volume\_3\_Issues/October\_15\\\_Volume\_3\_Issue\_10/1446450414\_02-11-2015.pdf`.

[14] M. Slaney and A.H. Schafhauser. *Audio fingerprint for content identification*. US Patent 8,949,872. 2015. URL: `https://www.google.no/patents/US8949872`.

[15] Ian Sommerville. *Software Engineering*. 9th ed. 2011.

[16] B.K. Vogel. *Apparatus and method for generating an audio fingerprint and using a two-stage query*. US Patent 8,886,531. 2014. URL: `http://www.google.com/patents/US8886531`.

[17] Avery Li-Chun Wang. *An Industrial-Strength Audio Search Algorithm*. Tech. rep. Shazam Entertainment, Ltd., 2003. URL: `https://www.ee.columbia.edu/~dpwe/papers/Wang03-shazam.pdf`.

[18] M. Wells et al. *Automatic identification of sound recordings*. US Patent 7,881,931. 2011. URL: `https://www.google.com/patents/US7881931`.

Appendix B:
Weekly reports
Meeting summaries

MetaTracks - Weekly Report

Kristoffer L. Nesvik - Glenn S. Skjong - Kristian Støylen

Week 1: 04.01.2016 - 08.01.2016

Comments

The first week of the project we focused on getting to know the actual problem definition, and brainstorming ideas on how we could tackle the problem presented by the problem description. We discovered several possible solutions, including relevant programming languages, technology stacks and work-flows.

New technologies

- ShareLaTeX
- C# programming language

Actions planned for this week

- Structuring and planning the preliminary project report.
- Choosing roles in the project group.
- Creating a schedule for the project work.
- Plan the first meeting with the supervising group.

Completed actions this week

- Finalized the preliminary report structure.
- Decided on project roles.
  - Kristoffer Nesvik: Group leader
  - Glenn Skjong: Scrum leader
  - Kristian Støylen: Working secretary
- Finalized the project schedule and accounted for schedule change.
- Planned the first project meeting for the coming week.

Completed unplanned actions this week

- Brainstormed a possible technology stack.
- Planned a possible software program flow.
- Started setup of development environment.

Planned actions for upcoming week

- First meeting with the supervising group.
- Continue work on the preliminary project report.
- Find research data relevant to the project.

Issues or overdue actions

No issues or overdue actions this week.

MetaTracks - Weekly Report

Kristoffer L. Nesvik - Glenn S. Skjong - Kristian Støylen

Week 2: 11.01.2016 - 15.01.2016

Comments

Starting the second week of the project, we had the first meeting between us (the project group) and the employer. As we got some more information about the project (minus information withheld the public which will be presented at a later stage), we could continue expanding the preliminary report. We also continued planning related to which technology to use, which platforms to develop for and sprint planning.

New technologies

- Microsoft Visual Studio
- ASP.NET

Actions planned for this week

- Continue work on the preliminary report.
- Planning of upcoming work on the product.
- First meeting with the supervising group.

Completed actions this week

- First meeting between the project group and the supervising group.
- Planned possible "Sprint" distribution and deliverables.
- Continued work on the preliminary report.
- Made sketches of the planned program flow.
- Continued planning of technologies and software stack.

Completed unplanned actions this week

- Placed order for virtual development server at the University.

Planned actions for upcoming week

- Finalize the preliminary project report.
- Set up the virtual server.
- Set up the software stack.

Issues or overdue actions

No issues or overdue actions this week.

MetaTracks - Meeting Summary

Kristoffer L. Nesvik - Glenn S. Skjong - Kristian Støylen

Meeting date: 11.01.2016

Attendance: Kristoffer L. Nesvik - Glenn S. Skjong - Kristian Støylen
Supervisor: Christian A. Strømmen

Summary:
Project group discusses product requirements. Product was to be used in iOS devices, namely iPhone 6s and iPad Pro, and product should use .NET for communication with server on Amazon Web Services.

MetaTracks - Weekly Report

Kristoffer L. Nesvik - Glenn S. Skjong - Kristian Støylen

Week 3: 18.01.2016 - 22.01.2016

New technologies

- Visual C# Razor Syntax
- Microsoft Project Professional 2016

Comments

The third week of the project mainly revolved around finalizing the preliminary report, leaving one full week for miscellaneous fixes and quality assurance. In addition to finalizing the report, setting up the virtual server and setting up the development environment have been planned for this week.

Actions planned for this week

- Finalize preliminary report.
- Set up virtual server.
- Set up software stack and development environment.
- Research and related theory.

Completed actions this week

- Finalized the preliminary report.
- Most of the development environment has been set up.

Completed unplanned actions this week

- Application "mockups".
- Planning of specific technological solutions; hashing, storage, algorithm.

Planned actions for upcoming week

- Set up the virtual development server.
- Quality assurance of preliminary project report.
- A longer meeting with the supervising group.
- Planning and setup of the Scrum Product Backlog.
- Set up Atlassian JIRA and GitHub repository.

Issues or overdue actions

- We were unable to set up the virtual server this week, as we are still waiting for our application to be processed. This item has been planned for next week.

MetaTracks - Weekly Report

Kristoffer L. Nesvik - Glenn S. Skjong - Kristian Støylen

Week 4: 25.01.2016 - 29.01.2016

Comments

Week four of the project mainly consisted of wrapping up the preliminary report. In addition to that, an early version of the product backlog was worked out. At the end of the week, the project group had a meeting with the employer to discuss how the project planning is coming along.

Actions planned for this week

- Finalizing the preliminary project report.
- Meeting with employer.

Completed actions this week

- Finalized the preliminary project report.
- Meeting with the employer.

Completed unplanned actions this week

- Preliminary work on the product backlog and user stories.
- Early work on the fingerprint extraction algorithm.

Planned actions for upcoming week

- Research on algorithms and possible solutions.
- Start preliminary work on the algorithm implementation in MATLAB or Python.
- Learning relevant theory on signal processing.

Issues or overdue actions

- Configuration of Atlassian JIRA (moved to next week)
- Configuration of virtual server (insufficient user account permissions, moved to next week)

MetaTracks - Meeting Summary

Kristoffer L. Nesvik - Glenn S. Skjong - Kristian Støylen

Meeting date: 28.01.2016

Attendance: Kristoffer L. Nesvik - Glenn S. Skjong - Kristian Støylen
Supervisor: Christian A. Strømmen

Summary:
Project group finalizes preliminary report with employer, and discusses what open source code licenses could be used.

GPL license can not be used as it requires code to become open source. LGPL license was recommended as it does not require entire project to be open source.
Development starts after preliminary report is finished.

Project group updates employer on early ideas how fingerprinting algorithm should work. Employer wants each fingerprint to be 250ms or smaller.

MetaTracks - Weekly Report

Kristoffer L. Nesvik - Glenn S. Skjong - Kristian Støylen

Week 5: 01.02.2016 - 05.02.2016

Comments

Week five of the project included a lot of learning and research, mainly in the field of signal processing. Early work on the fingerprint extraction algorithm was carried out, this is hopefully finished during next week. We explored several fingerprint extraction schemes and signal transformation techniques, which will be partially or fully implemented in the algorithm.

Actions planned for this week

- Early work on the fingerprint extract algorithm.
- Brainstorming the fingerprint extraction 'pipeline'.
- Analyze different input data with regards to compression, encoding and sampling rates.

Completed actions this week

- Early draft of the fingerprint extraction algorithm.
- Brainstorming of the fingerprint extraction pipeline.
- Early configuration of the virtual server.

Completed unplanned actions this week

- Short meeting with Hans-Georg Schaatun, discussing signal processing, robust hashing, perceptual hashing, Fourier transformations and "wavelets".

Planned actions for upcoming week

- Implementing a working fingerprint extraction algorithm.

- More research on the underlying theory of signal processing, including robust hashing, perceptual hashing, Fourier transformations and wavelets.

Issues or overdue actions

- Configuration of the virtual server, moved to next week due to still insufficient user rights. Awaiting response from the University virtualization lab.

MetaTracks - Weekly Report

Kristoffer L. Nesvik - Glenn S. Skjong - Kristian Støylen

Week 6: 08.02.2016 - 12.02.2016

Comments

Week 6 of the proejct work consisted of carrying out early work on the fingerprint extraction algorithm and its accompanying front-end application. In short this included expansion of the software stack and source code. For next week, a functional early draft of the standalone application is planned, which we can show the employer.

Actions planned for this week

- Setup of Atlassian JIRA.
- Setup and configuration of virtual Amazon DynamoDB.
- Work on the fingerprint extraction algorithm.
- Work on the standalone fingerprint extraction application.

Completed actions this week

- Setup of Atlassian JIRA.
- Setup and configuration of virtual Amazon DynamoDB.
- Work on the fingerprint extraction algorithm.
- Work on the standalone fingerprint extraction application.

Completed unplanned actions this week

- No unplanned actions were completed this week.

Planned actions for upcoming week

- Working preliminary version of the standalone fingerprint extracton application.

- Working DynamoDB instance.
- Front-end design.
- Meeting with the employer.

Issues or overdue actions

No issues or overdue actions this week.

MetaTracks - Weekly Report

Kristoffer L. Nesvik - Glenn S. Skjong - Kristian Støylen

Week 7: 05.02.2016 - 19.02.2016

Comments

Week 7 of the project work consisted of wrapping up the implementation of the first system prototype. At the end of the week, we held a meeting between the project group and the employer where we presented the work that had been done this far and planned the upcoming weeks.

Actions planned for this week

- Completing the first project ”Sprint”.
- Preparations of the upcoming meeting.

Completed actions this week

- Completed the first project ”Sprint”.
- Got a working prototype for the database management application.

Completed unplanned actions this week

- No unplanned actions were completed this week.

Planned actions for upcoming week

- Start the work on the second ”Sprint”.
- Finish the algorithm for fingerprint generation and database population.
- Start designing the front end application.
- Implement microphone access on the front end application.

Issues or overdue actions

- Some functions relating to the fingerprint generation algorithm was not finished. These actions has been moved to the upcoming week.

MetaTracks - Meeting Summary

Kristoffer L. Nesvik - Glenn S. Skjong - Kristian Støylen

Meeting date: 19.02.2016

Attendance: Kristoffer L. Nesvik - Glenn S. Skjong - Kristian Støylen
Supervisor: Christian A. Strømmen

Summary:
Project group shows first update for implementation. First prototype of early fingerprinting process showed. Using NAudio library to process and split audiofiles with overlap.

Group Shares GitHub repository and JIRA information with employer. Also discussed Scrum product backlog on what to include and leave out.

MetaTracks - Weekly Report

Kristoffer L. Nesvik - Glenn S. Skjong - Kristian Støylen

Week 8: 22.02.2016 - 26.02.2016

Comments

Week eight of the project consisted mainly of continued work on the algorithm and Web API, which is coming along nicely.

Actions planned for this week

- Making the Web API able to query the database for fingerprints.
- Improving the Short-Time Fourier Transform for the algorithm.
- Starting work on a front-end Windows application.

Completed actions this week

- Making the Web API able to query the database for fingerprints.
- Started work on the Windows application.
- Started work on improving the algorithm.

Completed unplanned actions this week

- Preliminary design of the Web API.
- User control for the Web API.
- AWS Server relocation. (Instance moved from Oregon, US to Ireland, EU)

Planned actions for upcoming week

- Work on the fingerprint extraction algorithm.
- Work on the Windows client.
- Source code documentation.

- Early work on the bachelor thesis.

Issues or overdue actions

- The fingerprint extraction algorithm is slightly overdue, but this is to be expected due to the nature of the project. (R&D)

MetaTracks - Weekly Report

Kristoffer L. Nesvik - Glenn S. Skjong - Kristian Støylen

Week 9: 29.02.2016 - 04.03.2016

Comments

Week nine of the project work consisted of expanding on the three current projects making up our solution: The Web API, the Windows front-end application and the "Database Population Application".

Actions planned for this week

- Continue development of the aforementioned sub-projects.
- Preparing for the weekly meeting with the employer.

Completed actions this week

- Implemented spectrogram computation. (majority of this weeks work)
- Designed the database population application front-end.
- Explored different solutions for further development.

Completed unplanned actions this week

- No unplanned actions were carried out this week apartment from general research on signal processing.

Planned actions for upcoming week

- Implement wavelet computation.
- Implement "energy level bins" for the computed spectrograms.
- Work on the bachelor report.
- Source code documentatio.

Issues or overdue actions

No issues or overdue actions this week.

MetaTracks - Meeting Summary

Kristoffer L. Nesvik - Glenn S. Skjong - Kristian Støylen

Meeting date: 04.03.2016

Attendance: Kristoffer L. Nesvik - Glenn S. Skjong - Kristian Støylen
Supervisor: Christian A. Strømmen

Summary:
Showed early version of Web API and progress on fingerprinting algorithm. Using NAudio library to split audiofiles and MatLab to create spectrogram. This process is too slow, so we were told to find a different solution.

Were allowed to use commercial product BASS audio library as this is much bigger and has much more functionality compared to the NAudio library we were using.

MetaTracks - Weekly Report

Kristoffer L. Nesvik - Glenn S. Skjong - Kristian Støylen

Week 10: 07.03.2016 - 11.03.2016

Comments

Week 10 consisted of continuing work on all the current system components. The work was halted somewhat due to an exam the coming week, and preparations for this.

Actions planned for this week

- Improvements to the Web API.
- Work on the Windows and iOS application.
- Improvements to the underlying algorithm.

Completed actions this week

- Improvements to the Windows application.
- Improvements to the underlying algorithm.

Completed unplanned actions this week

- No unplanned actions were completed this week.

Planned actions for upcoming week

- Continue work on the Web API.
- Continue work on the Windows and iOS appliations.
- Continue work on the underlying algorithm.

Issues or overdue actions

- Work on the iOS application is slightly overdue due to build complications.

MetaTracks - Weekly Report

Kristoffer L. Nesvik - Glenn S. Skjong - Kristian Støylen

Week 11: 14.03.2016 - 18.03.2016

Comments

Week 11 mainly consisted of continuing work on the algorithm and related sub-projects including the shared function library / API. A meeting with the employer was planned for the end of the week. During the meeting, we agreed to deliver a functional minimum viable product (an iOS application) at the end of week 13.

Actions planned for this week

- Meeting with the employer.
- Continue work on the algorithm.
- Continue work on the Web API, iOS application and Windows application.
- Continue work on the shared library.

Completed actions this week

- Meeting with the employer.
- Continued work on the algorithm.
- Continued work on the Web API, iOS application and Windows application.
- Continued work on the shared library.

Completed unplanned actions this week

- No unplanned actions were completed this week.

Planned actions for upcoming week

- Continue work on the algorithm.

- Continue work on the Web API, iOS application and Windows application.
- Continue work on the shared library.
- Integration of third-party library "Bass.NET" into the iOS application.
- Robust and/or perceptual hashing implementation.

Issues or overdue actions

- Some minor items are overdue because of the exam.

MetaTracks - Meeting Summary

Kristoffer L. Nesvik - Glenn S. Skjong - Kristian Støylen

Meeting date: 17.03.2016

Attendance: Kristoffer L. Nesvik - Glenn S. Skjong - Kristian Støylen
Supervisor: Christian A. Strømmen

Summary:
Project group showed update on project. Showed better Web API and Early fingerprinting process. Were able to generate early fingerprints at this point, but would still need to be improved a lot to be functional.

Employer bought us a Macbook Pro and we discussed to deliver a functional minimum viable product (iOS application) by the next meeting. We were allowed to use Xamarin.iOS to make C# source code run on iOS.

MetaTracks - Weekly Report

Kristoffer L. Nesvik - Glenn S. Skjong - Kristian Støylen

Week 12: 21.03.2016 - 25.03.2016

-

Comments

Week 12 of the project work will lead up to a working prototype that will be shown to the employer during week 13. Source code refactoring, continued work on the iOS application, web API, algorithm, bachelor report and documentation will be the main focus for week 12.

Actions planned for this week

- Refactoring and cleaning of all subprojects.
- Work on the iOS application.
- Work on the Web API.
- Work on the bachelor report and documentation.
- Source code documentation.
- Work on the algorithm.
- Miscellaneous minor technical work.

Completed actions this week

- Work on the iOS application
- Work on the algorithm
- Miscellaneous technical work

Completed unplanned actions this week

- Some minor work on the front-end design.
- Finished a plethora of unplanned actions related to the algorithm and iOS application.

Planned actions for upcoming week

- Finalizing the working iOS application prototype.
- Finalizing the working API prototype.
- Finalizing the working algorithm prototype.

Issues or overdue actions

No issues or overdue actions this week.

MetaTracks - Weekly Report

Kristoffer L. Nesvik - Glenn S. Skjong - Kristian Støylen

Week 13: 28.03.2016 - 01.04.2016

Comments

Week 13 of the project work consisted of improving the prototype so it will be ready for a small presentation.

Actions planned for this week

- Application improvements, mainly to the iOS application.

Completed actions this week

- Application improvements, mainly to the iOS application.

Completed unplanned actions this week

- No unplanned actions were completed this week.

Planned actions for upcoming week

- Finalizing the 'Theory' part of the bachelor thesis.
- Algorithm reworking and fine-tuning.
- Linking and deploying the iOS application, including unit testing.
- Working on the Web API serialization.
- Cleaning up the source code.

Issues or overdue actions

- Building a working non-simulator iOS application. (A device-ready application)

MetaTracks - Weekly Report

Kristoffer L. Nesvik - Glenn S. Skjong - Kristian Støylen

Week 14: 04.04.2016 - 08.04.2016

Comments

Week 14 mainly focused on improving the applications through testing, debugging and refactoring, and a short meeting with the employer.

Actions planned for this week

- Meeting with the employer.
- Writing the 'Theory' part of the bachelor thesis.
- Algorithm reworking and fine-tuning.
- Linking and deploying the iOS application, including unit testing.
- Working on the Web API serialization.
- Cleaning up the source code.

Completed actions this week

- Meeting with the employer.
- Algorithm reworking and fine-tuning.
- Linking and deploying the iOS application, including unit testing.
- Cleaning up the source code.

Completed unplanned actions this week

- No unplanned actions were completed this week.

Planned actions for upcoming week

- Algorithm fine-tuning.
- Adapting the algorithm for the 'ARM' processor architecture.

Issues or overdue actions

No issues or overdue actions this week.

MetaTracks - Meeting Summary

Kristoffer L. Nesvik - Glenn S. Skjong - Kristian Støylen

Meeting date: 07.04.2016

Attendance: Kristoffer L. Nesvik - Glenn S. Skjong - Kristian Støylen
Supervisor: Christian A. Strømmen

Summary:
Project group showed iOS implementation to employer. This showed the generation of fingerprints and early recognition which was slow because we used an old slow iPad 2. Also showed Database Population App which opens file and creates fingerprints with hashes to be uploaded to database.Employer bought us a iPad Pro to use which was faster than our old iPad we used for testing earlier.

Told to keep improving algorithm while working on implementation of a recognition algorithm. Asked to try make working demo by next meeting, where we also should include Kjell Inge Tomren so he could see our progress.

MetaTracks - Weekly Report

Kristoffer L. Nesvik - Glenn S. Skjong - Kristian Støylen

Week 15: 11.04.2016 - 15.04.2016

Comments

Week 15 consisted of continued development of the iOS application and algorithm. Friday consisted of a meeting with our supervisor and employer, where we agreed to have a working deliverable prototype for the coming week.

Actions planned for this week

- Continue development of iOS application and algorithm.
- Meeting with employer and supervisor.
- Writing parts of the bachelor thesis.

Completed actions this week

- Continue development of iOS application and algorithm.
- Meeting with employer and supervisor.
- Wrote parts of the bachelor thesis.

Completed unplanned actions this week

- No unplanned actions were completed this week.

Planned actions for upcoming week

- Continue development of iOS application and algorithm, leading up to the first deliverable prototype.
- Continued work on the bachelor thesis.

Issues or overdue actions

No issues or overdue actions this week.

MetaTracks - Meeting Summary

Kristoffer L. Nesvik - Glenn S. Skjong - Kristian Støylen

Meeting date: 15.04.2016

Attendance: Kristoffer L. Nesvik - Glenn S. Skjong - Kristian Støylen
Supervisor: Christian A. Strømmen, Kjell Inge Tomren

Summary:
Project Group made a presentation of entire product for Kjell Inge so he could get up to date. Group also showed demo of application. This demo works great on trailers and songs, but for movies and TV-series it has problems generating viable unique fingerprints.

MetaTracks - Weekly Report

Kristoffer L. Nesvik - Glenn S. Skjong - Kristian Støylen

Week 16: 18.04.2016 - 22.04.2016

Comments

Week 16 saw a challenge in that the entire framework codebase had to be rewritten due to the build process of the Xamarin framework. This will probably be the main focus for the remainder of the project, and involves translating the source code from C# to Swift and/or Objective-C.

Actions planned for this week

- Rewrite the codebase.
- Working on the bachelor thesis.
- Code refactoring, fixes and API documentation.

Completed actions this week

- Planning the rewriting process.
- Work on the bachelor thesis.
- Code refactoring.

Completed unplanned actions this week

- Started rewriting the codebase in native iOS-code.

Planned actions for upcoming week

- Continue rewriting the codebase in native iOS-code.

Issues or overdue actions

Following the code rewriting, the entire framework algorithm is overdue.

MetaTracks - Weekly Report

Kristoffer L. Nesvik - Glenn S. Skjong - Kristian Støylen

Week 17: 25.04.2016 - 29.04.2016

Comments

Week 17 mainly consisted of rewriting the codebase in native iOS-code.

Actions planned for this week

- Rewriting the entire C# codebase in native iOS code.
- Work on the bachelor thesis.

Completed actions this week

- Continuation of rewriting the source code.

Completed unplanned actions this week

- No unplanned actions were completed this week.

Planned actions for upcoming week

- Continue the rewriting of the entire C# codebase in native iOS code.
- Work on the bachelor thesis.
- Work on the product poste.
- Source code documentation and refactoring. (preparation for hand-in)

Issues or overdue actions

Following the code rewriting, the entire framework algorithm is overdue.

MetaTracks - Weekly Report

Kristoffer L. Nesvik - Glenn S. Skjong - Kristian Støylen

Week 18: 02.05.2016 - 06.05.2016

Comments

Week 18 revolves around wrapping up everything; finishing the source-code rewriting, finishing most of the bachelor thesis, the product poster, and formatting most of the material for hand-in. This will be the main focus for the coming weeks, as well.

Actions planned for this week

- Source code rewriting.
- Work on the bachelor thesis.
- Work on the product poster.
- Weekly and daily reports formatting.
- Work on the source code documentation.

Completed actions this week

- Work on the source code rewriting.
- Work on the bachelor thesis.
- Work on the product poster.
- Work on the source code documentation.

Completed unplanned actions this week

- No unplanned actions were completed this week.

Planned actions for upcoming week

- Continued work on the source code rewriting.
- Work on the bachelor thesis.
- Finish the product poster.
- Weekly and daily reports formatting.
- Work on the source code documentation.

Issues or overdue actions

Following the code rewriting, the entire framework algorithm is overdue.

MetaTracks - Weekly Report

Kristoffer L. Nesvik - Glenn S. Skjong - Kristian Støylen

Week 19: 09.05.2016 - 13.05.2016

Comments

Week 19 consisted of wrapping up the project, delivering it to the employer, and doing the final steps in documenting the source code. The remaining weeks we will write the system documentation, bachelor thesis, and anything that might be missing or incomplete in the weekly or daily reports. A full system documentation will also have to be written.

Actions planned for this week

- Write the system documentation.
- Work on the bachelor thesis.
- Finish the product poster.
- Weekly and daily reports formatting.

Completed actions this week

- Delivered the rewritten source code to the employer.
- Continued work on the bachelor thesis.
- Continued work on the system documentation.
- Finished the product poster.
- Finished the source code documentation.

Completed unplanned actions this week

- No unplanned actions were completed this week.

Planned actions for upcoming week

- Work on the bachelor thesis
- Work on the system documentation

Issues or overdue actions

No issues or overdue actions this week.

MetaTracks - Weekly Report

Kristoffer L. Nesvik - Glenn S. Skjong - Kristian Støylen

Week 20: 16.05.2016 - 20.05.2016

Comments

Week 20 showed great progress, as the bachelor thesis is coming along nicely. The weekly reports have been formatted, the system documentation was almost finished and the product poster was handed in. In addition, any free time was used to continue work on the native iOS source code.

Actions planned for this week

- Mostly finish the system documentation.
- Mostly finish the bachelor thesis.
- Hand in the product poster.
- Finish the weekly reports and prepare them for hand-in.

Completed actions this week

- Mostly finished the bachelor thesis.
- Mostly finished the system documentation.
- The product poster was delivered.
- The weekly reports were finished, apart from the coming week.

Completed unplanned actions this week

- No unplanned actions were completed this week.

Planned actions for upcoming week

- Finish the bachelor thesis.

- Finish the system documentation.
- Meeting with the employer.

Issues or overdue actions

No issues or overdue actions this week.

MetaTracks - Weekly Report

Kristoffer L. Nesvik - Glenn S. Skjong - Kristian Støylen

Week 21: 23.05.2016 - 27.05.2016

Comments

The last week of an interesting and challenging semester. Everything was finalized, presented and our acoustic fingerprinting library will hopefully be implemented by the employer in the time moving forward.

Actions planned for this week

- Finish the bachelor thesis.
- Finish the system documentation.
- Meeting with the employer.
- Presentation of the bachelor thesis.
- Deliver everything.

Completed actions this week

- Finished the bachelor thesis.
- Finished the system documentation.
- Meeting with the employer.
- The bachelor thesis was presented.
- Everything was delivered on-time in full.

Completed unplanned actions this week

- No unplanned actions were completed this week.

Appendix C: System Documentation Source Code

SYSTEM DOCUMENTATION

MetaTracks

A system for real-time media identification using computer-vision techniques

NTNU

Faculty of Engineering and Natural Sciences

Norwegian University of Technology and Science

Larsgården 2 - 6009 Ålesund - Norway

Spring semester 2016

Contents

1 Introduction

The following system documentation will cover the specifications, purpose, functions and development process of the 'Acoustic Fingerprinting Framework' built as part of the final bachelor project, spring semester 2016.

2 Requirements specification

Our project assignment as it was proposed by MetaTracks have been to develop a mobile application framework with an accompanying API (Application Programming Interface), which take advantage of acoustic fingerprinting to identify which movie or series the user of the application is currently watching.

In addition to identifying which movie or series is currently being watched by the user, the application must be able to identify where the user is in the current playback, while also taking into account the possibility that the user may have paused, skipped in or stopped the playback entirely.

This type of project is a R&D (Research and Development) project, where the feasibility and/or the final product of the project is unknown. Exploration of different technologies and methods, repeated testing and even changing the entire project structure and functionality several times is typical in these types of projects.

3 System purpose and usage

The Acoustic Fingerprinting Framework API will expose the following functions for the user, contained in the static **RecordManager** class:

indexMovies(): Indexes the fingerprint database, giving access to all the movies currently contained in the database. The movies are returned as an array. (currently an array containing Strings)

getFingerprints(string movie): Uses the parameter 'movie' to download the acoustic fingerprints for the given movie from the database.

initialize(): Readies the front-end application for recording. On iOS and OSX systems, this initializes the AVAudioRecorder. On Windows, this initializes the NAudio recorder. The function has to be called before startSyncing(). The BassLoader class is also initialized, readying Fourier transforms, downsampling, mixing and hashing.

startSyncing(): Continuously records, creates new fingerprints and compares the generated fingerprints with the fingerprints downloaded from the fingerprint database. This application will run until stopSyncing is called. The last matched time stamp in the field "LatestTimestamp".

stopSyncing(): Stops the recording and finishes the recognition currently going on before it stops.

4 Legal overview

4.1 BASS Audio Library by Un4seen Development

The BASS audio library is a free for non-commercial use library, which can be licensed for a fee. As of May 2016 the price for the different commercial licenses is:

Shareware License	EUR 125
Single Commercial License	EUR 950
iOS or Androd License	EUR 475
Unlimited Commercial License	EUR 3450

More information about each license can be found on Un4seen's official website.
http://www.un4seen.com/

4.2 Bass.NET

While Bass audio library requires a license, Bass.NET also requires a license. Bass.NET is not used in the final product, but it was used in the development process while using C#. A Bass.NET license will have to be bought in addition to the normal Bass license for commercial products.

Shareware License	EUR 29
Single Commercial License	EUR 199
Unlimited Commercial License	EUR 499

For more information about each license agreement, visit http://bass.radio42.com/

If a special license is needed, a combined BASS and BASS.NET license can be purchased directly from Un4seen official website.

4.3 Exocortex.DSP

Exocortex.DSP is licensed under BSD 2-Clause License. It allows commercial use, modification and distribution of derived work. However the license states Exocortex.DSP can not be held liable for any damages. The license also requires applications to include copyright and full text of license.

```
Following is the license text in full:

Copyright (c) 2001, 2002 Ben Houston [ ben@exocortex.org ]
Exocortex Technologies [ www.exocortex.org ]
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are met:

* Redistributions of source code must retain the abovecopyright notice,
this list of conditions and the following disclaimer.

* Redistributions in binary form must reproduce the above
  copyright notice, this list of conditions and the following disclaimer in
  the documentation and/or other materials provided with the distribution.

* Neither the name of the {organization} nor the names of its
  contributors may be used to endorse or promote products derived from
  this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
(INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

5 Installation instructions

The acoustic fingerprinting frameworks has been built to streamline the import processes, and are self-contained in a single shared library. Following are the installation instructions for Visual Studio with Mono and XCode respectively.

5.1 Through Visual Studio and Mono

To import the library in a Visual Studio/Mono application, one of the following project types are required:

Single View App (Mono)

Master Detail App (Mono)

Tabbed App (Mono)

Bindings or Class library (Mono)

Any Visual C# front-end, including WPF, Windows Forms or Console Application.

In any of the projects listed above, reference the AcousticFingerprintingLibrary.dll or the AcousticFingerprintingLibrary shared project. Following is a working code snippet importing and using the exposed library functions.

```
using System;
using AcousticFingerprintingLibrary;

namespace ConsoleApplication
{
    class Program
    {
        static void Main(string[] args)
        {
            using (RecordManager rm = new RecordManager())
            {
                // Initializes the RecordManager
                rm.initialize();
                // Indexes the movie database, saving results to indexedMovies
                var indexedMovies = rm.indexMovies();
                // Get the fingerprints for the first movie in the database
                rm.getFingerprints(indexedMovies[0]);
                // Start the fingerprint matching and recording
                rm.startSyncing();
                // Break when any match is found
                if (rm.LatestTimeStamp != null)
                {
                    rm.stopSyncing();
                }
            }
        }
    }
}
```

5.2 Through XCode

To import the framework into an XCode workspace, one of the following project types are required:

Single View App (iOS or OSX)

Master Detail App (iOS or OSX)

Tabbed App (iOS or OSX)

Open the target Build Settings and reference the AcousticFingerprintingLibrary.framework in the Embed Binaries field. Following is a working code snippet import and using the exposed library functions.

```
import AudioRecognitionLibrary

class ViewController: UIViewController,UITableViewDelegate,
    UITableViewDataSource {
    let manager : RecordManager = RecordManager()

    override func viewDidLoad() {
        super.viewDidLoad()
        // Initializes the RecordManager
        RecordManager.initialize()
        // Indexes the movie database, saving results to indexedMovies
        var indexedMovies = RecordManager.indexMovies()
        // Get the fingerprints for the first movie in the database
        RecordManager.getFingerprints([indexedMovies[0]])
        // Start the fingerprint matching and recording in a background
            thread
        let qualityOfServiceClass = QOS_CLASS_BACKGROUND
        let backgroundQueue =
            dispatch_get_global_queue(qualityOfServiceClass, 0)
        dispatch_async(backgroundQueue, {
            RecordManager.startSyncing()
        })
}
```

6 System architecture

6.1 UML diagram - Acoustic Fingerprinting Library

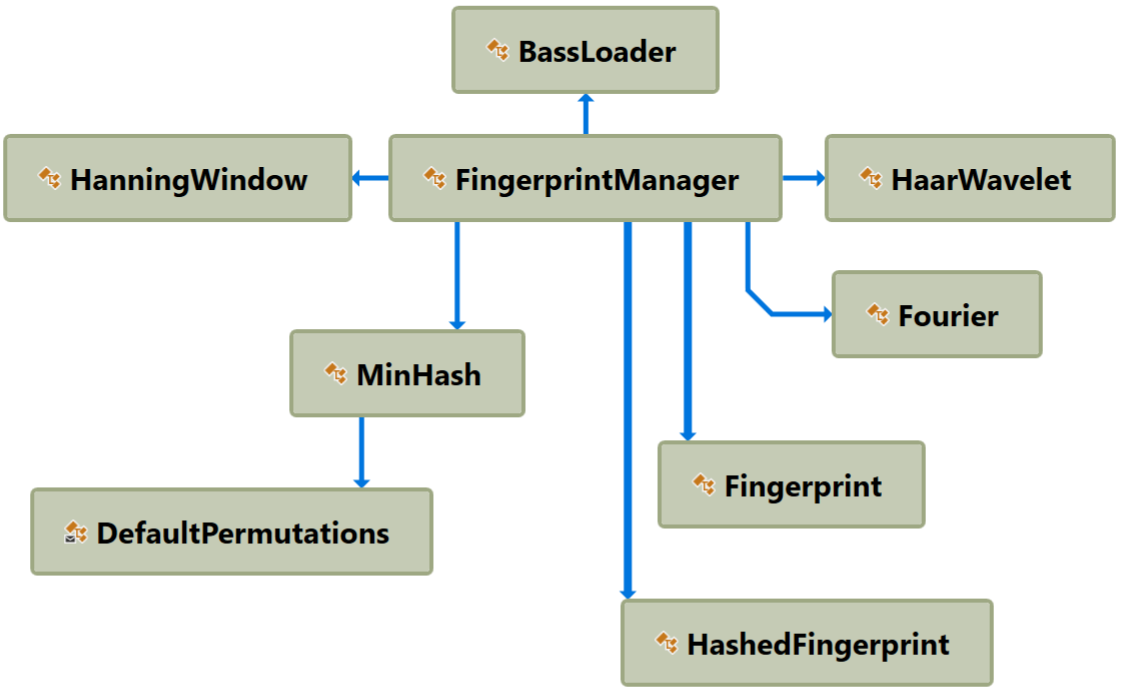


Figure 6.1: Class Diagram

6.2 Class Descriptions

6.2.1 FingerprintManager

The acting 'main'class and entry point. Makes use of all other classes to create and compare fingerprints.

6.2.2 BassLoader

Hadles everything related to BASS, including the resampling and decoding of audio files.

6.2.3 HaarWavelet

Handles the Haar Wavelet Transformation, used in the generation of acoustic fingerprints.

6.2.4 Fourier

Derived from the Exocortex.DSP library source code. Handles Fast Fourier Transformation calculations used for creation of audio spectrograms.

6.2.5 Fingerprint

An object class for storing fingerprint signatures, time stamp and sequence indicator variables in key-value pair.

6.2.6 HashedFingerprint

Am object class for storing **hashed** fingerprints. Includes the same fields as the Fingerprint class.

6.2.7 MinHash

Handles the generation of our Min-Hash and LSH grouping of hashes.

6.2.8 DefaultPermutations

Includes preloaded hash permutations. We decided to store the permutations in a class rather than a CSV file for ease of use.

6.2.9 HanningWindow

Function for generating the Hanning Window used prior to the Fourier Transformation.

7 System functions

7.0.1 Acoustic Fingerprinting Framework

The Acoustic Fingerprinting Framework is built to run on most modern systems, outlined in the following section.

Hardware

Supported architectures include:

- x86_64
- i386
- arm7
- arm64

Operating Systems

Supported operating systems include:

- Windows 7, 8, 8.1 and 10
- Mac OS X 10.9 and up
- iOS 9.2 and up

Libraries and frameworks

The Acoustic Fingerprinting Framework package all libraries and frameworks required for building the full library. These include:

- BASS by Un4seen Development
- BASSMIX by Un4seen Development
- AudioToolBox Framework (iOS and OSX only)
- SystemConfiguration Framework (iOS and OSX only)
- CFNetwork Framework (iOS and OSX only)
- Accelerate Framework (iOS and OSX only)
- Bass.NET (.NET wrapper for BASS)
- Xamarin iOS library (For building the iOS application through Mono)

Build environment

There are currently two options for building the Acoustic Fingerprinting Library:

> Through Visual Studio, the framework can be built for deployment to applications built under the Mono runtime ('Xamarin') framework.
>
> Through Apple's XCode, the framework can be built natively.

7.1 Source code

For brevity, the source code is included in the delivery and not in full in this document.