Bird Recognition using Audio Fingerprinting

Author: Laurin Brandner, G4L

Supervisor: Martina Vazquez

October 25th, *2013*

Maturaarbeit Alte Kantonsschule Aarau

Abstract

This project is all about creating a software that is capable of recognizing birds by their song. The first part mainly focuses on the theory of audio fingerprinting, the procedure of modifying or analyzing audio data. Here, a closer look will be taken at the different algorithms, including the Fast Fourier Transform and the Haar-Wavelet Transform, used to do so.
In the second part two audio fingerprinting algorithms are developed. One of them is then used in a bird recognition service called Whistles which includes an iPhone application, a server and a database. The aim of this service is to create a user friendly, functional and convenient application. The general architecture of such a product and the requirements which need to be met are then discussed. Thereafter, the limitations of this architecture are specified.
The last section looks at the results of the project, how they can be interpreted and ways to improve the project in the future.

Acknowledgements

First of all, I want to thank Mrs. Vazquez, my supervisor who has supported me throughout this project and guided it and myself in the right direction.
I also want to thank Mr. Jordi who generously lent me some of his physics books to help me understand the algorithms used in this project.
Very importantly the ornithological station Sempach for supplying the birdsong recordings used in this project. I thank my family as well for supporting me throughout this project.

Contents

1 Introduction

1.1 Preamble

Nowadays, the internet is ubiquitous. We use smartphones, tablets and computers to keep connected to friends, find entertainment and gather knowledge. In the early days, computers were only accessible through a terminal. Throughout the last couple of decades the technology has evolved tremendously. We have now been using computers with flexible and beautiful user interfaces for quite some time. However, ever since computers have existed, engineers have tried to connect them more tightly to the users. They have done so by developing touch screen, speaking assistants and so on. The advantage of these interfaces is that they use the communication medium we are more familiar with. Developing an application that recognizes birds by their whistling is exactly the same thing, meaning you would not expect a device to recognize an animal just by the noises it makes. That is why it seems a little like magic.

1.2 Problem Statement and Motivation

Surely everybody was once in the woods and has heard a bird singing. Although we actually know many birds by their appearance and we know some popular noises they make, the majority of people cannot associate a bird to its individual whistling melody. This project addresses this problem.
The primary goal of this project is therefore to create an intuitive application that is able to tell what kind of bird is singing. This is a very ambitious goal. Therefore, only ten birds are used to develop the algorithm. The application which is the main product of this service is supposed to be functional and well designed.

1.3 Objectives

1.3.1 Main Objectives

Audio Fingerprinting

To develop a program that is able to generate comparable data from a bird's whistling.

iPhone Application

To develop an iPhone application which makes use of the audio fingerprinting program.

Server and Database

To implement a server and a database. The server is responsible for the communication and the comparison of the data, the database is needed to store this.

Recognition

The service should be capable of recognizing ten birds by their song.

1.3.2 Additional Objectives

Pause and Resume Functionality

Being able to pause and resume the recording could improve the accuracy of the algorithm.

The recognition algorithms can be extended as much as possible as an additional objective as it is such a comprehensive goal.

1.4 Overview

Before we take a closer look at the theory behind this project we need to define what will later be referred as an identification cycle. An identification cycle is the process from recording a bird on an iPhone to a response from the iPhone that identifies the recorded bird.

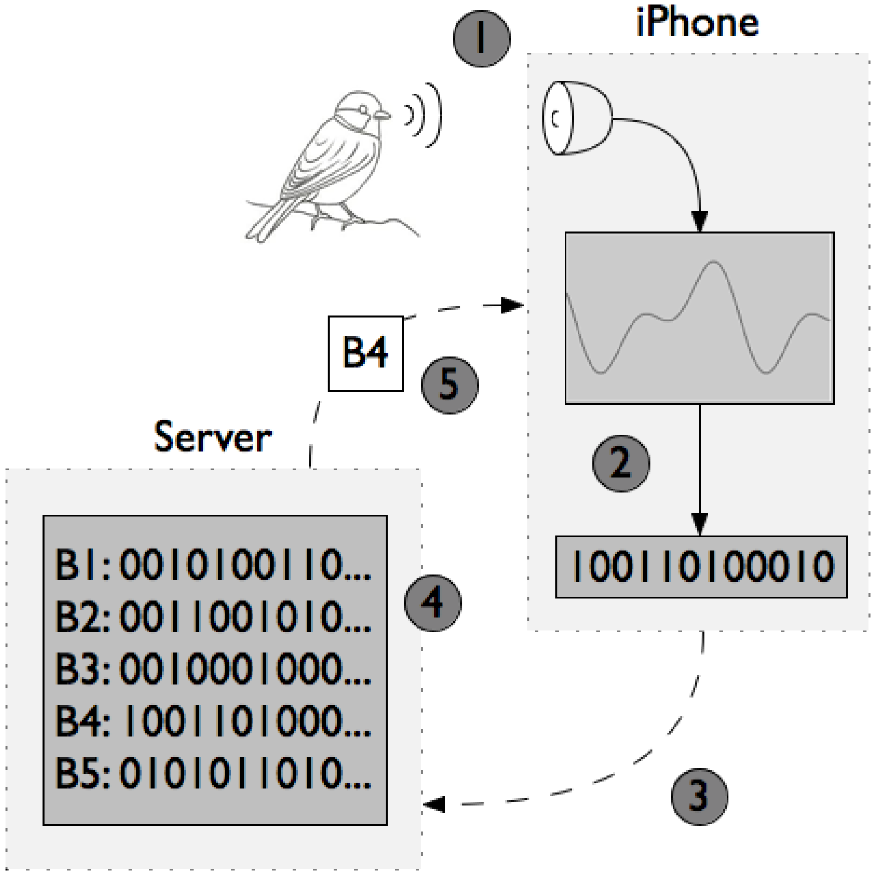


Figure 1: One identification cycle

Figure 1 illustrates the procedure to identify a bird by its whistling. The dotted rectangles indicate the devices while the dashed arrows show the communication between them through the internet. The dark gray rectangles are the most important parts of the recognition algorithm.
Fist of all, the bird is recorded using a microphone which is built or plugged into the iPhone (step 1). The output of the microphone is then processed. From the big and hard to compare raw audio data a fingerprint is generated (step 2). This fingerprint is less data and thus easier to compare and convey. The fingerprint is then sent to the server over the internet (step 3). The server accesses its database, which contains birds and the belonging fingerprints. It goes through every fingerprint and compares it to the received one in order to evaluate the bird that matches best (step 4). It then returns this bird, which in this example turns out to be bird 4 (B4), and sends it over the internet back to the client (step 5). The client, in this case an iPhone, can store and display the result.

2 Theory

2.1 Audio

2.1.1 Acoustics

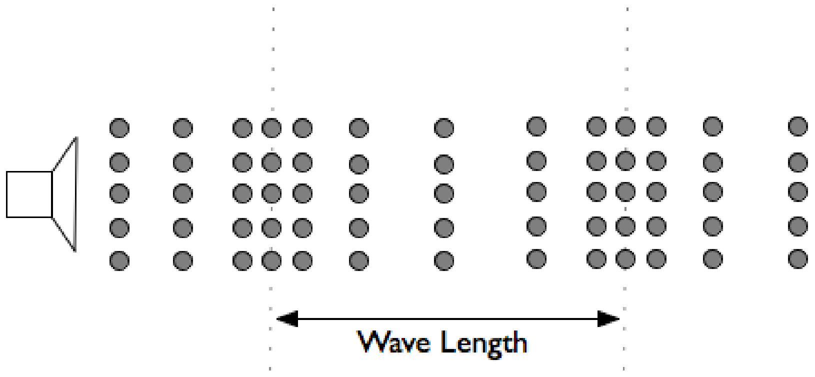


Figure 2: An acoustic wave generated by a speaker

Generally speaking, sound is the audible motion of small particles in a fluid, no matter what state of aggregation it is. However, only air is considered as it is the most relevant fluid for this project.
As an object or a membrane starts vibrating, the surrounding particles start moving along with its motion. These particles will crash into other particles which will carry on doing the same thing until the energy of the movement is too small to affect its surroundings.
The original vibration has been transferred by the fluid to its environment. If someone was close enough to the audio source, he would hear the noise.
Due to the fact that the sound propagates in the form of a wave there is a change in density of the air. Because this change occurs in a distinct interval an offset is noticeable. This offset between the highest densities is called the wavelength[1, p. 10]. The shorter this offset the higher the frequency. We perceive a high frequency, a fast alternation of denser and rarer fluid, as a high tone. The average audible range of a human being is between 20Hz and 20 kHz (20000 Hz). What we perceive as a higher volume, a louder tone, is an increased pressure on the ears. The more violently the object vibrates the higher this pressure is.

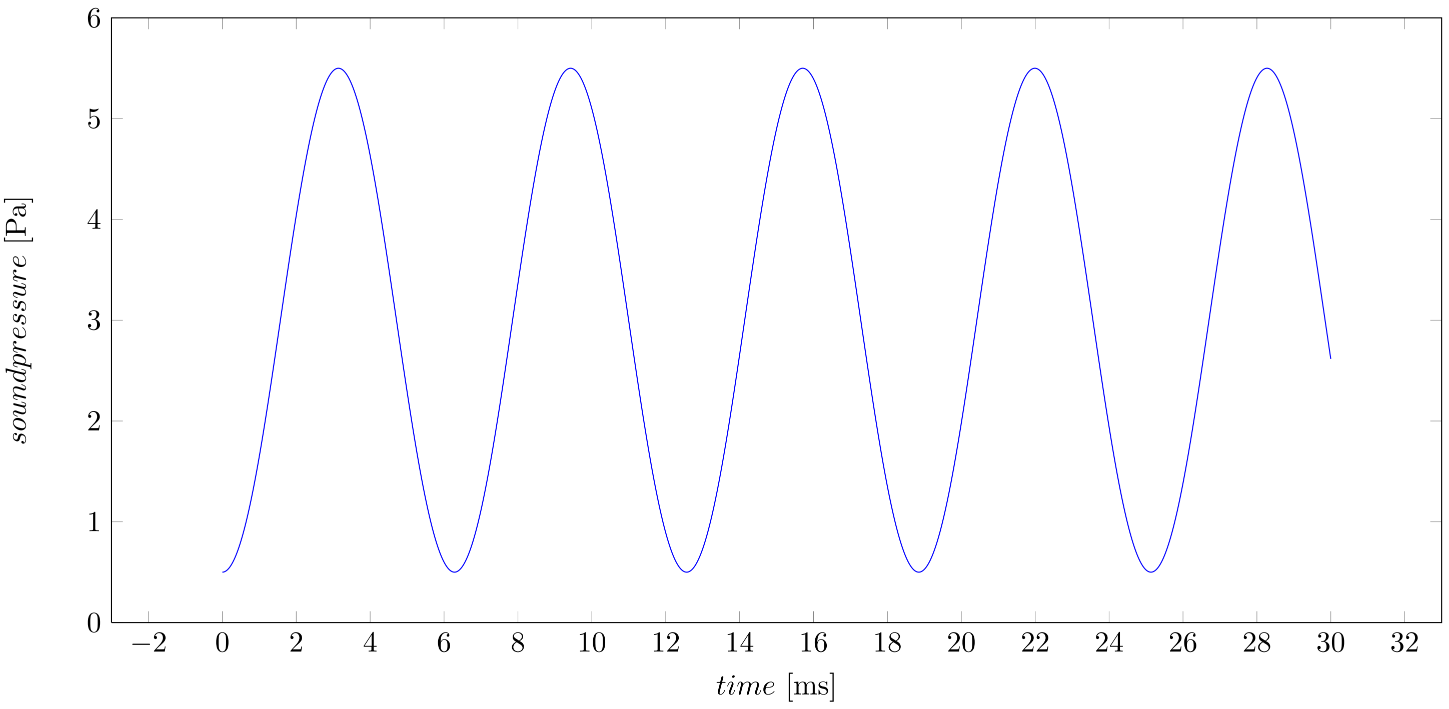


Figure 3: A graph showing the sound wave of Figure 2

2.1.2 Microphone

Although the microphone seems to do exactly the same thing as as our ears it works differently. It contains a thin membrane that moves just like the particles in the air. The microphone then can translate this motion into signals.
It is crucial for understanding the following algorithms that a computer which is recording with a microphone plugged in does not store a certain frequency at a given time but rather a certain value of pressure. This means that the computer never gets to know what pitch it has recorded.

2.1.3 Digital Audio

A computer saves audio signals by virtually plotting them on a graph.

Diagram 4 shows a PCM (pulse code modulation) representation of an audio signal. PCM is the standard format in digital audio computing for it is the raw digital format. PCM plots samples (a specific amplitude at a given time) at a uniform interval. The smaller this interval the more accurate is the resulting audio file[2, p. 27]. The problem in audio programming is that sound is analogue which means that the data is continuous. An infinite number of samples would be required in order to fully represent one audio signal. This is obviously not possible. Thus an audio signal rather needs to be estimated than actually copied. But what sample rate (samples per second) is needed to estimate the signal adequately?

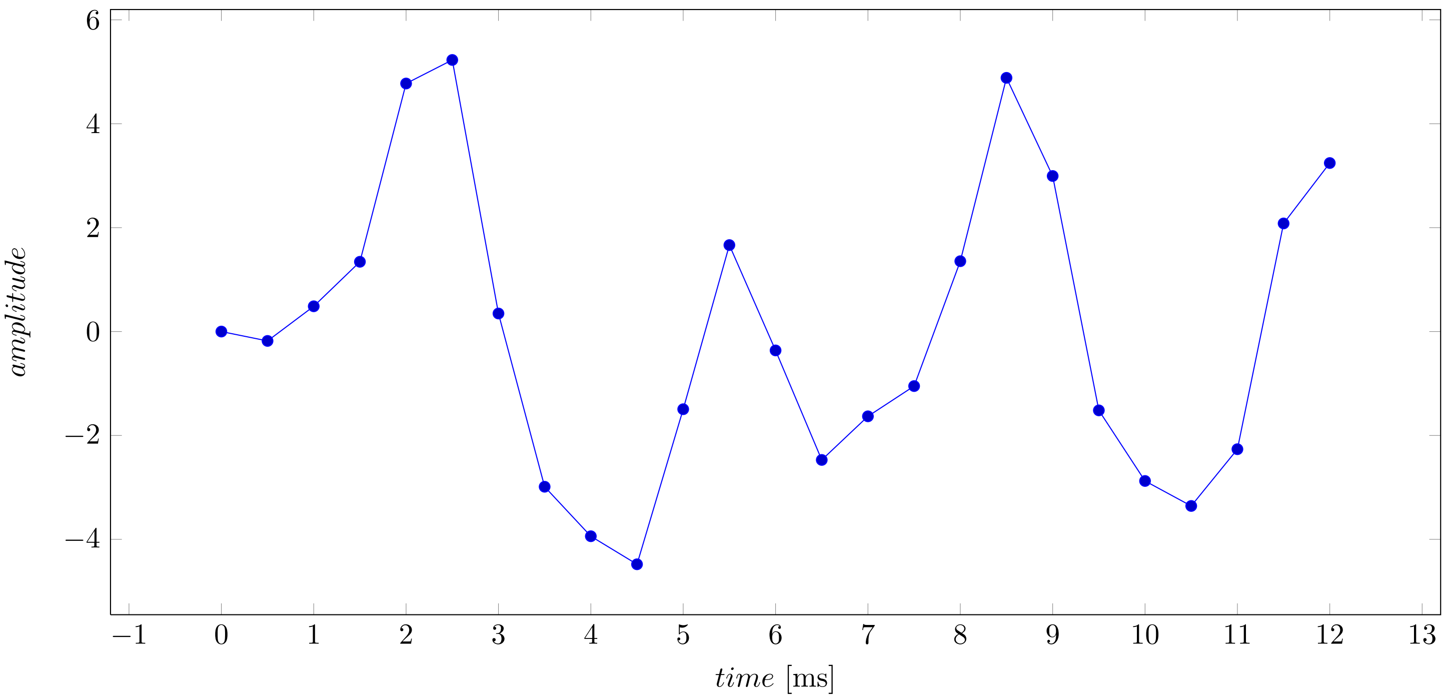


Figure 4: A sound wave in PCM

This is where the Nyquist-Shannon Sampling Theorem[3] helps. This theory is defined as follows:

If a function x(t) contains no frequencies higher than B hertz, it is completely determined by giving its ordinates at a series of points spaced 1/(2B) seconds apart.

Because 20 kHz is the highest perceivable frequency for human beings a sample rate (measured in samples per second) of 44100 is sufficient. 44.1 kHz is the default sample rate that is used to save audio files. Besides the sample rate there are other important properties that make up the audio file:

Channels

Channels are separate streams of samples that are call all be played at the same time. In order to play back stereo or surround sound channels are needed.

Frame

A frame is a bundle or package that contains the samples of all channels at a given time.

Bit rate

The bit rate defines the amount of data saved over a certain amount of time. It depends on how much data is used for one sample and at what sample rate the audio file is saved.

In Figure 5 a song with 2 channels is represented. For every millisecond there is one sample saved per

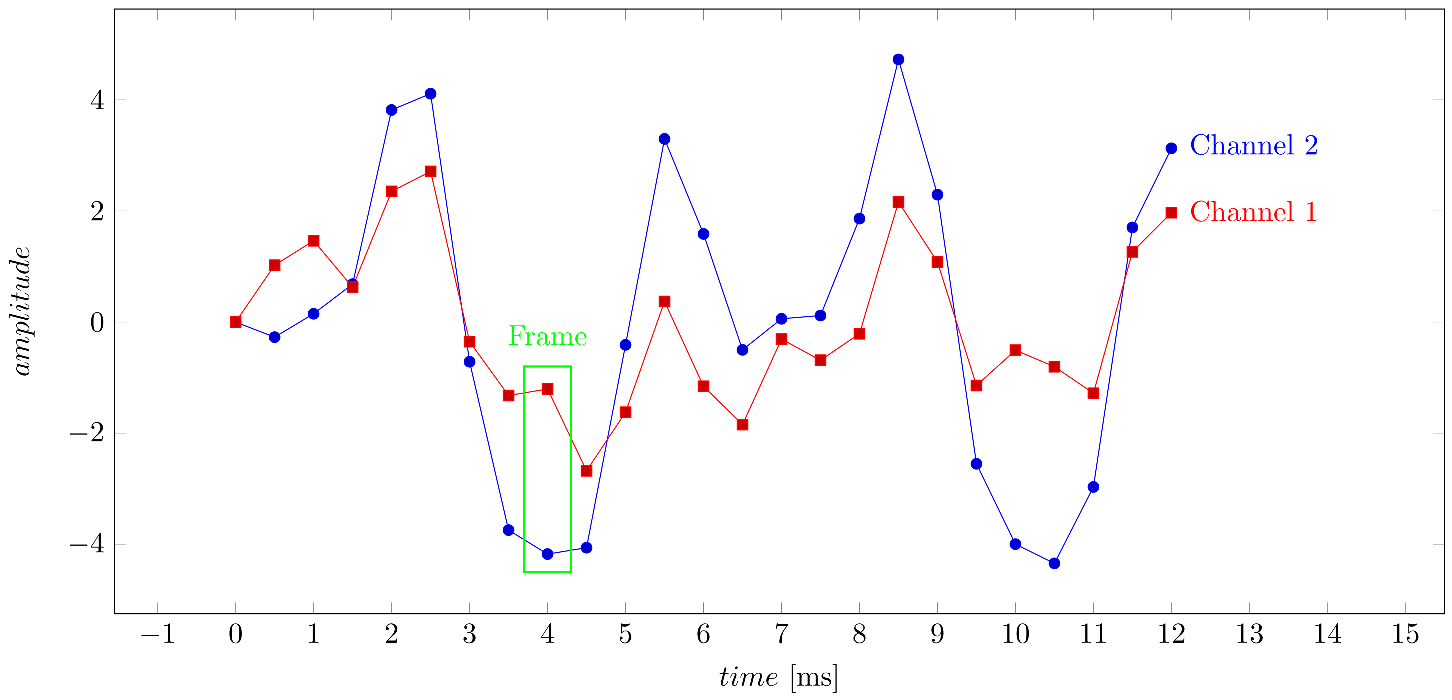


Figure 5: A song in PCM with 2 channels

channel. Hence, the sample rate is 1000 as there are 1000 samples per second. If one sample holds 16 bits of data, the bit rate would be 16 kilobits.

There are many different audio formats (e.g. MP3, WAV) that store the data differently. Although they share the same idea of the graph representation they make use of different algorithms to store information more efficiently (e.g. variable sample rates). They all have their advantages and disadvantages as they serve different purposes.

The software of this project mainly supports PCM for it makes it easier to implement and has hardly any performance loses.

2.2 Whistling

Birds are said to have the most diverse and to us humans most important voice although we are not even capable of hearing all of it. A bird often whistles more than 200 tones per second. This is too fast for our ears.
This immense speed allows the animals to transmit plenty of information. They use it to interact socially with other individuals.
Bird watchers have classified the different purposes of their twitter[4, p. 184]:

Breeding

Not only singing birds but all birds sing during the breeding season which takes place from February to June. More precisely only the males sing. It is usually used to woo a female and to defend territory.

Communicating

Throughout the whole year birds use their whistling to communicate with each other. They tell each other what they are doing, they indicate the presence of food and they just greet each other. Birds are believed to be able to identify each other by their song.

Begging

The females and later on the hatched chicks ask the males for food using begging calls. Chicks only make these noises until they are fledged.

Warning

Birds use warning calls in two situations. They either warn another bird of a threat like a cat or a human, or they threaten another individual.

The whistling we usually associate a bird with is the one used during the breeding season and this is the category chosen for the purposes to this project.

2.3 Audio Fingerprinting

Just as a fingerprint has its own unique properties, so does a music track or an audio signal in general. Thus audio fingerprinting is the process of generating and comparing audio signals, not by their metadata like title, artist or duration but by their actual content[5].
Jaap Haitsma [6] has stated that a fingerprinting algorithm can be defined using 5 parameters:

Robustness

The robustness of the algorithm depends on how much background noise can be handled without yielding an incorrect result.

Reliability

The reliability is the average correct result, also known as the false positive rate. It describes the comparison accuracy without any sound degradation.

Fingerprint size

The size of the fingerprint has a tremendous impact on the performance of the matching/comparison process, whereby a too small fingerprint holding too little data influences the reliability in a negative way.

Granularity

The granularity is the duration of a given audio signal used to find the matching result. In this project it refers to the duration of the recordings.

Scalability

Considering this project, scalability means the ability to increase the number of birds that could be identified whitout compromising the reliability and performance.

There are many different techniques and algorithm that are used to achieve this. The following sections describe the most crucial algorithms used in this project. These algorithms usually serve a wider range of applications than just audio analysis. Because in nature, everything, like color and sound, is based on frequencies, these algorithms are normally used to analyze things that represent the real world. Examples would be pictures, ECG waves, exe scanners and so on.

2.3.1 Fast Fourier Transform

Jean Baptiste Joseph Fourier published his work "The Analytic Theory of Heat" in 1822. This book containing the basic ideas of the Distinct Fourier Transform had a huge impact on the world of mathematics and physics as it was useful in many areas. Nowadays, a similar algorithm sharing the same principles and ideas as the Distinct Fourier Transform, is ubiquitous in computer science. It is called the Fast Fourier Transform (FFT) and has its origin by the german mathematician Carl Friedrich Gauss in 1805. The FFT was revolutionary to computing because its speed allowed previously impossible problems to be solved within minutes[7, p. 117].

Applying a Fourier Transform to a function reveals information about the frequencies and their amplitudes of this function. It does so without losing any information. It only translates the function from the time domain into the frequency domain and vice versa. This implies that the original and the transformed graph contain exactly the same information but show them differently. This makes it tremendously useful as a computer considers sound to be a set of samples while we consider it to be a set of frequencies. The FFT is therefore an algorithm that connects these two conflicting aspects.

The FFT is based on the following thesis:

Every periodic function can be described by multiplying sines and cosines.

Considering 1000 samples representing a miscellaneous recording. The FFT extracts every frequency and their amplitude. Because the number of samples is fixed, the number of frequencies the data is capable of representing is fixed as well. The sinusoid with the highest possible frequency has 500 peaks and 500 valleys. The lowest frequency is exactly one peak from the first sample to the last.
All sinusoids essentially have to cross the x-axis in the first and last sample in order to be representable by 1000 samples. It turns out that the number of sine waves that meet this requirement is equal to the

number of samples. To obtain a full analysis, the amplitude of those sinusoids is necessary. Thus, all of them are compared to the unit sinusoid, the sine of x[8].

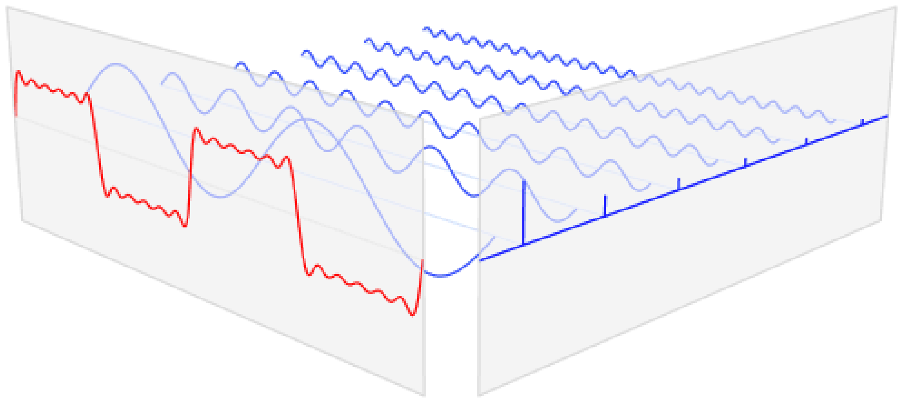


Figure 6: The Fast Fourier Transform applied on a signal (Barbosa, Lucas V., 2013)

Figure 6 shows the summarized procedure of a FFT. On the left plane, the original signal is shown. On the right plane the final result is visible. It first decomposes a signal into every single sine or cosine wave and then sorts them according to their frequency along the axis.

Taking a closer look at the right plane it is obvious that it does not reveal any information about the time. Thus after having a Fourier Transform applied to a function it is not possible to know at what time a frequency was played. Fortunately, there is a method to minimize this issue. In order to estimate the time a frequency occurs the Fourier Transform needs to be applied on a short sequence, a window. The smaller the window, the better the time estimation. However, small windows are generally speaking more expensive to calculate. The product of a Fourier Transform with a sliding window is called a spectrogram.

Unlike the Fourier Transform, a spectrogram holds information about the magnitude, the time and the frequency. These properties are plotted on either a three dimensional graph or a two dimensional graph which uses colors to map the third property like Figure 7.

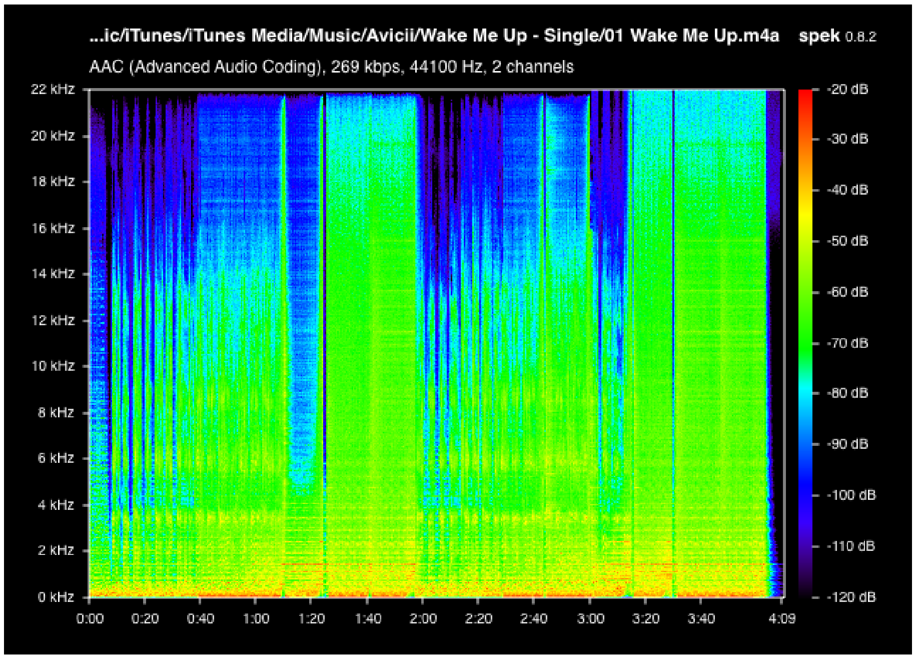


Figure 7: A spectrogram of the song Wake Me Up by Avicii

The FFT is fast because it needs less operations for the same result as the Distinct Fourier Transform, the original algorithm. The following formulas show quite clearly that the FFT can be a lot faster, especially when the data set contains millions of points. The formulas describe the number of operations needed (o) for the number of input points(n).

$$o = n^2$$

Figure 8: The number of operations needed for the Distinct Fourier Transform

$$o = nlog(n)$$

Figure 9: The number of operations needed for the Fast Fourier Transform

2.3.2 Haar Wavelet Transform

The Haar Wavelet Transform, developed by Alfred Haar, is an image processing and compression algorithm. It abstracts the input data which results in a less detailed representation. This makes it easier to compare the data as it applies a kind of filter as that neglects details and emphasizes outstanding properties and patterns[7, p. 150]. There are many variations of this algorithm so we are only going to discuss the one used for this very project.

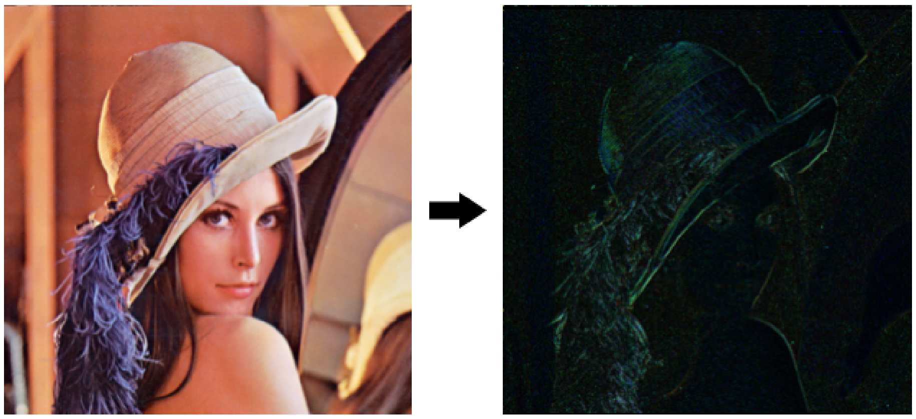


Figure 10: A Haar Wavelet Transform applied to an image of Lena (Wikipedia, 2010)

Figure 10 shows what the outcome of the Haar Wavelet Transform is. The algorithm itself is actually quite simple and is based on a method called averaging and differencing[9].

$$P = \begin{pmatrix} 538 & 940 & 1940 & 1794 \\ 1840 & 213 & 1320 & 913 \\ 192 & 591 & 492 & 1921 \end{pmatrix}$$

Figure 11: A simple image consisting of 4x2 pixels

To explain how it works, consider the matrix in Figure 11 which represents an image. We treat every row like a string and perform the averaging and differencing method mentioned above. The same is then done to the columns.

538	940	1940	1794
739	1867	**-201**	**73**
1303	**-564**	**-201**	**73**

Table 1: Averaging and differencing the first row of Figure 11

Table 1 indicates that 2 steps were required to average and difference the row because one row has a length of $4 = 2^2$. The first row is a copy of the first row of Figure 11. Consider the first row as two pairs of values. The first value of the second row is the average of the first pair. The second value in the second row is the average of the second pair. The bold values are called the detail coefficients. They are the offset from the averages and do not change for the subsequent rows within their column [9].

$$-201 = 538 - 739$$

Figure 12: How the third value of the second row is calculated

$$T = \begin{pmatrix} 969 & -249 & 177 & 80 \\ 95 & -212 & -293 & -38 \\ 461 & -235 & -81 & -292 \end{pmatrix}$$

Figure 13: The transformed matrix P

The resulting matrix is in real life abundant in zeros. That is why the resulting image of Figure 10 is almost black.

2.4 Programming Languages

There are many different and elaborate programming languages that have been developed in the last 40 year[10]. This section will give some further information on the programming languages I have used to implement this project.

2.4.1 C

The C programming language was developed by Dennis Ritchie and released in 1972[11, p. 6]. Considered by programmers as a "low-level language", one of its key properties is that it serves well for many different purposes and provides a lot of control at the same time. Due to its performance it is most often used in the core of libraries. Because C is very common it is supported by numerous devices.

2.4.2 Objective-C

Objective-C is a superset of C. It has its roots in the 1980's and was developed by NeXT, a company that was bought by Apple[12, p. 33]. Objective-C inherits all the properties and features of C and combines them with its own traits in order to ease the development of an application for Apple's platforms, the Mac and the iOS devices. One of the most crucial characteristics of Objective-C is that its an object oriented language. This type of language considers modular parts of a program as sovereign objects. But this simplification leads to a loss of control and performance.

Apple provides powerful and comprehensive frameworks written in C that address complex problems like signal processing, and graphics rendering in a very performant way. Because of the tight connection between C and Objective-C the source code may mix the two languages without difficulty. This makes it easy to use the specific advantages for a given problem.

2.4.3 Ruby

Developed by Yukihiro Matsumoto, Ruby was released in 1995 which makes it a fairly young language[13, p. 2]. Although its syntax is inspired by C, the language hardly has anything in common with it and hence with Objective-C. Moreover, Ruby is not really a programming language but a script language and as the creator states, Ruby might seem simple but is very complex in its implementation[14].

Rails, a web development framework based on Ruby, has become very popular in the last 5 years. In web development everything needs to be very safe and efficient as many users might request the same data simultaneously. Because Rails addresses these issues it has proved very helpful.

2.5 Server

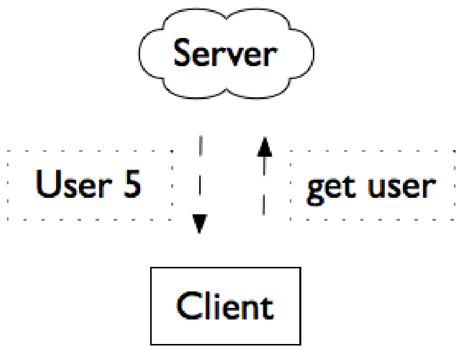


Figure 14: RESTful server architecture (Rodrigues, Alex, 2008)

In this particular situation a client, which may be an application or a program, requests information about a user from a server. It can send this request by calling a URL specified by the Application Programming Interface (API) of the server. The API can be compared to a protocol that defines what resources may be accessed using which URL. RESTful is the dominant architecture used for web services. It is defined by its distinct separation of the client and the server and its lack of states.

A server is basically a computer that allows clients to communicate with services hosted on a network. Every computer can serve as a server, but it is usually a computer that has been designed to handle many requests asynchronously. Severs usually have access on a database where they can save information used for their service. For the situation in Figure 14, the server has information about the users saved which it can access and send back to the client. Responses are usually sent in a universal format allowing many different clients to parse the data in spite of the programming language used to develop them. The format the response is parsed in depends on the data. Websites are usually parsed in HTML while user preferences could be JSON or XML.

Method	Get User Information
URL	`http://server.ch/user.json`
Response	`[{id:5,name:'Brandner',prename:'Laurin'}]`

Table 2: A possible Application Programming Interface for the server and its response in JSON

Table 2 shows a possible API of the server. The server responds with a dictionary parsed in JSON. In order to actually send this request parameters appended to the URL might be required to authenticate the user.

Generally speaking, the internet is nothing but a huge network of various computers and servers. Every time a user visits a webpage, the internet browser sends a request to a particular server. The server then sends back the content of the website.

3 Development

3.1 Requirements

In order to develop an application which is capable of recognizing birds by their whistling, it is essential to define the different features of the application and the properties of the identification algorithm. These features are separated into qualitative requirements which are needed for a working application and functional requirements which are additional features that improve the results.

3.1.1 Qualitative Requirements

Bird species whistle approximately the same theme amongst each other. Their melody is actually only a certain frequency for a given time. Thus the program needs to be able to record audio signals. As stated in section 2.1.2, a microphone does not record the recorded signals as pitches. Therefore the application has to use a Fourier Transform so it can estimate the frequencies.
Birds are most likely found and recorded outside in a forest or a city where there are many background noises. Unlike the human ear, a microphone is not able to focus on a specific audio source, therefore the application has to decide what audio signals are taken into account. Since the user will most likely try to identify the loudest bird the audio signals with the highest magnitude are used. Other background noises that might worsen the result ought to be neglected by using different algorithms and techniques to filter them out.
Another attribute of the algorithm should be its time independence. The user might have started recording when the bird is already singing. Being time independent, the algorithm is able to produce the same result neglecting the difference in time.
The most difficult part of an identification is not the calculation of arbitrary data and make it comparable but to actually compare the data with a robust manner. This issue is addressed by research and experiments.
There are many arguments that speak for an online comparison. A local comparison, a comparison on the device the data is to be compared on and was previously recorded and calculated with, is disadvantageous in terms of storage efficiency, maintenance and scaling. The unknown bird which is to be identified is compared to a set of known birds. The data of these birds has to be stored somewhere. Hosting them in the internet on a server makes it more efficient since the same information set is saved only on one device, the server, and not on every device the application is installed on. If the comparison of the birds takes place on a server it would also be easier to maintain the algorithm and fix bugs. Scaling, extending the range of known birds, would be less difficult as well.
Because the algorithm is supposed to solve a real life problem, it should also expect real life situations. Birds never sing exactly the same way. This means that the algorithm should produce the same result

for slightly differing audio signals from the same species of bird.

3.1.2 Functional Requirements

The main product of this project is an application. This application should be practical to everybody hence a simple and user friendly graphical interface is very important. The application should only contain the most important features so that recording and identifying is as easy as possible. It should be obvious to everybody what component what action performs making the overall experience intuitive.

For the comparison of two music tracks it is fair to assume that these signals may be equal. They sound the same no matter which speakers are used. However, assuming that two audio signals originating from a bird could be equal would be misleading. The probability that a bird actually whistles two times in exactly the same manner is very small. The complexity of this problem further enhances when considering that a bird might suddenly stop singing. Continuous recording would spoil the data as it would keep adding useless information that does not belong to a bird. Thus the ability to pause and resume the recording would improve the recall ratio of the algorithm as a continuous flow of useful data is ensured.

These complicated calculations, the usage of internet connections and user manipulation of the graphical interface in general may lead to errors occurring. Failures of little importance should be concealed as far as possible, but the user should be notified about fatal errors the application cannot resolve and in this way increases the user's understanding and trust of the application.

Unfortunately, the recognition algorithm will never be perfect, it will only be a matter of time until the result is wrong due to either a bug or a poor recording. When this happens, users should be able to inform the developer of the indicated errors as easily as possible. This would facilitate the scaling and improvement of the project.

The microphone of mobile devices like the iPhone are designed to be used for phone calls. In order to improve the quality of the recordings better microphone accessories can be plugged into the cell phone. The application should make use of these if desired.

3.2 Implementation

In this section, a closer look will be taken at the way this project was implemented. Figure 15 depicts the interactions between the various classes of the iPhone and the server in order to be able to identify a bird's song.

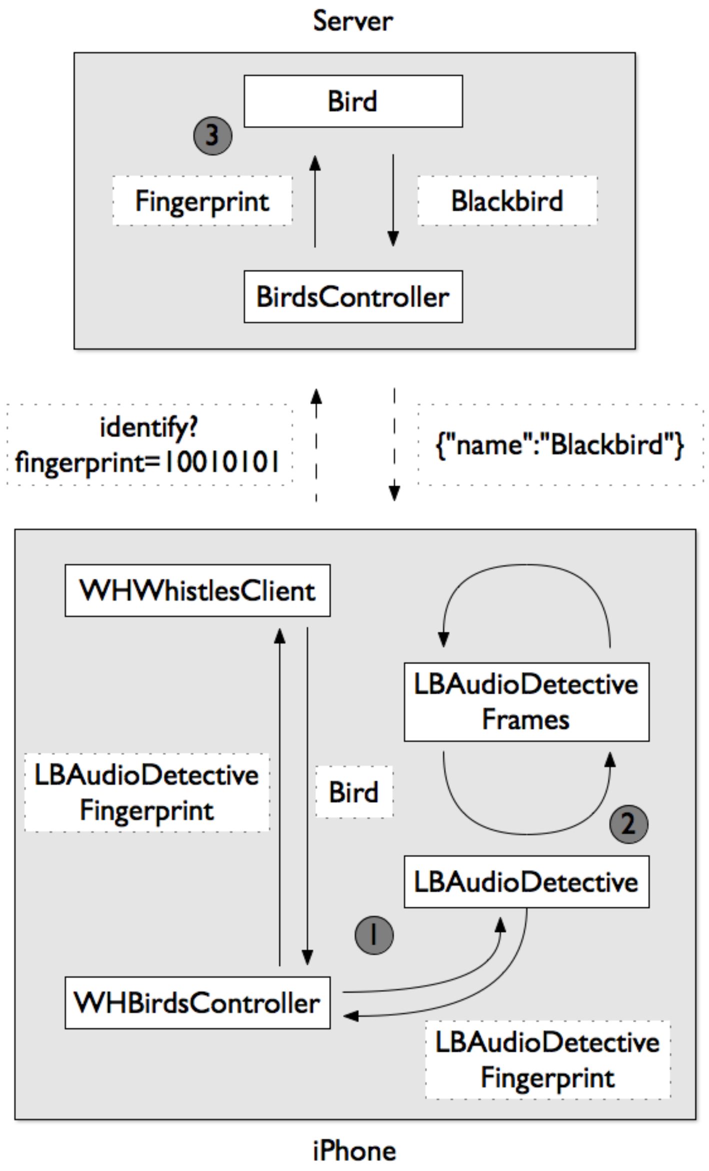


Figure 15: Illustration of the whole service with its classes

Because the recording is done in a very special manner due to the implementation of the audio analyzing algorithm, this explanation is not sequential. Firstly, the implementation of the audio fingerprinting, the generation of the fingerprint from the recorded audio, is described. This is step 2 in Figure 15. Then the recording, indicated by step 1, will be described in detail. Finally the matching of the fingerprints, step 3, is explained.

3.2.1 Audio Fingerprinting

During the development period of this project two algorithms have been deployed, each using different techniques. The Melody Analyzing Algorithm (MAA) is very simple and basic, it is described first. Following this the second, more elaborate Audio Fingerprinting Algorithm (AFA) is described, which is also the algorithm finally used as its result are better when compared to the MAA. For this reason, the MAA is not shown in Figure 15 but has been incorporated here to simplify the understanding of the progress and development of this project.
Note that the values discussed throughout this section are the default values. They can be modified to yield different results.

3.2.1.1 Melody Analyzing Algorithm

To fulfill the requirements, a core library has been programmed. It is programmed in C because of the language's efficiency. The MAA[1] consists of one core type[2]:

LBAudioDetective.m

LBAudioDetective's major task is the recording and audio fingerprinting itself. It is highly flexible in terms of settings and has a comprehensive range features. [3]

The MAA can not really be considered an fingerprinting algorithm. It does not pursue the idea of deploying a fingerprint and instead tries to analyze and compare the melody in the recording.
The best way to describe a melody or an audio signal as a graph is a spectrogram. LBAudioDetective takes advantage of this method and creates one. It does this in a very basic manner and splits the audio signal into multiple windows of 512 KB in size. To these windows a FFT is applied which makes it possible to plot a spectrogram[15]. The window size is a trade off between performance and accuracy. Because

[1]Note that this is not an official term. The first algorithm was given this name to simplify this text.

[2]Note that this is called type because it has been developed in a C environment. To C, classes are unknown. However, it has been developed in a object oriented manner. That is why it is called type.

[3]LBAudioDetective is present in the implementation of both algorithms. Although it has the same task, its implementation is different in both. Thus, two versions of LBAudioDetective have been developed which are totally different in their implementation.

the window size has an impact on the accuracy of the spectrogram, the windows need to be rather big which leads to an imprecise spectrogram.

Although implementing the FFT individually would be possible it was decided to take a different approach. Apple provides an incredibly fast framework specialized for digital signal processing. It is called Accelerate. Apple's documentation of the Accelerate framework states, "The Accelerate framework is a set of libraries containing high-performance vector-accelerated libraries"[16]. It is extremely fast because of the hardware acceleration it is based on. Using a framework such as this instead of developing it by oneself has two main advantages: Firstly, a specialized team of developers takes care of the backward compatibility, secondly the developer does not have to reinvent the wheel but can rely on the expertise of these professionals. This well proven framework saves a lot of development time.

To make a comparison between two spectrograms possible, the pitches are categorized by their value. 20 kHz, the highest perceivable frequency, is split into 5 ranges. For every window, the frequency with the highest magnitude within a certain range is stored. Therefore, the result of one window of 512 KB is an array of 5 values.

Listing 1: Pseudo code for algorithm 1

```
1  AudioData data = RecordAudio;
2  for (every 512 KBs of data) {
3          Dictionary pitch_and_magnitudes = apply_fft(element)
4
5          for (every pitch and its magnitude in pitch_and_magnitudes) {
6                  Category category = category_for_pitch(pitch);
7
8                  if (magnitude > categories.magnitude) {
9                          category.magnitude = magnitude;
10                         category.pitch = pitch;
11                 }
12         }
13 }
```

Listing 1 shows a possible way of implementing the MAA. Note that it retrieves the corresponding category for every pitch. So if the pitch is 3800 Hz, the variable category on line 6 would have a frequency range of 0 Hz - 4400 Hz. The purpose of this separation is to assess every bird equally, regardless of the predominant frequency range of its whistling.
The final result of the MAA is a two dimensional array which holds 5 frequencies for every FFT window. These are easy to compare.

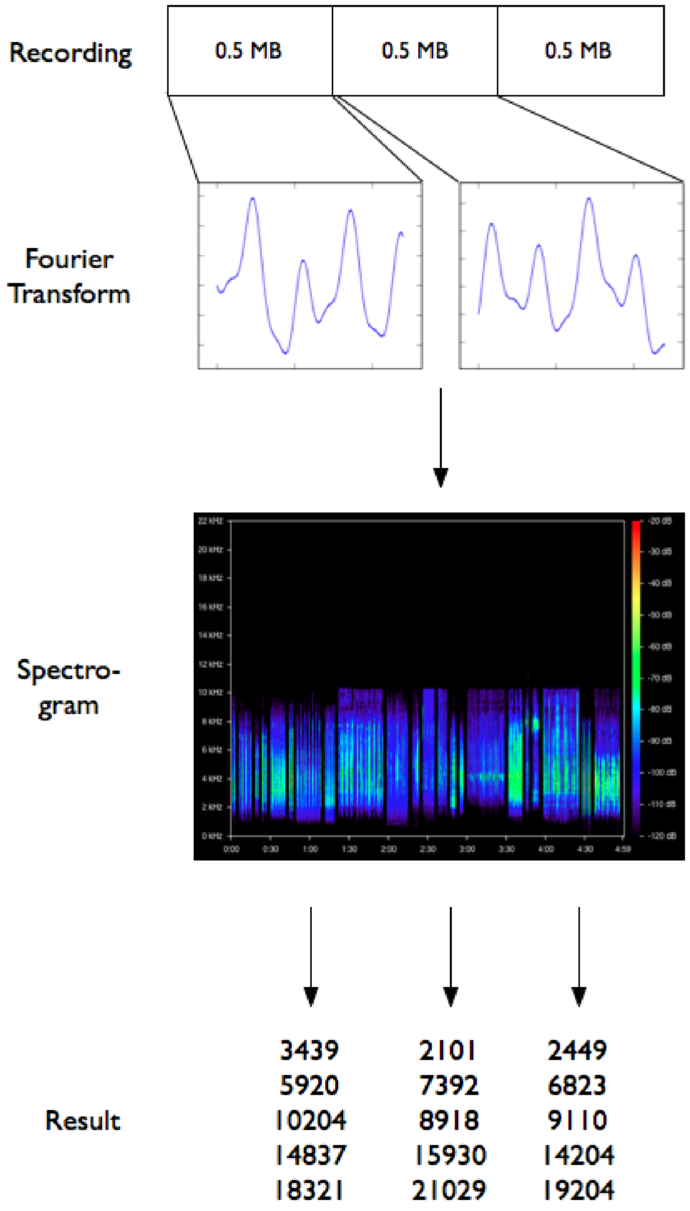


Figure 16: Illustration of the Melody Analyzing Algorithm's implementation

3.2.1.2 Audio Fingerprinting Algorithm

In addition to LBAudioDetective, two other core types have been developed to implement the AFA:

LBAudioDetectiveFingerprint.m
: LBAudioDetectiveFingerprint is the product of the whole audio fingerprinting done by LBAudioDetective. The main attribute of it is its comparability. A fingerprint consists of many subfingerprints that are basically an array of booleans.

LBAudioDetectiveFrame.m
: LBAudioDetectiveFrame is an internal type of LBAudioDetective. It is used to gather the data from the recording and apply various algorithms to it. It is not used for any other purpose. A frame produces one subfingerprint which in itself is a summary of multiple FFTs.

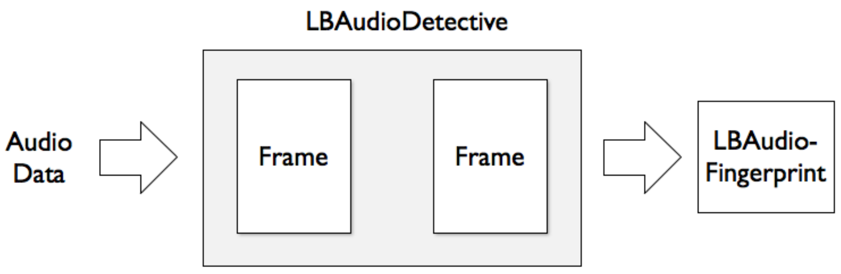


Figure 17: A simplified fingerprint generation process

The AFA is much more complex than the MAA and relies on research carried out by Google Inc[17]. The major difference between the MAA and the AFA is that it creates a more sophisticated spectrogram and abstracts this using various methods.

Before creating a spectrogram, the input audio data is converted into a different format. More precisely, LBAudioDetective downsamples[4] the whole audio signal. The outcome of decreasing the sample rate from the default value of 44100 Hz to 5512 Hz is a reduction of insignificant information. Since the highest frequency describable with a sample rate of 5512 Hz is 2756 Hz a filtering of the sounds automatically takes place. This is useful as the major audible frequencies of human beings lies within this range[17]. Moreover, because the sample rate defines the general quality of an audio signal, decreasing it erases a lot of background noise.

[4]Downsampling refers to the degradation of the sample rate

Like the MAA, the AFA creates a spectrogram. However, the alignment of the windows for the FFT is different. The windows, which are by default 2 MB, overlap. More precisely, for every 64th sample frame[5] a new window is defined. This makes the final fingerprint change slower in time. This makes it more robust for the time independence increases[18]. The FFT is applied to these windows and the result is then split up into 32 evenly spread categories which sum up every pitch in the corresponding frequency range. Combining pitches in this way instead of extracting the most significant ones means that more information can be taken into account.
Thus the result of the FFT of one window is an array holding 32 values. 128 of these arrays are then combined into one instance of LBAudioDetectiveFrame which represents a part of the final fingerprint, a subfingerprint.

LBAudioDetectiveFrame stores its arrays in a matrix. This allows it to decompose itself using the Haar-Wavelet Transform. The results of this transform are then sorted by their absolute values. The purpose of this sorting is to find the most significant data of the signal. Nevertheless, this would still be too much data. To further abstract the data set, it only stores the signs of the values. Therefore, LBAudioDetectiveFrame extracts the top 200 values' signs. This abstraction improves the performance and the robustness. The signs are then encoded using two booleans. 10 stands for a positive value, 01 stands for a negative one, 00 represents zero. Because one sign is now replaced by two booleans, the final subfingerprint is an array of 400 booleans. Consequently, one LBAudioDetectiveFrame becomes one subfingerprint which in turn is merged into one LBAudioDetectiveFingerprint which stores all subfingerprints for one signal.

3.2.2 Recording and Reading

The MAA is too simple and inaccurate so it was not incorporated in the product. This section focuses only on the main principles of the AFA. However, the AFA and the MAA share similar principles. LBAudioDetective is not only capable of analyzing available audio files stored on the device (reading) but also analyzing the bird simultaneously to recording it[6].

For the implementation of the audio recording a framework called CoreAudio has been used. It is one of Apple's standard frameworks. It is written in C and C++ thus very low-level and therefore performant. It provides a lot of control over the recorded audio data.

[5]Do not confuse sample frame with the window or LBAudioDetectiveFrame. While the sample frame refers to the samples of all channels at a given time, the window is a range to which the FFT is applied. LBAudioDetectiveFrame is a different type that represents a raw subfingerprint.

[6]Due to the compatibility issues with other frameworks, the final application does record and analyze subsequently. This makes it the user experience more reliable.

Core Audio's power lays in the audio units. Audio units are modular units that serve different purposes in terms of audio processing. Audio units may be connected and linked up with each other to build an audio processing graph. The units then process their input in their specific way and send it over to the next member of the row. This way an order is created.
There are many different types of audio units, the two most important types are the effect units, which modify the input by the use of an effect, and the I/O units (Input/Output), which either receive input from a microphone or send their output to speakers. I/O units are often referred to as AUHAL units. AUHALs are audio units from the HAL (hardware abstraction layer). HAL serves as an interface which allows the programmer to handle every input or output device, regardless of its technology, in the same way.
Audio units provide various buses as well. Buses or pipes are the connection points. These can either be part of the input or the output scope.

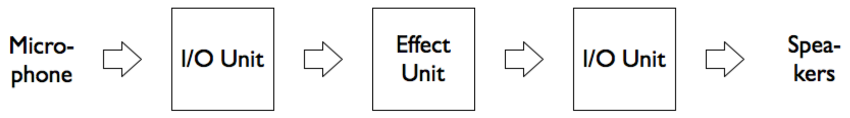


Figure 18: An audio processing graph that records audio, applies an effect and then plays it back simultaneously

Core Audio was originally developed for the Mac OS and later on ported for the iOS platform. Because iOS devices have only a fraction of the desktop computers computing power available, the frameworks iOS port has numerous restrictions. One of them is the I/O unit. There is only one I/O unit available on the iOS platform, the remote I/O unit (RIO unit). One of the buses in the input scope is connected to the device's input, the output of the microphone and one of the other buses in the output scope is connected to the device's output, the speakers input.
Considering the fact that this project only needs to analyze audio signals, a fairly simple audio processing graph is sufficient.

Figure 19 shows the final graph used in LBAudioDetective to get data. Bus 0 of the input scope and bus 0 of the output scope are gray as they are not used for this task. As soon as the graph is initialized, LBAudioDetective can listen to bus 1 of the input scope. This way it is notified on a regular basis when data is available. This allows LBAudioDetective to define the FFT windows whenever required.

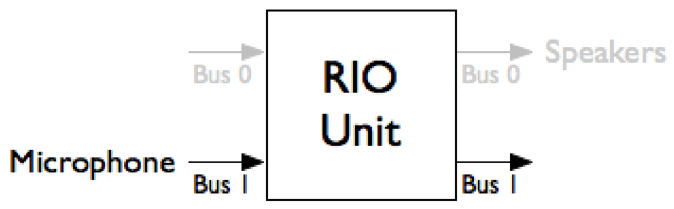


Figure 19: The final audio processing graph used in LBAudioDetective

For reading audio files stored on the device, an even simpler setup is sufficient. Core Audio provides a type called AudioFileID which makes it possible to read the whole audio file in any way imaginable.

Listing 2: Pseudo code that reads an audio file

```
1 int size = AudioFileGetSize(audioFile);
2 for (int i = 0; i < size; i += 64) {
3         int readNumberFrames = ExtAudioFileRead(audioFile, 2048, bufferList);
4 
5         // Use the window currently saved in bufferList
6 
7         ExtAudioFileSeek(audioFile, i);
8 }
```

Listing 2 shows how AudioFileID can be used to read overlapping windows. On line 3 it reads a portion of 2 MB in size and saves it into an instance of AudioBufferList. After applying the FFT to it and storing it into an instance of LBAudioDetectiveFrame the AudioFileID can jump 64 sample frames. It does this iteration until it reaches the end of the file.

3.2.3 Matching

Matching refers to the process of comparing two fingerprints and their respective subfingerprints. Since they are never exactly the same, they need to be compared in detail. Throughout this project, the matching algorithms have been developed for both the iPhone and the server. Although the matching will eventually only be performed by the server, it simplifies the development and testing process when only using an iPhone.

3.2.3.1 Melody Analyzing Algorithm

For this algorithm, matching is very simple. The subfingerprints which consist of five numbers are subtracted by each other to find the difference. If the difference does not exceed a certain threshold the subfingerprints match.

Listing 3: Matching pseudo code for the Melody Analyzing Algorithm

```
for (index = 0; i < length; i += 2) {
    d1 = abs(subfingerprint1[1].freq)-abs(subfingerprint2[1].freq);
    d2 = abs(subfingerprint1[2].freq)-abs(subfingerprint2[2].freq);
    d3 = abs(subfingerprint1[3].freq)-abs(subfingerprint2[3].freq);
    d4 = abs(subfingerprint1[4].freq)-abs(subfingerprint2[4].freq);
    d5 = abs(subfingerprint1[5].freq)-abs(subfingerprint2[5].freq);

    d = d1+d2+d3+d4+d5;
    if (d < 400) {
        match = YES;
    }
}
```

3.2.3.2 Audio Fingerprinting Algorithm

LBAudioDetectiveFingerprint, the fingerprint type used for the AFA, contains an array of 400 booleans. These are very easy to compare and make it possible to calculate the similarity. The AFA always compares 2 booleans at a time. This is because two booleans are needed to encode one wavelet.

Listing 4: Matching pseudo code for the Audio Fingerprinting Algorithm

```
1  brutto = 0;
2  hits = 0;
3  for (index = 0; i < 400; i += 2) {
4      b1_1 = subfingerprint1[index];
5      b1_2 = subfingerprint1[index+1];
6      b2_1 = subfingerprint2[index];
7      b2_2 = subfingerprint2[index+1];
8
9      if (b1_1 OR b1_2) {
10         brutto++;
11
12         if (b1_1 == b2_1 AND b1_2 == b2_2) {
13             hits++;
14         }
15     }
16 }
17
18 match = hits/brutto;
```

On line 3 in Listing 4, the algorithm iterates through every pair of booleans and then checks if one of the booleans of the first subfingerprint is equal to YES. If so, the brutto is increased by one. The variable hits is in turn increased if the two corresponding booleans match.

3.2.4 iPhone Application

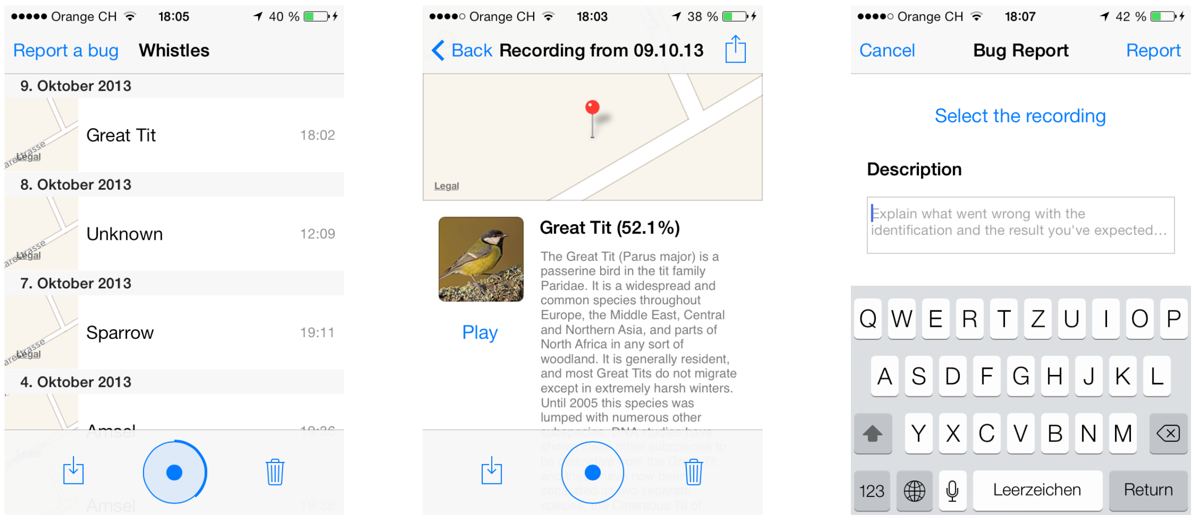
The aim of this project was to develop an application which would be easy for people to use. Therefore it should not only look nice but also have the most important features available. It should also be informative and logical to use. Whistles, the product, has four major classes:

WHBirdsViewController.m
: WHBirdsViewController is the root view controller. It displays an archive of the recordings, sets the recording input devices and triggers the recording and analyzing.

WHBirdViewController.m
: WHBirdViewController displays further information on a bird that was previously recorded.

WHBugViewController.m
: WHBugViewController makes it possible to report bugs.

WHWhistlesClient.m
: WHWhistlesClient connects the iPhone with the server. This makes it possible to send and receive data such as fingerprints and birds.



The Great Tit (Parus major) is a passerine bird in the tit family Paridae. It is a widespread and common species throughout Europe, the Middle East, Central and Northern Asia, and parts of North Africa in any sort of woodland. It is generally resident, and most Great Tits do not migrate except in extremely harsh winters. Until 2005 this species was lumped with numerous other

Explain what went wrong with the identification and the result you've expected...

(a) WHBirdsViewController (b) WHBirdViewController (c) WHBugViewController

Figure 20: Screenshots of Whistles

The root view, controlled by WHBirdsViewController, is a table view that has one cell for every recording. It shows where and when it was recorded and what kind of a bird it is if it was recognized. Maps are shown instead of a pictures of the bird because the map helps the user find the right bird in the archive, especially when considering the fact that he might have recorded the same sort of bird many times.

The tab bar bears the main features of the application. It has three buttons that can be used to control the recording. The left one is used to select the input microphone. The middle button is used for recording. In order to easily pause the recording, the user is expected to keep pressing the button as long as the recording should last. Releasing it makes the recording pause. The trash button on the right hand side reverts the current progress of the recording. Figure 20a shows the archive containing 5 recordings of which 4 have been identified. Tapping on one of them reveals the WHBirdViewController's view. The content of this view depends on the status of the recording.
Figure 20b shows this information view for a identified bird. It allows the user to play the recording again. The map shows again where the bird was recorded. Note that this map is not a major part of this project[7]. Note that the tab bar is still visible in this view. This makes it easy to start recording a new bird at any given time. Because it is very important to have as much information as possible, Whistles allows the user to report issues and send any problem or bug which is then saved on the server.

Unlike the view controllers, there are classes that do not directly display something on the screen. An example for this is WHWhistlesClient, which is responsible for the connection between the server and the iPhone, as shown in Figure 21.

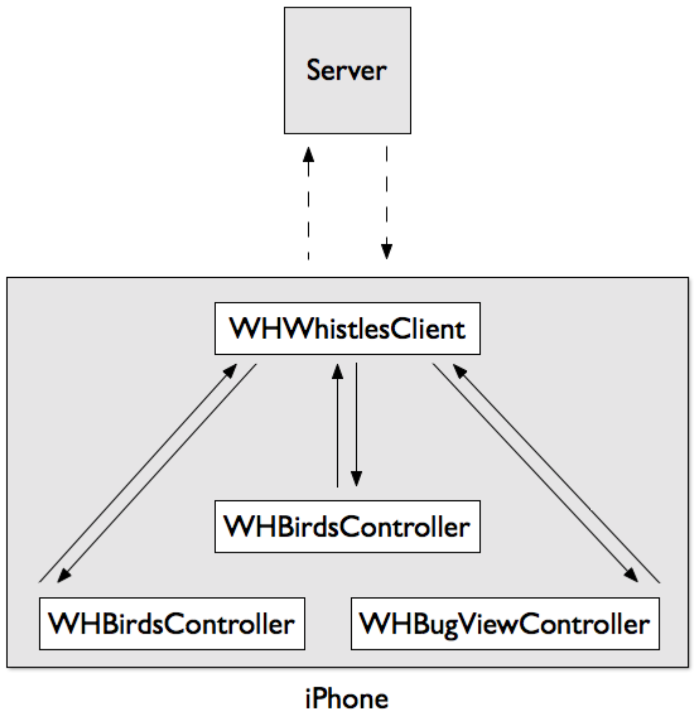


Figure 21: Whistles' view controller and their connection to the server

[7]Just like CoreAudio, Apple provides a framework called MapKit that lets the application keep track of locations and makes it easy to display them on the screen. It gathers the location by mixing cellular, wifi and GPS data.

3.2.5 Server

The server's main task is the matching. To make this possible, it needs to be able to receive a request, access the database and find the fingerprint that matches the best to the one received. Moreover it should be able to save bug reports. The servers core consists of four ruby classes:

ApplicationController
: The application controller is generally used as a root class or controller for the whole server. For this project it is only used to receive and handle the bug reports.

BirdsController
: This class handles all bird related requests.

Bird
: The bird class is not only a class but a model which stands for an entry in the database.

Fingerprint
: This is the server counterpart of LBAudioDetectiveFingerprint.

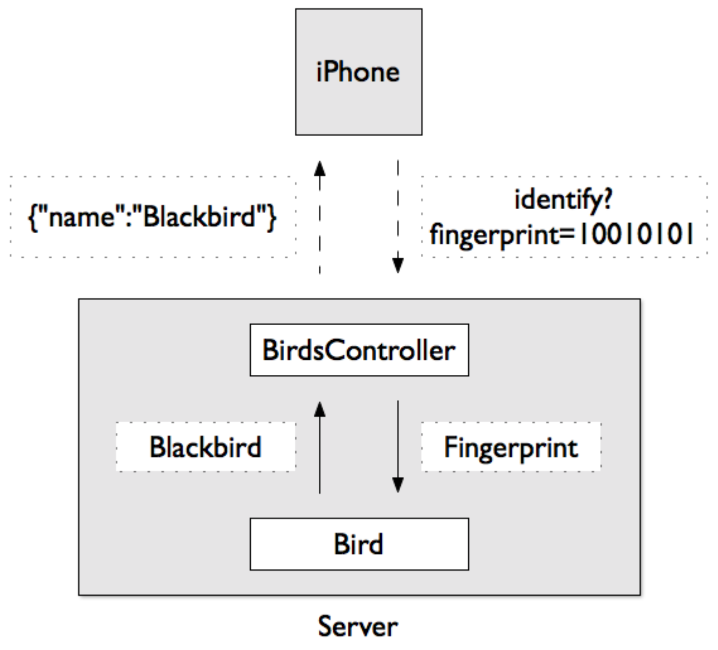


Figure 22: The matching process from the server point of view

Figure 22 depicts an identification request. The iPhone has recorded a bird and sends it to the server. The bird controller handles this request, creates a fingerprint instance with the received data and hands it over to the bird class. The bird class then performs the actual matching. It iterates through all database entries and returns the bird instance, in this case the blackbird, which holds the fingerprint that matches the best. The bird controller in turn responds to the request with the bird encoded in JSON.

Once the server's core was finished, the database was populated with ten birds and their additional information like descriptions and pictures. The blackbird, the blue tit, the chaffinch, the sparrow, the great tit, the crow, the wren, the kestrel, the chiffchaff and the pigeon have been stored in the database and were used for the tests. These birds have been chosen by their popularity in Switzerland and the quality of the recordings available from the national ornithological station in Sempach, Switzerland[8]. The number of birds is specified by the aims of this project.

id	2
name	Blackbird
`created_at`	`2013-09-08 18:39:11`
`updated_at`	`2013-09-08 18:39:11`
fingerprint	10010101001010...
description	The Common Blackbird...

Table 3: The database entry of the Blackbird

Table 3 shows an example for the bird model. The attributes name, fingerprint and description are required by the service whereby the other attributes are internal and not directly needed.

[8]I have used the recordings available on their website as it was well recorded and was therefore of good quality.

4 Results

To assess the two algorithms a rather simple test suite has been developed. The suite consists of four tests. These tests work in the same way but different resources have been used to generate the fingerprints.
In the beginning, ten fingerprints are generated using recordings from the ten birds mentioned earlier. Each recording is nine seconds long and they represent the database of the server. Next, ten other fingerprints are created using recordings from the ten birds. These recordings however were only four seconds long (granularity). This granularity was chosen in consideration of the duration of the bird's songs. The main theme of the 10 birds assessed are shorter than 4 seconds.
The test suite focuses only on the algorithms as this is the primary objective of the project.
The task of the algorithm was to find matching fingerprint pairs. It compared all 20 fingerprints with each other to associate two matching fingerprints by their similarity.

To make the following assessment and its explanation as little confusing as possible the audio file used to generate the database fingerprint will be called "the original audio file". The audio file which is to be identified will be called the "unknown audio file".

The following tests have been used to assess the software:

Test 1
: The unknown audio file is a cropped copy of the original audio file. It is therefore a part of the original one.

Test 2
: Two different recordings of the same bird species were used to compute the matching fingerprint pair. This test assesses the reliability of the algorithm.

Test 3
: The same audio files as the ones used in Test 1 have been used. However, to the unknown audio file a random noise was added. This test assesses the robustness of the algorithm. It was repeated two times with different noise amplitudes[9] to find a threshold.

Test 4
: The same audio file as the one used to generate the database fingerprint is played back. The iPhone records and subsequently analyzes it to find the matching bird. This test imitates an actual use case of the application.

[9]This is measured by the amount of data the noise makes for every sample.

These four tests make it possible to review the different aspects that make up the accuracy of the product. Throughout the development this test suite has been used and this test driven development was proved very useful as the smallest change in the source code can lead to crashes or bugs[10].
The decimal precision has been set to one significant figure. This is sufficient to evaluate the outcome.

4.1 Melody Analyzing Algorithm

4.1.1 Test 1

Note that the MAA is not able to calculate equality as a percentage. Thus the values indicate the number of common frequencies. Without significant differences in equality, three birds were successfully identified. Due to the fact that it did not pass Test 1 it would have been redundant to assess it with the remaining tests.

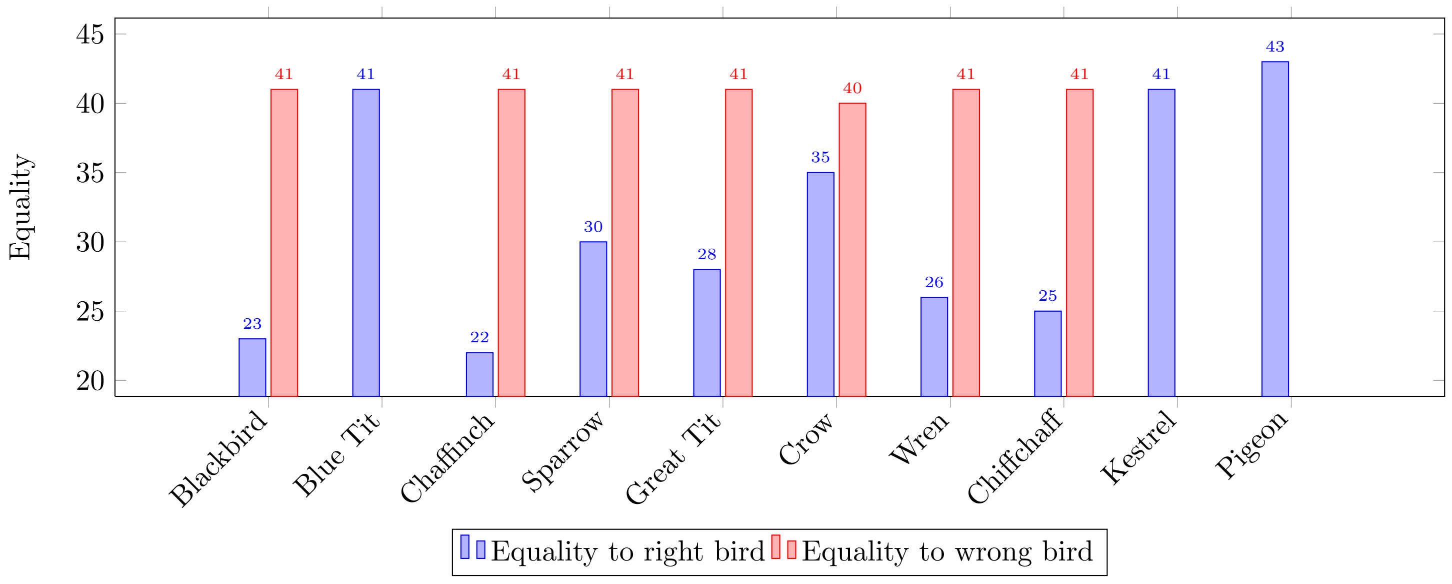


Figure 23: The test results of Test 1

[10]Because the programming environment used is fairly cryptic, the bugs would have been almost impossible to find without double checking every time with these tests.

4.2 Audio Fingerprinting Algorithm

4.2.1 Test 1

Clearly visible, the algorithm was able to recognize every bird with an accuracy over 90%.

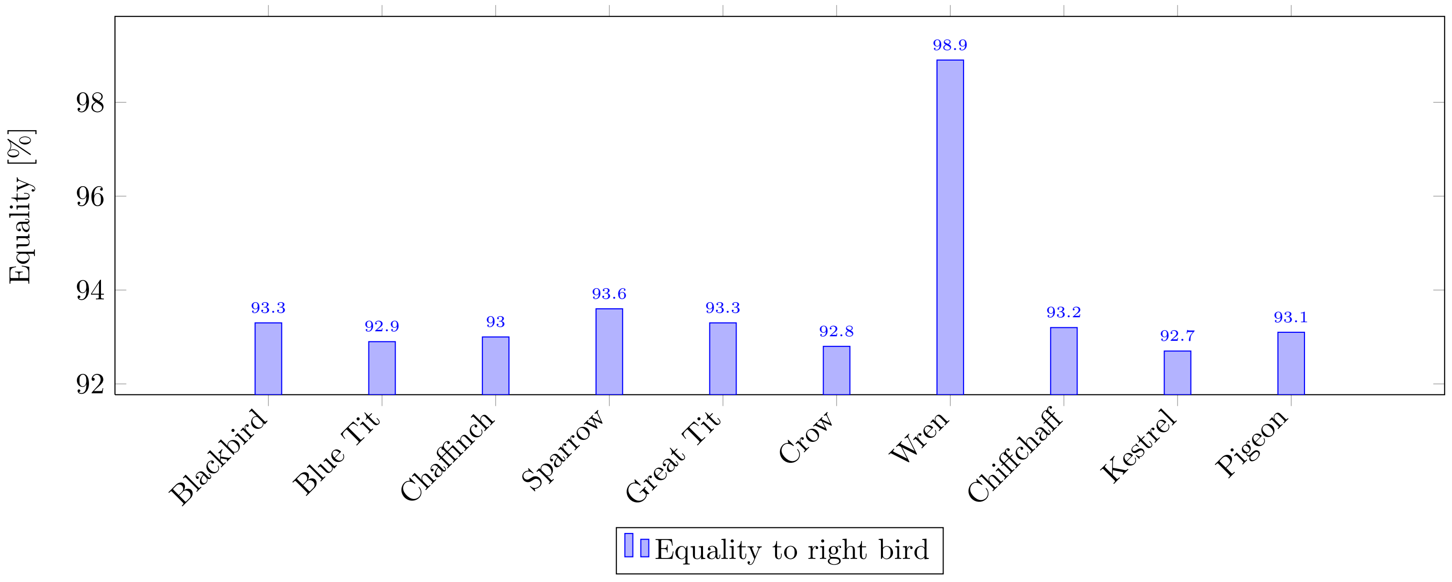


Figure 24: The test results of Test 1

4.2.2 Test 2

The birds used for this test have been downloaded from Youtube. This test assesses the reliability. The bird pairs are most likely not the same individual and are definitely not the same recording. Thus the recorded whistling does not sound the same.

Figure 25 indicates that only the recording of the kestrel could be satisfactorily identified. Although the average error is quite small the other birds could not be identified.

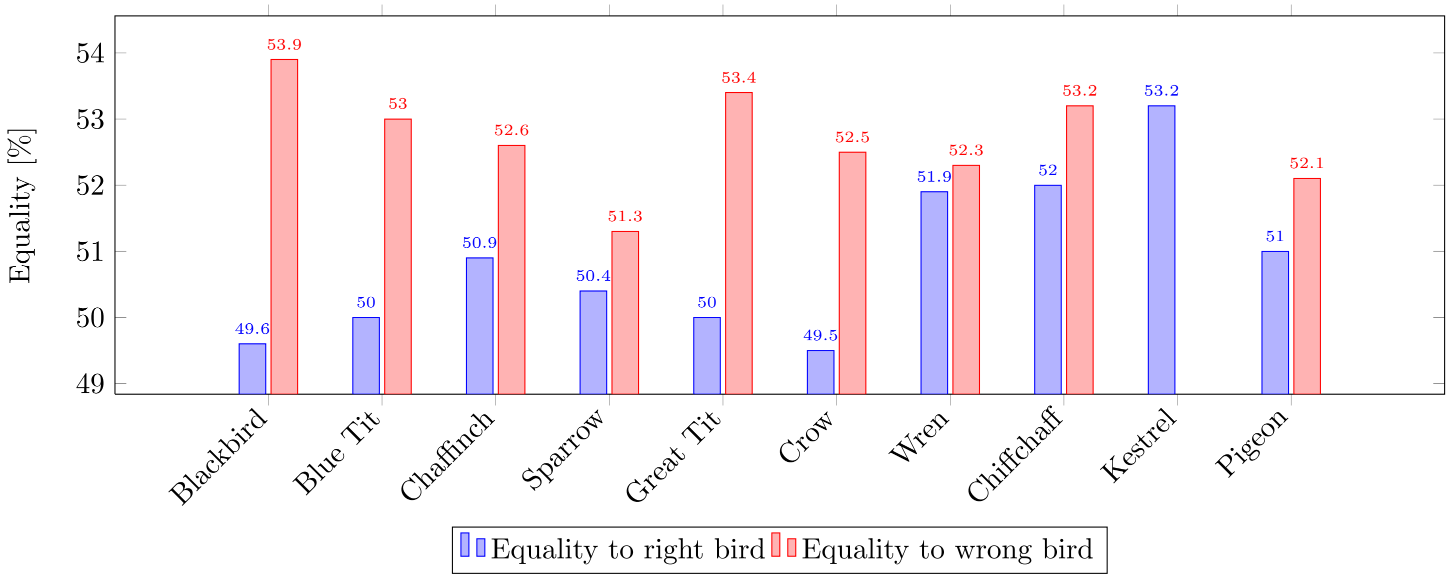


Figure 25: The test results of Test 2

4.2.3 Test 3

These test results need to be carefully evaluated because every recording reacts differently to the additional noise. For some recordings, like the one from the crow, it is hardly audible.

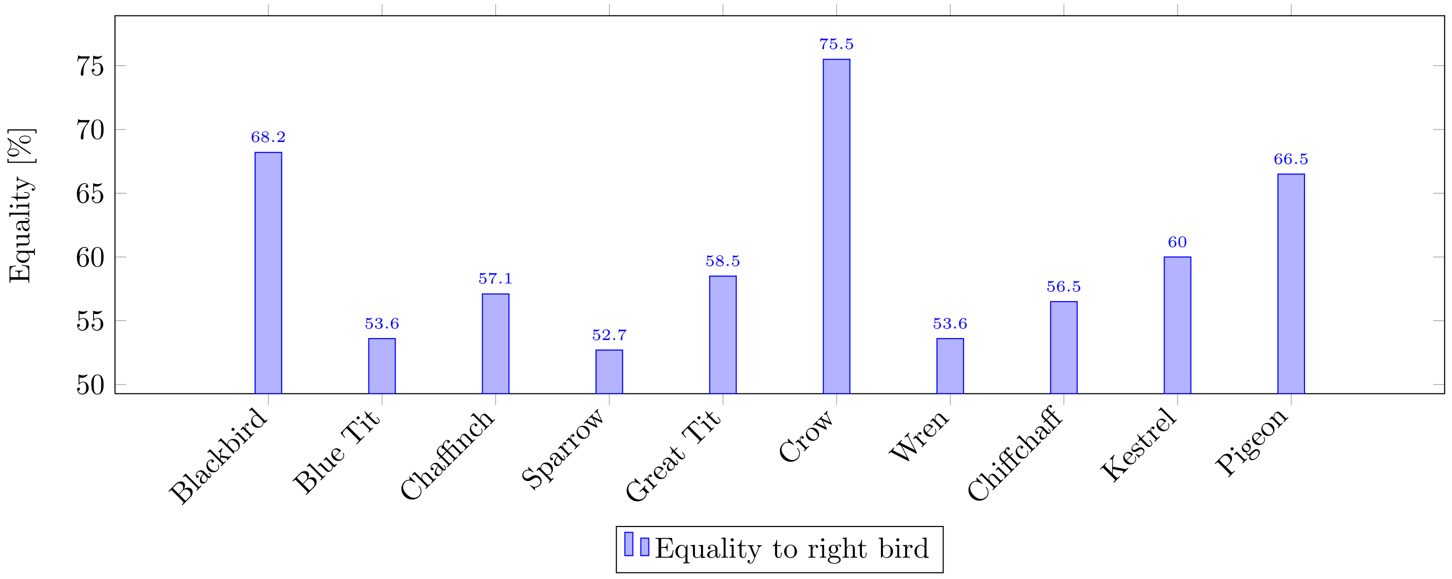


Figure 26: The test results of Test 3.1 with a random noise added (1.58%)

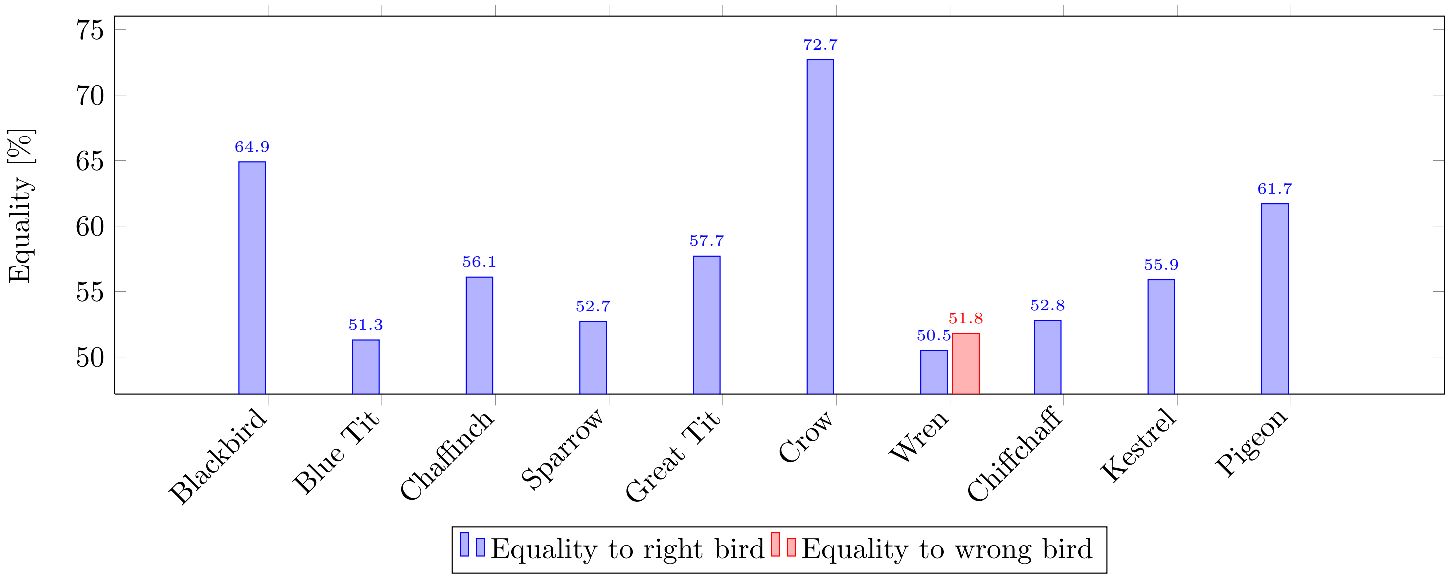


Figure 27: The test results of Test 3.2 with a random noise added (3.16%)

4.2.4 Test 4

Similar to the outcome of Test 2, the computed equality does not exceed the range from 49% to 54%. Two birds, the chaffinch and the pigeon, have successfully been recognized by the algorithm.

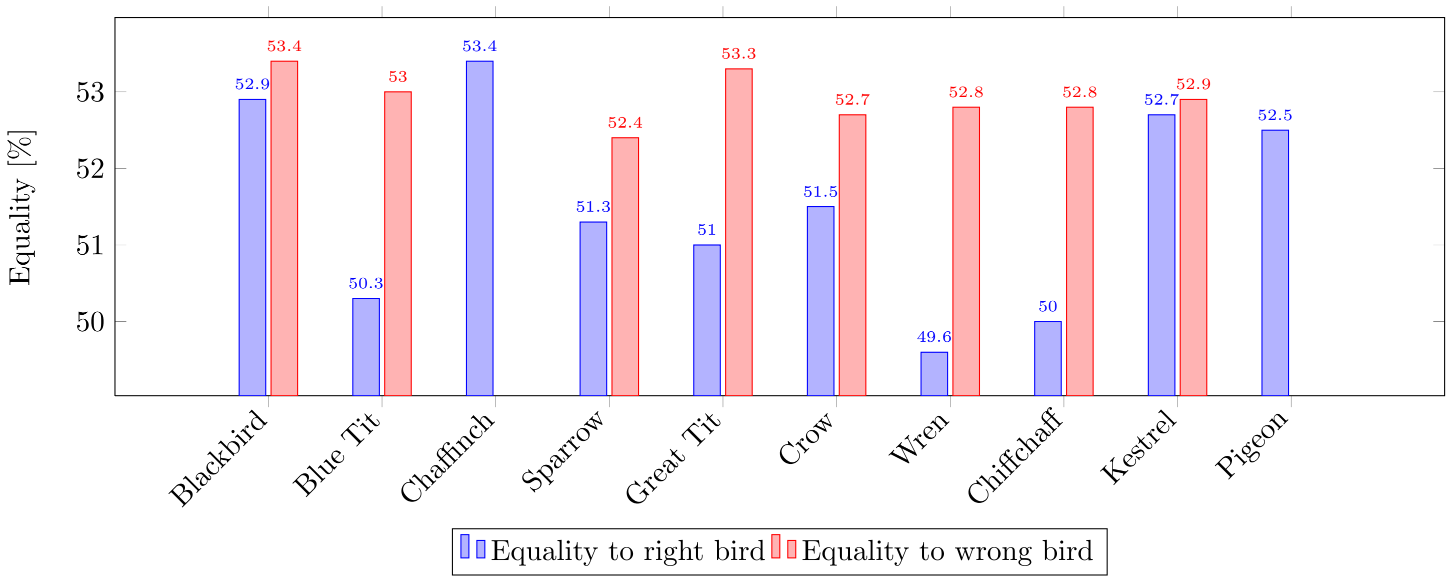


Figure 28: The test results of Test 4

5 Discussion

5.1 Application and Server

Overall, the service, which includes the iPhone application, the server and the database, has turned out very well.
The iPhone application is stable, easy to use and provides all the necessary features. It is capable of pausing and resuming the recording. It improves the user experience with lightweight animations and details such as fading playback.
The server is able to access the database properly and features everything required to fulfill its task. Everything has been developed in an efficient and clean way which makes further development easier.

5.2 Audio Fingerprinting

In this section the outcome of the tests will be explained. They are also compared to audio recognition services like Shazam or SoundHound as they use similar techniques. There are also references to the different properties that make up a fingerprinting algorithm as described in section 2.3.

5.2.1 Melody Analyzing Algorithm

5.2.1.1 Test 1

Unfortunately, although three birds passed the test, the algorithm as a whole has failed this test as the discrepancy between the passed and the failed recordings is too small.
There are a number of issues that lead to a failure in Test 1. One of them is that it neither filters the audio nor the frequencies. This lack of filtering leads to a less robust algorithm as it takes every tone into account. Moreover, due to the design of the MAA it is not capable of computing a relative equality. This is due to the fact that it only compares the two audio files by counting the number of similar frequencies. This makes it harder to find the best matching bird song considering that every fingerprint needs to have exactly the same length. However, there is another contradiction in its design. The goal of the algorithm is to analyze the melody of the input song. A melody's main property is its distinct order of tones. However, to estimate this melody the windows used to create the spectrogram is too big to retrieve the melody adequately. Using a smaller one would be a possibility but would also have a negative impact on the performance.

Because the MAA did not pass Test 1 an assessment of the other three tests was clearly redundant. Further results can not justify this algorithm.

5.2.2 Audio Fingerprinting Algorithm

5.2.2.1 Test 1

As already mentioned, this test does not check the quality and accuracy of the algorithm but whether it actually works. But why is the computed equality somewhere between 92% and 99%? The problem here is the time shifting. Second 2 of the original audio file might be second 5 of the unknown audio file. The FFT windows which are calculated during the two analysis' are never concurrent which leads to subfingerprints that do not bear exactly the same information. The AFA tries to avoid this issue by overlapping the FFT windows to minimize the time shift.
Thus it can be concluded that the common error, which is up to 7.3%, is due to the fact that the algorithm struggles to handle the time shifting.

5.2.2.2 Test 2

This test shows that the algorithm is not reliable. All the fingerprints are about 50% equal to each other. This can be explained as follows: Since the fingerprint produced by the AFA is a long array of booleans, the probability that two random fingerprints are considered equal is a chance of 50%.
Due to the fact that the kestrels two fingerprints' equality is not significantly higher than 50%, it is only luck that it was identified successfully.
So why did LBAudioDetective fail this test? Firstly, there is a severe incapability of handling the time shift here as well, especially when the recordings are not equal, which is the common case. This inability is most probably the lone cause for the failure. The reliability is also influenced by this. Additionally, the abstraction of the fingerprints may not have been sufficient, a greater reduction of detail could improve the result.

5.2.2.3 Test 3

Although Test 3.1's additional noise was only half as loud as in Test 3.2, LBAudioDetective's accuracy was not twice as good. The robustness of LBAudioDetective is therefore not linear. It is not able to handle it better the less noise is added. The maximal sound degradation LBAudioDetective can cope with is around 3%. The average accuracy is 56.63% in Test 3.2 which is slightly higher than 55%. Nevertheless, this result is remarkable, considering that the random noise of Test 3.2 is clearly audible. This shows that the abstraction processes of the algorithm is able to filter out the significant information and successfully neglect background noise.

5.2.2.4 Test 4

This outcome appears similar to that of Test 2. However, a mixture of the time shift issue and background noise most probably caused this outcome. It shows that LBAudioDetective is not reliable in real life situations.

The evaluation of these test results can explain the major issues LBAudioDetective currently has. However there is a further problem in the actual task of recognizing birds.
When comparing LBAudioDetective with Shazam, which is able to recognize songs, the recognition of birds by their song is very different. Most significantly, birds are animals and are spread all over the world. They have accents which make certain species sound different for the various regions they inhabit. Not even an individual sings the same song exactly the same every time. Since they are animals they can pause their whistling or react to a specific situation whenever they want. Subsequently, compared to a typical music track, a bird song never sounds exactly the same which means greater reliability is required. To minimize this issue, recordings of birds from many different regions would be required to generate a comprehensive fingerprint. The fingerprint size would increase tremendously which in turn would have a huge impact on the performance.
Furthermore, a bird song is very short when compared to a music track. Although the duration may vary, the song is often shorter than ten seconds. A good example for this is the sparrow's song which consists of one short tweet per second. Between these tweets, there is no data which could be used to identify the bird [11]. Due to the fact that the songs are comparatively short, the fingerprint does not contain a lot of data. The smaller the data is the higher is the probability that they are equal. Hence, it is a lot easier to fingerprint a track which is three minutes long as it contains more data and more characteristics.
In consideration of this fact, the scalability is severely limited for this approach.
Finally, birds usually sing in a group. Thus, it is hard to find one bird singing by itself close enough to record it with the iPhone. Recording a mixture of various birds makes it almost impossible for the application to identify the singing birds. Thus, developing such a service which is useful in everyday life would be very difficult if not impossible given the current technology.

[11]No data is in reality also a characteristic. However, it is not a significant one as many birds have this gap between their main song.

6 Conclusion

6.1 Summary

The aim of this project was to develop a whole service, including an iPhone application and a server, called Whistles that recognizes birds by their song. In introduction, the theory of audio fingerprinting including algorithms like the FFT and the Haar-Wavelet Transform used to implement such fingerprinting technologies were discussed. Next, the functional and qualitative requirements needed to produce a working service were evaluate. Then, the melody analyzing algorithm was developed. It was not based on any well founded research and proved to be rather unsuccessful. Learning from this and with further research a new algorithm, the AFA was developed. It is a far more sophisticated algorithm that addresses more problems than the first approach. This has lead to a robuster and more reliable algorithm. To utilize development an iPhone application with a well designed and intuitive user interface was made. In addition, a server, which is able to communicate with the application and which has access to a database that stores ten birds and their metadata, was created.

6.2 Outlook

The final product could be improved in various aspects, especially the reliability. There are a number of approaches which could be implemented to make the algorithm more reliable. One of them would be a neural network which could learn the song of a bird with its artificial intelligence. It could be based on the same theory used to implement the algorithm I developed but it would match the birds in a totally different way. It could neglect the accents of an individual and prove itself to be a good solution.
Further improvements would include the development of a more efficient service but as we have seen, any reduction in fingerprint size could likely result in a reduction of the reliability. Therefore, further developments would need to concentrate on the robustness and reliability, improving these first.

6.3 Afterword

The primary goal of developing an iPhone application which is able to record and recognize bird's whistling in a garden requires a very robust and reliable fingerprinting algorithm. Its development has proven difficult and was not possible for me to achieve in the given time. Nevertheless, considering the requirements of the app and the high personal standards for this project, there are certain areas of real success. Besides a well designed iPhone application and a fully functioning server with its own database, a powerful fingerprinting algorithm was developed that can solve different problems such as finding duplicates in a digital music library.
My personal motivation for developing a project like this was my own curiosity. I was fascinated by the

idea that a machine could be able to learn what animals and music sound like. It seems somewhat like magic when you have no idea of what is happening in the background of such a service. Therefore, I wanted to try this myself. I could have replicated already existing services that recognize music tracks but it was important for me to develop a new idea. Because of this and most importantly I have learned a great deal. Not only about recognizing audio signals with a machine but also about general computer science. I have studied many algorithms which are widely used in science and which are capable of solving difficult problems. I have also become more proficient in a wider variety of programming languages. This project has therefore proved to be a workbench on which I have been able to extend my knowledge.

7 Appendix

A Glossary

AFA Audio Fingerprinting Algorithm, the second fingerprinting algorithm developed which is based on research of different institutes. 18, 22, 23, 26, 39, 41

API Application Programming Interface, a protocol provided by a company to program for an OS [19]. 15

array In programming, an array refers to a list like construct. It can therefore store multiple items in a specific order. 20, 22, 23, 26, 39

boolean In programming, a boolean or flag refers to a value of one bit of size. It can therefore be 1 (YES) or 0 (NO)[20]. 22, 23, 26, 39

FFT Fast Fourier Transform. 9–11, 19, 20, 23–25, 39, 41

framework A framework is a group of tightly connected classes to solve one specific problem[21]. 14, 20, 23, 24, 28

hardware acceleration The CPU (central processing unit), the general computing unit of a computer, loses a lot of performance because it has do be able to fulfill many different tasks. The GPU (graphics processing unit) is a lot more powerful than the CPU but its capabilities are also very limited. Hardware acceleration refers to the usage of components meant for specific operations[22]. 20

MAA Melody Analyzing Algorithm, an audio analyzing paradigm developed by myself. 18, 20, 22, 23, 33, 38, 39

B References

[1] Mrowinski, Dieter, Scholz, Günther, 2005, Audiometrie: Eine Anleitung für die praktische Hörprüfung, third edition, Stuttgard, Thieme, 10-11

[2] Adamson, Chris, Avila, Kevin, 2012, Learning Core Audio, second edition, London, Pearson, 25-31

[3] University of California, Berkeley, The Nyquist-Shannon Sampling Theorem, `ptolemy.eecs.berkeley.edu/eecs20/week13/nyquistShannon.html`, last visited: 2013-8-4

[4] Müller, Ralph, 2010, Die geheime Sprache der Vögel, first edition, Aarau, AT Verlag, 184-192

[5] Gravell, Dan, What is audio fingerprinting?, 2012, `www.blisshq.com/music-library-management-blog/2012/08/21/what-is-audio-fingerprinting/`, last visited: 2013-9-5

[6] Haitsma, Japp, Kalker, Ton, A Highly Robust Audio Fingerprinting System, `www.nhchau.com/files/AudioFingerprint-02-FP04-2.pdf`, last visited: 2013-9-12

[7] Burke Hubbard, Barbara, 1997, The World According to Wavelets: The Story of a Mathematical Technique in the Making, second edition, Florida, CRC Press, 117-153

[8] Bernsee, Stephan M., The DFT a Pied: Mastering The Fourier Transform in One Day, `www.dspdimension.com/admin/dft-a-pied/`, last visited: 2013-10-2

[9] Mulcahy, Colm, Image Compression using the Haar Wavelet Transform, `www5.spelman.edu/~colm/csam.pdf`, last visited: 2013-10-2

[10] Ferguson, Andrew, A History of Computer Programming Languages, `cs.brown.edu/~adf/programming_languages.html`, last visited: 2013-7-30

[11] Kernighan, Brian, Ritchie, Dennis, 1988, The C Programming Language: Ansi C, second edition, London, Pearson, 6-8

[12] Negm-Awad, Amin, 2008, Objective-C und Cocoa, third edition, Heidelberg, Smart Books Publishing AG, 32-33

[13] Matusmoto, Yukihiro, Flanagan, David, 2008, The Ruby Programming Language, first edition, Newton, O'Reilly, 2-4

[14] Ruby Language, Ruby, A Programmer's Best Friend, `www.ruby-lang.org/en/about/`, last visited: 2013-7-30

[15] Li-Chun Wang, Avery, An Industrial-Strength Audio Search Algorithm, `www.ee.columbia.edu/~dpwe/papers/Wang03-shazam.pdf`, last visited: 2013-9-5

[16] Apple, Taking Advantage of the Accelerate Framework, developer.apple.com/performance/acceleratefframework.html, last visited: 2013-9-11

[17] Baluja, Shumeet, Covell, Michele, Content Fingerprinting Using Wavelets, www.nhchau.com/files/cvmp_BalujaCovell.A4color.pdf, last visited: 2013-9-12

[18] Sergiu, Ciumac, 2013, Duplicate songs detector via audio fingerprinting, www.codeproject.com/Articles/206507/Duplicates-detector-via-audio-fingerprinting, last visited: 2013-9-11

[19] Techterms, http://www.techterms.com/definition/api, last visited: 2013-10-22

[20] Techterms, http://www.techterms.com/definition/boolean, last visited: 2013-10-22

[21] Rouse, Margaret, 2005, Definition: Framework, http://whatis.techtarget.com/definition/framework, last visited: 2013-10-22

[22] Hitchcock, Russell, 2011, Hardware Acceleration, http://www.windowsnetworking.com/articles-tutorials/common/Hardware-Acceleration.html, last visited: 2013-10-22

C List of Figures

D List of Tables

E Source Code

E.1 LBAudioDetective.m

```
//
//  LBAudioDetective.m
//  LBAudioDetective
//
//  Created by Laurin Brandner on 21.04.13.
//  Copyright (c) 2013 Laurin Brandner. All rights reserved.
//

#import <AVFoundation/AVFoundation.h>
#import <AudioUnit/AudioUnit.h>
#import <Accelerate/Accelerate.h>

#import "LBAudioDetective.h"
#import "LBAudioDetectiveFrame.h"

SInt32 AudioStreamBytesPerSample(AudioStreamBasicDescription asbd) {
    return asbd.mBytesPerFrame/asbd.mChannelsPerFrame;
}

const OSStatus kLBAudioDetectiveArgumentInvalid = 1;

const UInt32 kLBAudioDetectiveDefaultWindowSize = 2048;
const UInt32 kLBAudioDetectiveDefaultAnalysisStride = 64;
const UInt32 kLBAudioDetectiveDefaultNumberOfPitchSteps = 32;
const UInt32 kLBAudioDetectiveDefaultNumberOfRowsPerFrame = 128;
const UInt32 kLBAudioDetectiveDefaultSubfingerprintLength = 200;

typedef struct LBAudioDetective {
    AUGraph graph;
    AudioUnit rioUnit;

    AudioStreamBasicDescription recordingFormat;
    AudioStreamBasicDescription processingFormat;

    ExtAudioFileRef inputFile;

    LBAudioDetectiveFingerprintRef fingerprint;

    UInt32 subfingerprintLength;
    UInt32 windowSize;
    UInt32 analysisStride;
    UInt32 pitchStepCount;
    SInt16* recordBuffer;
    UInt32 recordBufferIndex;

    UInt32 maxNumberOfSubfingerprints;
    LBAudioDetectiveFrameRef* frames;
    UInt32 numberOfFrames;

    LBAudioDetectiveCallback callback;
    __unsafe_unretained id callbackHelper;

    struct FFT {
        FFTSetup setup;
        COMPLEX_SPLIT A;
        UInt32 log2n;
        UInt32 n;
        UInt32 nOver2;
    } FFT;
} LBAudioDetective;

OSStatus LBAudioDetectiveInitializeGraph(LBAudioDetectiveRef inDetective);
void LBAudioDetectiveReset(LBAudioDetectiveRef inDetective, Boolean keepFingerprint);

OSStatus LBAudioDetectiveMicrophoneOutput(void* inRefCon, AudioUnitRenderActionFlags*
    ioActionFlags, const AudioTimeStamp* inTimeStamp, UInt32 inBusNumber, UInt32
    inNumberFrames, AudioBufferList* ioData);

OSStatus LBAudioDetectiveAnalyzeIfWindowFull(LBAudioDetectiveRef inDetective, UInt32
    inNumberFrames, AudioBufferList inData, AudioStreamBasicDescription inDataFormat, const
    AudioTimeStamp* inTimeStamp, LBAudioDetectiveFrameRef ioFrame);
void LBAudioDetectiveSynthesizeFingerprint(LBAudioDetectiveRef inDetective,
    LBAudioDetectiveFrameRef* inFrames, UInt32 inNumberOfFrames,
    LBAudioDetectiveFingerprintRef* ioFingerprint);
OSStatus LBAudioDetectiveComputeFrequencies(LBAudioDetectiveRef inDetective, void* inSamples
    , UInt32 inNumberFrames, AudioStreamBasicDescription inDataFormat, UInt32
    inNumberOfFrequencyBins, Float32* outData);

OSStatus LBAudioDetectiveConvertToFormat(void* inData, AudioStreamBasicDescription
    inFromFormat, AudioStreamBasicDescription inToFormat, UInt32 inNumberFrames, void*
    outData);

#pragma mark Utilites

#define LBErrorCheck(error) (LBErrorCheckOnLine(error, __LINE__))
#define LBAssert(condition) (LBErrorCheckOnLine(!condition, __LINE__))

static inline void LBErrorCheckOnLine(OSStatus error, int line) {
    if (error == noErr) {
        return;
    }

    char errorString[7];
    *(UInt32*)(errorString+1) = CFSwapInt32HostToBig(error);
    if (isprint(errorString[1]) && isprint(errorString[2]) && isprint(errorString[3]) &&
        isprint(errorString[4])) {
        errorString[0] = errorString[5] = '\'';
        errorString[6] = '\0';
    }
    else {
        sprintf(errorString, "%d", (int)error);
    }

    fprintf(stderr, "Error %s on line %i\n", errorString, line);
}

#pragma mark -
#pragma mark (De)Allocation

LBAudioDetectiveRef LBAudioDetectiveNew() {
    size_t size = sizeof(LBAudioDetective);
    LBAudioDetective* instance = (LBAudioDetective*)malloc(size);
    memset(instance, 0, size);

    instance->recordingFormat = LBAudioDetectiveDefaultRecordingFormat();
    instance->processingFormat = LBAudioDetectiveDefaultProcessingFormat();

    instance->subfingerprintLength = kLBAudioDetectiveDefaultSubfingerprintLength;
    LBAudioDetectiveSetWindowSize(instance, kLBAudioDetectiveDefaultWindowSize);
    instance->analysisStride = kLBAudioDetectiveDefaultAnalysisStride;
    instance->pitchStepCount = kLBAudioDetectiveDefaultNumberOfPitchSteps;

    return instance;
}

OSStatus LBAudioDetectiveDispose(LBAudioDetectiveRef inDetective) {
    if (inDetective == NULL) {
        return kLBAudioDetectiveArgumentInvalid;
    }

    OSStatus error = noErr;
    error = LBAudioDetectiveStopProcessing(inDetective);
    LBErrorCheck(error);

    if (inDetective->graph) {
        error = AUGraphUninitialize(inDetective->graph);
        LBErrorCheck(error);

        error = AUGraphClose(inDetective->graph);
        LBErrorCheck(error);
    }

    if (inDetective->inputFile) {
        error = ExtAudioFileDispose(inDetective->inputFile);
        LBErrorCheck(error);
    }

    LBAudioDetectiveFingerprintDispose(inDetective->fingerprint);

    free(inDetective->FFT.A.realp);
    free(inDetective->FFT.A.imagp);
    vDSP_destroy_fftsetup(inDetective->FFT.setup);

    free(inDetective);

    return error;
}

#pragma mark -
#pragma mark Getters

AudioStreamBasicDescription LBAudioDetectiveDefaultRecordingFormat() {
    Float64 hardwareSampleRate = [[AVAudioSession sharedInstance] sampleRate];
    UInt32 bytesPerSample = sizeof(SInt32);

    AudioStreamBasicDescription asbd = {0};
    memset(&asbd, 0, sizeof(AudioStreamBasicDescription));
    asbd.mFormatID = kAudioFormatLinearPCM;
        asbd.mFormatFlags = kAudioFormatFlagIsSignedInteger|kAudioFormatFlagIsPacked;
        asbd.mBitsPerChannel = 8*bytesPerSample;
        asbd.mFramesPerPacket = 1;
        asbd.mChannelsPerFrame = 1;
        asbd.mBytesPerPacket = bytesPerSample*asbd.mFramesPerPacket;
        asbd.mBytesPerFrame = bytesPerSample*asbd.mChannelsPerFrame;
    asbd.mSampleRate = hardwareSampleRate;

    return asbd;
}

AudioStreamBasicDescription LBAudioDetectiveDefaultProcessingFormat() {
    UInt32 bytesPerSample = sizeof(Float32);
```

```
        AudioStreamBasicDescription asbd = {0};
    memset(&asbd, 0, sizeof(AudioStreamBasicDescription));
        asbd.mFormatID = kAudioFormatLinearPCM;
        asbd.mFormatFlags = kAudioFormatFlagIsFloat | kAudioFormatFlagIsPacked;
        asbd.mBitsPerChannel = 8*bytesPerSample;
        asbd.mFramesPerPacket = 1;
        asbd.mChannelsPerFrame = 1;
        asbd.mBytesPerPacket = bytesPerSample*asbd.mFramesPerPacket;
        asbd.mBytesPerFrame = bytesPerSample*asbd.mChannelsPerFrame;
        asbd.mSampleRate = 5512.0;

    return asbd;
}

Float64 LBAudioDetectiveGetRecordingSampleRate(LBAudioDetectiveRef inDetective) {
    return inDetective->recordingFormat.mSampleRate;
}

Float64 LBAudioDetectiveGetProcessingSampleRate(LBAudioDetectiveRef inDetective) {
    return inDetective->processingFormat.mSampleRate;
}

UInt32 LBAudioDetectiveGetNumberOfPitchSteps(LBAudioDetectiveRef inDetective) {
    return inDetective->pitchStepCount;
}

UInt32 LBAudioDetectiveGetSubfingerprintLength(LBAudioDetectiveRef inDetective) {
    return inDetective->subfingerprintLength;
}

UInt32 LBAudioDetectiveGetWindowSize(LBAudioDetectiveRef inDetective) {
    return inDetective->windowSize;
}

UInt32 LBAudioDetectiveGetAnalysisStride(LBAudioDetectiveRef inDetective) {
    return inDetective->analysisStride;
}

LBAudioDetectiveFingerprintRef LBAudioDetectiveGetFingerprint(LBAudioDetectiveRef
    inDetective) {
    return inDetective->fingerprint;
}

#pragma mark -
#pragma mark Setters
```

```
OSStatus LBAudioDetectiveSetRecordingSampleRate(LBAudioDetectiveRef inDetective, Float64
    inSampleRate) {
    inDetective->recordingFormat.mSampleRate = inSampleRate;

    return noErr;
}

OSStatus LBAudioDetectiveSetProcessingSampleRate(LBAudioDetectiveRef inDetective, Float64
    inSampleRate) {
    inDetective->processingFormat.mSampleRate = inSampleRate;

    return noErr;
}

OSStatus LBAudioDetectiveSetNumberOfPitchSteps(LBAudioDetectiveRef inDetective, UInt32
    inNumberOfPitchSteps) {
    inDetective->pitchStepCount = inNumberOfPitchSteps;

    return noErr;
}

OSStatus LBAudioDetectiveSetSubfingerprintLength(LBAudioDetectiveRef inDetective, UInt32
    inSubfingerprintLength) {
    inDetective->subfingerprintLength = inSubfingerprintLength;

    return noErr;
}

OSStatus LBAudioDetectiveSetWindowSize(LBAudioDetectiveRef inDetective, UInt32 inWindowSize)
     {
    OSStatus error = noErr;

    free(inDetective->FFT.A.realp);
    free(inDetective->FFT.A.imagp);
    vDSP_destroy_fftsetup(inDetective->FFT.setup);

    inDetective->windowSize = inWindowSize;

    inDetective->FFT.log2n = round(log2(inWindowSize));
    inDetective->FFT.n = (1 << inDetective->FFT.log2n);
    if (inDetective->FFT.n == inWindowSize) {
        error = kLBAudioDetectiveArgumentInvalid;
    }
    inDetective->FFT.nOver2 = inWindowSize/2;

        inDetective->FFT.A.realp = (Float32 *)calloc(inDetective->FFT.nOver2, sizeof(Float32
            ));
        inDetective->FFT.A.imagp = (Float32 *)calloc(inDetective->FFT.nOver2, sizeof(Float32
            ));
        inDetective->FFT.setup = vDSP_create_fftsetup(inDetective->FFT.log2n, FFT_RADIX2);

    return error;
}

OSStatus LBAudioDetectiveSetAnalysisStride(LBAudioDetectiveRef inDetective, UInt32
    inAnalysisStride) {
    inDetective->analysisStride = inAnalysisStride;

    return noErr;
}

#pragma mark -
#pragma mark Other Methods

OSStatus LBAudioDetectiveProcessAudioURL(LBAudioDetectiveRef inDetective, NSURL* inFileURL)
    {
    OSStatus error = noErr;

    if (!inFileURL) {
        error = kLBAudioDetectiveArgumentInvalid;
    }

    LBAudioDetectiveReset(inDetective, FALSE);

    if (inDetective->inputFile) {
        error = ExtAudioFileDispose(inDetective->inputFile);
        inDetective->inputFile = NULL;
        LBErrorCheck(error);
    }

    error = ExtAudioFileOpenURL((__bridge CFURLRef)(inFileURL), &inDetective->inputFile);
    LBErrorCheck(error);

    error = ExtAudioFileSetProperty(inDetective->inputFile,
        kExtAudioFileProperty_ClientDataFormat, sizeof(AudioStreamBasicDescription), &
        inDetective->processingFormat);
    LBErrorCheck(error);

    UInt32 propertySize = sizeof(SInt64);
    SInt64 dataLength = 0;
    error = ExtAudioFileGetProperty(inDetective->inputFile,
        kExtAudioFileProperty_FileLengthFrames, &propertySize, &dataLength);
    LBErrorCheck(error);
```

```
UInt32 numberFrames = inDetective->windowSize;
AudioBufferList bufferList;
Float32 samples[numberFrames]; // A large enough size to not have to worry about buffer
    overrun

bufferList.mNumberBuffers = 1;
bufferList.mBuffers[0].mData = samples;
bufferList.mBuffers[0].mNumberChannels = inDetective->processingFormat.mChannelsPerFrame
    ;
bufferList.mBuffers[0].mDataByteSize = numberFrames*AudioStreamBytesPerSample(
    inDetective->processingFormat);

UInt64 imageWidth = (dataLength - inDetective->windowSize)/inDetective->analysisStride;
SInt64 offset = 0;
UInt32 readNumberFrames = numberFrames;

UInt32 f = 0;
UInt32 framesCount = imageWidth/kLBAudioDetectiveDefaultNumberOfRowsPerFrame;
LBAudioDetectiveFrameRef frames[framesCount];
LBAudioDetectiveFrameRef currentFrame = NULL;
UInt32 remainingData = imageWidth%kLBAudioDetectiveDefaultNumberOfRowsPerFrame;

for (UInt64 i = 0; i < imageWidth-remainingData; i++) {
    UInt32 frameIndex = (i % kLBAudioDetectiveDefaultNumberOfRowsPerFrame);
    if (frameIndex == 0) {
        if (currentFrame) {
            frames[f] = currentFrame;
            f++;
        }

        currentFrame = LBAudioDetectiveFrameNew(
            kLBAudioDetectiveDefaultNumberOfRowsPerFrame);
    }

    error = ExtAudioFileRead(inDetective->inputFile, &readNumberFrames, &bufferList);
    LBErrorCheck(error);

    Float32 data[inDetective->pitchStepCount];
    error = LBAudioDetectiveComputeFrequencies(inDetective, (SInt16*)bufferList.mBuffers
        [0].mData, readNumberFrames, inDetective->processingFormat, inDetective->
        pitchStepCount, data);
    LBErrorCheck(error);
    LBAudioDetectiveFrameSetRow(currentFrame, data, frameIndex, inDetective->
        pitchStepCount);

    offset += inDetective->analysisStride;
    error = ExtAudioFileSeek(inDetective->inputFile, offset);
```

```
        LBErrorCheck(error);
    }
    if (currentFrame && LBAudioDetectiveFrameFull(currentFrame)) {
        frames[f] = currentFrame;
    }

    LBAudioDetectiveFingerprintRef fingerprint = LBAudioDetectiveFingerprintNew(0);
    LBAudioDetectiveSynthesizeFingerprint(inDetective, frames, framesCount, &fingerprint);

    inDetective->fingerprint = fingerprint;

    for (UInt64 i = 0; i < framesCount; i++) {
        LBAudioDetectiveFrameDispose(frames[i]);
    }

    LBAudioDetectiveReset(inDetective, TRUE);

    return error;
}

OSStatus LBAudioDetectiveProcess(LBAudioDetectiveRef inDetective, UInt32
    inMaxNumberOfSubfingerprints, LBAudioDetectiveCallback inCallback, id inCallbackHelper)
    {
    inDetective->maxNumberOfSubfingerprints = inMaxNumberOfSubfingerprints;
    inDetective->callback = inCallback;
    inDetective->callbackHelper = inCallbackHelper;
    return LBAudioDetectiveStartProcessing(inDetective);
}

OSStatus LBAudioDetectiveStartProcessing(LBAudioDetectiveRef inDetective) {
    if (inDetective->graph == NULL || inDetective->rioUnit == NULL) {
        LBAudioDetectiveInitializeGraph(inDetective);
    }

    LBAudioDetectiveReset(inDetective, FALSE);
    inDetective->recordBuffer = (SInt16*)calloc(AudioStreamBytesPerSample(inDetective->
        recordingFormat), inDetective->windowSize);

    return AUGraphStart(inDetective->graph);
}

OSStatus LBAudioDetectiveStopProcessing(LBAudioDetectiveRef inDetective) {
    OSStatus error = noErr;
    Boolean isProcessing = FALSE;

    if (inDetective->graph) {
        error = AUGraphIsRunning(inDetective->graph, &isProcessing);
        LBErrorCheck(error);
    }

    if (isProcessing) {
        error = AUGraphStop(inDetective->graph);
        LBErrorCheck(error);

        LBAudioDetectiveFingerprintRef fingerprint = LBAudioDetectiveFingerprintNew(0);
        LBAudioDetectiveSynthesizeFingerprint(inDetective, inDetective->frames, inDetective
             ->numberOfFrames, &fingerprint);
        inDetective->fingerprint = fingerprint;
        LBAudioDetectiveReset(inDetective, TRUE);
    }

    return error;
}

OSStatus LBAudioDetectiveResumeProcessing(LBAudioDetectiveRef inDetective) {
    OSStatus error = noErr;

    if (inDetective->graph == NULL) {
        LBAudioDetectiveStartProcessing(inDetective);
    }
    else {
        Boolean isProcessing = FALSE;
        error = AUGraphIsRunning(inDetective->graph, &isProcessing);
        LBErrorCheck(error);

        if (!isProcessing) {
            error = AUGraphStart(inDetective->graph);
            LBErrorCheck(error);
        }
    }

    return error;
}

OSStatus LBAudioDetectivePauseProcessing(LBAudioDetectiveRef inDetective) {
    OSStatus error = noErr;

    if (inDetective->graph != NULL) {
        Boolean isProcessing = FALSE;
        error = AUGraphIsRunning(inDetective->graph, &isProcessing);
        LBErrorCheck(error);

        if (isProcessing) {
            error = AUGraphStop(inDetective->graph);
            LBErrorCheck(error);
        }
    }

    return error;
}

#pragma mark -
#pragma mark Processing

OSStatus LBAudioDetectiveInitializeGraph(LBAudioDetectiveRef inDetective) {
    OSStatus error = noErr;

#if defined(__IPHONE_OS_VERSION_MIN_REQUIRED)
    // Create new AUGraph
    error = NewAUGraph(&inDetective->graph);
    LBErrorCheck(error);

    // Initialize rioNode (microphone input)
    AudioComponentDescription rioCD = {0};
    rioCD.componentType = kAudioUnitType_Output;
    rioCD.componentSubType = kAudioUnitSubType_RemoteIO;
    rioCD.componentManufacturer = kAudioUnitManufacturer_Apple;
    rioCD.componentFlags = 0;
    rioCD.componentFlagsMask = 0;

    AUNode rioNode;
    error = AUGraphAddNode(inDetective->graph, &rioCD, &rioNode);
    LBErrorCheck(error);

    // Open the graph so I can modify the audio units
    error = AUGraphOpen(inDetective->graph);
    LBErrorCheck(error);

    // Get initialized rioUnit
    error = AUGraphNodeInfo(inDetective->graph, rioNode, NULL, &inDetective->rioUnit);
    LBErrorCheck(error);

    // Set properties to rioUnit
    AudioUnitElement bus0 = 0, bus1 = 1;
    UInt32 onFlag = 1, offFlag = 0;
    UInt32 propertySize = sizeof(UInt32);

    // Enable microphone input
        error = AudioUnitSetProperty(inDetective->rioUnit, kAudioOutputUnitProperty_EnableIO
            , kAudioUnitScope_Input, bus1, &onFlag, propertySize);
    LBErrorCheck(error);
    // Disable speakers output
        error = AudioUnitSetProperty(inDetective->rioUnit, kAudioOutputUnitProperty_EnableIO
            , kAudioUnitScope_Output, bus0, &offFlag, propertySize);
    LBErrorCheck(error);

    // Set the stream format we want
    propertySize = sizeof(AudioStreamBasicDescription);
    error = AudioUnitSetProperty(inDetective->rioUnit, kAudioUnitProperty_StreamFormat,
        kAudioUnitScope_Input, bus0, &inDetective->recordingFormat, propertySize);
    LBErrorCheck(error);

    error = AudioUnitSetProperty(inDetective->rioUnit, kAudioUnitProperty_StreamFormat,
        kAudioUnitScope_Output, bus1, &inDetective->recordingFormat, propertySize);
    LBErrorCheck(error);

    AURenderCallbackStruct callback = {0};
    callback.inputProc = LBAudioDetectiveMicrophoneOutput;
        callback.inputProcRefCon = inDetective;
    propertySize = sizeof(AURenderCallbackStruct);
        error = AudioUnitSetProperty(inDetective->rioUnit,
            kAudioOutputUnitProperty_SetInputCallback, kAudioUnitScope_Global, bus0, &
            callback, propertySize);
    LBErrorCheck(error);

    propertySize = sizeof(UInt32);
    error = AudioUnitSetProperty(inDetective->rioUnit,
        kAudioUnitProperty_ShouldAllocateBuffer, kAudioUnitScope_Output, bus1, &offFlag,
        propertySize);
    LBErrorCheck(error);

    // Initialize Graph
    error = AUGraphInitialize(inDetective->graph);
    LBErrorCheck(error);

#endif

    return error;
}

void LBAudioDetectiveReset(LBAudioDetectiveRef inDetective, Boolean keepFingerprint) {
    if (!keepFingerprint) {
        LBAudioDetectiveFingerprintDispose(inDetective->fingerprint);
        inDetective->fingerprint = NULL;
    }

    if (inDetective->recordBuffer) {
        free(inDetective->recordBuffer);
        inDetective->recordBuffer = NULL;
    }
    inDetective->recordBufferIndex = 0;

    if (inDetective->frames) {
        for (UInt32 i = 0; i < inDetective->numberOfFrames; i++) {
            LBAudioDetectiveFrameDispose(inDetective->frames[i]);
        }
        free(inDetective->frames);
        inDetective->frames = NULL;
    }

    inDetective->maxNumberOfSubfingerprints = 0;
    inDetective->numberOfFrames = 0;
    inDetective->callback = NULL;
    inDetective->callbackHelper = nil;
}

OSStatus LBAudioDetectiveMicrophoneOutput(void* inRefCon, AudioUnitRenderActionFlags*
    ioActionFlags, const AudioTimeStamp* inTimeStamp, UInt32 inBusNumber, UInt32
    inNumberFrames, AudioBufferList* ioData) {
    LBAudioDetective* inDetective = (LBAudioDetective*)inRefCon;
    OSStatus error = noErr;

    AudioBufferList bufferList;
    SInt16 samples[inNumberFrames];
    memset(samples, 0, sizeof(samples));

    bufferList.mNumberBuffers = 1;
    bufferList.mBuffers[0].mData = samples;
    bufferList.mBuffers[0].mNumberChannels = inDetective->recordingFormat.mChannelsPerFrame;
    bufferList.mBuffers[0].mDataByteSize = inNumberFrames*AudioStreamBytesPerSample(
        inDetective->recordingFormat);

    error = AudioUnitRender(inDetective->rioUnit, ioActionFlags, inTimeStamp, 1,
        inNumberFrames, &bufferList);
    LBErrorCheck(error);

    Boolean computedAllSubfingerprints = FALSE;
    Boolean needsNewFrame = FALSE;
    LBAudioDetectiveFrameRef processingFrame = NULL;

    if (inDetective->numberOfFrames == 0) {
        needsNewFrame = TRUE;
    }
    else {
        processingFrame = inDetective->frames[inDetective->numberOfFrames-1];
        if (LBAudioDetectiveFrameFull(processingFrame)) {
            if (inDetective->numberOfFrames == inDetective->maxNumberOfSubfingerprints) {
                computedAllSubfingerprints = TRUE;
            }
            else {
                needsNewFrame = TRUE;
            }
        }
    }

    if (needsNewFrame) {
        inDetective->numberOfFrames++;
        inDetective->frames = (LBAudioDetectiveFrameRef*)realloc(inDetective->frames, sizeof
            (LBAudioDetectiveFrameRef)*inDetective->numberOfFrames);
        processingFrame = LBAudioDetectiveFrameNew(inDetective->subfingerprintLength);
        inDetective->frames[inDetective->numberOfFrames-1] = processingFrame;
    }

    if (processingFrame) {
        error = LBAudioDetectiveAnalyzeIfWindowFull(inDetective, inNumberFrames, bufferList,
             inDetective->recordingFormat, inTimeStamp, processingFrame);
        LBErrorCheck(error);
    }
    else if (computedAllSubfingerprints) {
        if (inDetective->callback) {
            dispatch_sync(dispatch_get_main_queue(), ^{
                inDetective->callback(inDetective, inDetective->callbackHelper);
            });
        }

        LBAudioDetectiveStopProcessing(inDetective);
    }

    return error;
}

OSStatus LBAudioDetectiveAnalyzeIfWindowFull(LBAudioDetectiveRef inDetective, UInt32
    inNumberFrames, AudioBufferList inData, AudioStreamBasicDescription inDataFormat, const
    AudioTimeStamp* inTimeStamp, LBAudioDetectiveFrameRef ioFrame) {
    OSStatus error = noErr;
    SInt64 delta = (SInt16)inDetective->windowSize-((SInt16)inDetective->recordBufferIndex+(
        SInt16)inNumberFrames);
    if (delta > 0) {
        // Window is not full yet
```

```
            memcpy(inDetective->recordBuffer+inDetective->recordBufferIndex, inData.mBuffers[0].
                mData, inNumberFrames*AudioStreamBytesPerSample(inDetective->recordingFormat));
            inDetective->recordBufferIndex += inNumberFrames;
        }
        else {
            // Store remaining data to the buffer in order to fill the window

            UInt32 remainingNumberFrames = (inNumberFrames+delta);
            memcpy(inDetective->recordBuffer+inDetective->recordBufferIndex, inData.mBuffers[0].
                mData, remainingNumberFrames*AudioStreamBytesPerSample(inDetective->
                recordingFormat));

            Float32 row[inDetective->pitchStepCount];
            error = LBAudioDetectiveComputeFrequencies(inDetective, inDetective->recordBuffer,
                inDetective->windowSize, inDetective->recordingFormat, inDetective->
                pitchStepCount, row);
            LBErrorCheck(error);

            LBAudioDetectiveFrameSetRow(ioFrame, row, LBAudioDetectiveFrameGetNumberOfRows(
                ioFrame), inDetective->pitchStepCount);

            inDetective->recordBufferIndex = -delta-1;
            memset(inDetective->recordBuffer, 0, inDetective->windowSize*
                AudioStreamBytesPerSample(inDetective->recordingFormat));
            memcpy(inDetective->recordBuffer, (SInt16*)inData.mBuffers[0].mData+
                remainingNumberFrames, inData.mBuffers[0].mDataByteSize-remainingNumberFrames*
                AudioStreamBytesPerSample(inDetective->recordingFormat));
        }

        return error;
    }

    void LBAudioDetectiveSynthesizeFingerprint(LBAudioDetectiveRef inDetective,
        LBAudioDetectiveFrameRef* inFrames, UInt32 inNumberOfFrames,
        LBAudioDetectiveFingerprintRef* ioFingerprint) {
        for (UInt32 i = 0; i < inNumberOfFrames; i++) {
            LBAudioDetectiveFrameRef frame = inFrames[i];

            if (LBAudioDetectiveFrameFull(frame)) {
                LBAudioDetectiveFrameDecompose(frame);
                Boolean subfingerprint[2*inDetective->subfingerprintLength];
                memset(subfingerprint, 0, sizeof(subfingerprint));

                LBAudioDetectiveFrameExtractFingerprint(frame, inDetective->subfingerprintLength
                    , subfingerprint);

                UInt32 subfingerprintLength = inDetective->subfingerprintLength;
                LBAudioDetectiveFingerprintSetSubfingerprintLength(*ioFingerprint, &
                    subfingerprintLength);
                LBAudioDetectiveFingerprintAddSubfingerprint(*ioFingerprint, subfingerprint);
            }
        }
    }

    OSStatus LBAudioDetectiveComputeFrequencies(LBAudioDetectiveRef inDetective, void* inSamples
        , UInt32 inNumberFrames, AudioStreamBasicDescription inDataFormat, UInt32
        inNumberOfFrequencyBins, Float32* outData) {
        OSStatus error = noErr;

        if (inDataFormat.mFormatFlags != inDetective->processingFormat.mFormatFlags ||
            inDataFormat.mBytesPerFrame != inDataFormat.mBytesPerFrame) {
            Float32 convertedSamples[inNumberFrames];
            error = LBAudioDetectiveConvertToFormat(inSamples, inDataFormat, inDetective->
                processingFormat, inNumberFrames, convertedSamples);
            LBErrorCheck(error);

            error = LBAudioDetectiveComputeFrequencies(inDetective, convertedSamples,
                inNumberFrames, inDetective->processingFormat, inNumberOfFrequencyBins, outData)
                ;
            LBErrorCheck(error);
        }

        Float32* samples = (Float32*)inSamples;

        vDSP_ctoz((COMPLEX*)samples, 2, &inDetective->FFT.A, 1, inDetective->FFT.nOver2);
        vDSP_fft_zrip(inDetective->FFT.setup, &inDetective->FFT.A, 1, inDetective->FFT.log2n,
            FFT_FORWARD);
        vDSP_ztoc(&inDetective->FFT.A, 1, (COMPLEX *)samples, 2, inDetective->FFT.nOver2);

        inDetective->FFT.A.imagp[0] = 0.0;

        UInt32 binsCount = inNumberOfFrequencyBins+1;
        Float64 maxFreq = inDetective->processingFormat.mSampleRate/2.0;
        Float64 minFreq = 318.0;

        Float64 logBase = exp(log(maxFreq/minFreq)/inNumberOfFrequencyBins);
        Float64 mincoef = (Float64)inDetective->windowSize/inDetective->processingFormat.
            mSampleRate*minFreq;
        UInt32 indices[binsCount];
        for (UInt32 j = 0; j < binsCount; j++) {
            UInt32 start = (UInt32)((pow(logBase, j)-1.0)*mincoef);
            indices[j] = start+(UInt32)mincoef;
        }
        UInt32 width = inNumberFrames/2.0;
        size_t size = inNumberOfFrequencyBins*sizeof(Float32*);
        memset(outData, 0, size);

        for (int i = 0; i < inNumberOfFrequencyBins; i++) {
            UInt32 lowBound = indices[i];
            UInt32 highBound = indices[i+1];
            UInt32 lowBoundIndex = ((2*lowBound)/(inDetective->processingFormat.mSampleRate/
                inNumberFrames))-1;
            UInt32 highBoundIndex = ((2*highBound)/(inDetective->processingFormat.mSampleRate/
                inNumberFrames))-1;
            Float32 p = 0.0;

            for (UInt32 k = lowBoundIndex; k < highBoundIndex; k++) {
                Float32 re = samples[2*k];
                Float32 img = samples[(2*k)+1];

                if (re > 0.0) {
                    re /= (Float32)(width/2);
                }
                if (img > 0.0) {
                    img /= (Float32)(width/2);
                }

                Float32 v = ((re*re)+(img*img));
                if (v == v && isfinite(v)) {
                    // Check if v got NaN or inf
                    p += v;
                }
            }

            outData[i] = p/(Float32)(highBound-lowBound);
        }

        return error;
    }

    #pragma mark -
    #pragma mark Utilities

    OSStatus LBAudioDetectiveConvertToFormat(void* inData, AudioStreamBasicDescription
        inFromFormat, AudioStreamBasicDescription inToFormat, UInt32 inNumberFrames, void*
        outData) {
        AudioConverterRef converter;
            OSStatus error = AudioConverterNew(&inFromFormat, &inToFormat, &converter);
        LBErrorCheck(error);
        AudioBufferList inBufferList;
        inBufferList.mNumberBuffers = 1;
        inBufferList.mBuffers[0].mNumberChannels = inFromFormat.mChannelsPerFrame;
        inBufferList.mBuffers[0].mDataByteSize = inNumberFrames*AudioStreamBytesPerSample(
            inFromFormat);
        inBufferList.mBuffers[0].mData = inData;

        AudioBufferList outBufferList;
        outBufferList.mNumberBuffers = 1;
        outBufferList.mBuffers[0].mNumberChannels = inToFormat.mChannelsPerFrame;
        outBufferList.mBuffers[0].mDataByteSize = inNumberFrames*AudioStreamBytesPerSample(
            inToFormat);
        outBufferList.mBuffers[0].mData = outData;

        error = AudioConverterConvertComplexBuffer(converter, inNumberFrames, &inBufferList, &
            outBufferList);
        LBErrorCheck(error);

        error = AudioConverterDispose(converter);
        LBErrorCheck(error);

        return error;
    }

    #pragma mark -
    #pragma mark Comparison

    OSStatus LBAudioDetectiveCompareAudioURLs(LBAudioDetectiveRef inDetective, NSURL* inFileURL1
        , NSURL* inFileURL2, UInt32 inComparisonRange, Float32* outMatch) {
        if (inComparisonRange == 0) {
            inComparisonRange = inDetective->subfingerprintLength;
        }

        OSStatus error = noErr;
        error = LBAudioDetectiveProcessAudioURL(inDetective, inFileURL1);
        LBErrorCheck(error);

        LBAudioDetectiveFingerprintRef fingerprint1 = LBAudioDetectiveFingerprintCopy(
            inDetective->fingerprint);

        error = LBAudioDetectiveProcessAudioURL(inDetective, inFileURL2);
        LBErrorCheck(error);

        LBAudioDetectiveFingerprintRef fingerprint2 = inDetective->fingerprint;

        *outMatch = LBAudioDetectiveFingerprintCompareToFingerprint(fingerprint1, fingerprint2,
            inComparisonRange);
        LBAudioDetectiveFingerprintDispose(fingerprint1);

        return error;
    }

    #pragma mark -
```

E.2 LBAudioDetectiveFrame.m

```
//
//  LBAudioDetectiveFrame.m
//  LBAudioDetective
//
//  Created by Laurin Brandner on 28.08.13.
//  Copyright (c) 2013 Laurin Brandner. All rights reserved.
//

#import "LBAudioDetectiveFrame.h"

typedef struct LBAudioDetectiveFrame {
    Float32** rows;
    UInt32 maxNumberOfRows;
    UInt32 numberOfRows;
    UInt32 rowLength;
} LBAudioDetectiveFrame;

void LBAudioDetectiveFrameDecomposeArray(Float32** ioArray, UInt32 inCount);

#pragma mark (De)Allocation

LBAudioDetectiveFrameRef LBAudioDetectiveFrameNew(UInt32 inMaxRowCount) {
    size_t size = sizeof(LBAudioDetectiveFrame);
    LBAudioDetectiveFrame* instance = malloc(size);
    memset(instance, 0, size);

    instance->rows = calloc(inMaxRowCount, sizeof(Float32*));
    instance->maxNumberOfRows = inMaxRowCount;

    return instance;
}

void LBAudioDetectiveFrameDispose(LBAudioDetectiveFrameRef inFrame) {
    if (inFrame == NULL) {
        return;
    }

    for (UInt32 i = 0; i < inFrame->numberOfRows; i++) {
        free(inFrame->rows[i]);
    }

    free(inFrame->rows);
    free(inFrame);
}
```

```
LBAudioDetectiveFrameRef LBAudioDetectiveFrameCopy(LBAudioDetectiveFrameRef inFrame) {
    LBAudioDetectiveFrame* instance = (LBAudioDetectiveFrame*)calloc(1, sizeof(
        LBAudioDetectiveFrame));

    instance->maxNumberOfRows = inFrame->maxNumberOfRows;
    instance->numberOfRows = inFrame->numberOfRows;
    instance->rowLength = inFrame->rowLength;
    instance->rows = calloc(inFrame->maxNumberOfRows, sizeof(Float32*));

    size_t rowSize = sizeof(Float32)*instance->rowLength;
    for (UInt32 i = 0; i < inFrame->numberOfRows; i++) {
        Float32* row = malloc(rowSize);
        memcpy(row, inFrame->rows[i], rowSize);
        instance->rows[i] = row;
    }

    return instance;
}

#pragma mark -
#pragma mark Getters

UInt32 LBAudioDetectiveFrameGetNumberOfRows(LBAudioDetectiveFrameRef inFrame) {
    return inFrame->numberOfRows;
}

Float32* LBAudioDetectiveFrameGetRow(LBAudioDetectiveFrameRef inFrame, UInt32 inRowIndex) {
    return inFrame->rows[inRowIndex];
}

Float32 LBAudioDetectiveFrameGetValue(LBAudioDetectiveFrameRef inFrame, UInt32 inRowIndex,
    UInt32 inColumnIndex) {
    return inFrame->rows[inRowIndex][inColumnIndex];
}

Boolean LBAudioDetectiveFrameFull(LBAudioDetectiveFrameRef inFrame) {
    return (inFrame->numberOfRows >= inFrame->maxNumberOfRows);
}

#pragma mark -
#pragma mark Setters

Boolean LBAudioDetectiveFrameSetRow(LBAudioDetectiveFrameRef inFrame, Float32* inRow, UInt32
     inRowIndex, UInt32 inCount) {
    if (LBAudioDetectiveFrameFull(inFrame)) {
        return FALSE;
    }
    size_t size = sizeof(Float32)*inCount;
    Float32* newRow = (Float32*)calloc(inCount, sizeof(Float32));
    memcpy(newRow, inRow, size);

    inFrame->rows[inRowIndex] = newRow;
    if (inFrame->rowLength == 0) {
        inFrame->rowLength = inCount;
    }
    else {
        inFrame->rowLength = MIN(inFrame->rowLength, inCount);
    }

    inFrame->numberOfRows++;
    return TRUE;
}

#pragma mark -
#pragma mark Other Methods

// Haar Wavlet Decomposition Code from http://www.codeproject.com/Articles/206507/Duplicates
    -detector-via-audio-fingerprinting

void LBAudioDetectiveFrameDecompose(LBAudioDetectiveFrameRef inFrame) {
    for (UInt32 row = 0; row < inFrame->numberOfRows; row++) {
        LBAudioDetectiveFrameDecomposeArray(&inFrame->rows[row], inFrame->rowLength);
    }

    for (UInt32 col = 0; col < inFrame->rowLength; col++) {
        Float32 column[inFrame->numberOfRows];

        for (UInt32 row = 0; row < inFrame->numberOfRows; row++) {
            column[row] = inFrame->rows[row][col];
        }

        Float32* columnPointer = (Float32*)&column;
        LBAudioDetectiveFrameDecomposeArray(&columnPointer, inFrame->numberOfRows);

        for (UInt32 row = 0; row < inFrame->numberOfRows; row++) {
            inFrame->rows[row][col] = column[row];
        }
    }
}

void LBAudioDetectiveFrameDecomposeArray(Float32** ioArray, UInt32 inCount) {
    Float32* array = *ioArray;
    for (UInt32 i = 0; i < inCount; i++) {
        array[i] /= sqrtf(inCount);
    }

    Float32 tmp[inCount];

    while (inCount > 1) {
        inCount /= 2;
        for (UInt32 i = 0; i < inCount; i++) {
            tmp[i] = ((array[2 * i] + array[2 * i + 1]) / sqrtf(2.0f));
            tmp[inCount + i] = ((array[2 * i] - array[2 * i + 1]) / sqrtf(2.0f));
        }
        for (UInt32 i = 0; i < 2*inCount; i++) {
            array[i] = tmp[i];
        }
    }
}

size_t LBAudioDetectiveFrameFingerprintSize(LBAudioDetectiveFrameRef inFrame) {
    return inFrame->numberOfRows*inFrame->rowLength*2*sizeof(Boolean);
}

UInt32 LBAudioDetectiveFrameFingerprintLength(LBAudioDetectiveFrameRef inFrame) {
    return inFrame->numberOfRows*inFrame->rowLength*2;
}

void LBAudioDetectiveFrameExtractFingerprint(LBAudioDetectiveFrameRef inFrame, UInt32
    inNumberOfWavelets, Boolean* outFingerprint) {
    NSMutableArray* array = [NSMutableArray arrayWithCapacity:inFrame->numberOfRows*inFrame
        ->rowLength];

    for (UInt32 row = 0; row < inFrame->numberOfRows; row++) {
        for (UInt32 column = 0; column < inFrame->rowLength; column++) {
            array[row*inFrame->rowLength+column] = @(inFrame->rows[row][column]);
        }
    }

    [array sortUsingComparator:^NSComparisonResult(NSNumber* obj1, NSNumber* obj2) {
        return [@(fabs(obj2.doubleValue)) compare:@(fabs(obj1.doubleValue))];
    }];

    for (UInt32 i = 0; i < inNumberOfWavelets; i++) {
        Float64 value = ((NSNumber*)array[i]).doubleValue;
        if (value > 0.0) {
            outFingerprint[2*i] = TRUE;
        }
        else if (value < 0.0) {
            outFingerprint[(2*i)+1] = TRUE;
        }
    }
}

Boolean LBAudioDetectiveFrameEqualToFrame(LBAudioDetectiveFrameRef inFrame1,
    LBAudioDetectiveFrameRef inFrame2) {
    if ((inFrame1->rowLength != inFrame2->rowLength) || (inFrame1->numberOfRows != inFrame2
        ->numberOfRows)) {
        return FALSE;
    }

    for (UInt32 r = 0; r < inFrame1->numberOfRows ; r++) {
        if (memcmp(inFrame1->rows[r], inFrame2->rows[r], inFrame1->rowLength*sizeof(Float32)
            ) != 0) {
//            for (UInt32 c = 0; c < inFrame1->rowLength; c++) {
//                if (inFrame1->rows[r][c] != inFrame2->rows[r][c]) {
//                    NSLog(@"r:%u c:%u %f != %f", (unsigned int)r, (unsigned int)c,
    inFrame1->rows[r][c], inFrame2->rows[r][c]);
//                }
//            }
            return FALSE;
        }
    }

    return TRUE;
}

#pragma mark -
```

E.3 LBAudioDetectiveFingerprint.m

```
//
//  LBAudioDetectiveFingerprint.m
//  LBAudioDetective
//
//  Created by Laurin Brandner on 30.08.13.
//  Copyright (c) 2013 Laurin Brandner. All rights reserved.
//

#import "LBAudioDetectiveFingerprint.h"
typedef struct LBAudioDetectiveFingerprint {
    Boolean** subfingerprints;
    UInt32 subfingerprintLength;
    UInt32 subfingerprintCount;
} LBAudioDetectiveFingerprint;

#pragma mark (De)Allocation

LBAudioDetectiveFingerprintRef LBAudioDetectiveFingerprintNew(UInt32 inSubfingerprintLength)
     {
    size_t size = sizeof(LBAudioDetectiveFingerprint);
    LBAudioDetectiveFingerprint* instance = malloc(size);
    memset(instance, 0, size);

    instance->subfingerprintLength = inSubfingerprintLength;

    return instance;
}

void LBAudioDetectiveFingerprintDispose(LBAudioDetectiveFingerprintRef inFingerprint) {
    if (inFingerprint == NULL) {
        return;
    }

    for (UInt32 i = 0; i < inFingerprint->subfingerprintCount; i++) {
        free(inFingerprint->subfingerprints[i]);
    }

    free(inFingerprint->subfingerprints);
    free(inFingerprint);
}

LBAudioDetectiveFingerprintRef LBAudioDetectiveFingerprintCopy(
    LBAudioDetectiveFingerprintRef inFingerprint) {
    size_t size = sizeof(LBAudioDetectiveFingerprint);
    LBAudioDetectiveFingerprint* instance = malloc(size);
    memset(instance, 0, size);

    instance->subfingerprintLength = inFingerprint->subfingerprintLength;
    instance->subfingerprintCount = inFingerprint->subfingerprintCount;
    instance->subfingerprints = (Boolean**)calloc(instance->subfingerprintCount, sizeof(
        Boolean*));

    size = sizeof(Boolean)*instance->subfingerprintLength;
    for (UInt32 i = 0; i < instance->subfingerprintCount; i++) {
        Boolean* subfingerprint = (Boolean*)calloc(instance->subfingerprintLength, sizeof(
            Boolean));
        memcpy(subfingerprint, inFingerprint->subfingerprints[i], size);

        instance->subfingerprints[i] = subfingerprint;
    }

    return instance;
}

#pragma mark -
#pragma mark Getters

UInt32 LBAudioDetectiveFingerprintGetSubfingerprintLength(LBAudioDetectiveFingerprintRef
    inFingerprint) {
    return inFingerprint->subfingerprintLength;
}

UInt32 LBAudioDetectiveFingerprintGetNumberOfSubfingerprints(LBAudioDetectiveFingerprintRef
    inFingerprint) {
    return inFingerprint->subfingerprintCount;
}

UInt32 LBAudioDetectiveFingerprintGetSubfingerprintAtIndex(LBAudioDetectiveFingerprintRef
    inFingerprint, UInt32 inIndex, Boolean* outSubfingerprint) {
    memcpy(outSubfingerprint, inFingerprint->subfingerprints[inIndex], inFingerprint->
        subfingerprintLength*sizeof(Boolean));

    return inFingerprint->subfingerprintLength;
}

#pragma mark -
#pragma mark Setters

Boolean LBAudioDetectiveFingerprintSetSubfingerprintLength(LBAudioDetectiveFingerprintRef
    inFingerprint, UInt32* ioSubfingerprintLength) {
    if (inFingerprint->subfingerprintCount > 0) {
        *ioSubfingerprintLength = inFingerprint->subfingerprintLength;
        return FALSE;
    }

    inFingerprint->subfingerprintLength = *ioSubfingerprintLength;
    return TRUE;
}

void LBAudioDetectiveFingerprintAddSubfingerprint(LBAudioDetectiveFingerprintRef
    inFingerprint, Boolean* inSubfingerprint) {
    size_t size = sizeof(Boolean)*inFingerprint->subfingerprintLength;
    Boolean* newSubfingerprint = (Boolean*)calloc(inFingerprint->subfingerprintLength,
        sizeof(Boolean));
    memcpy(newSubfingerprint, inSubfingerprint, size);

    inFingerprint->subfingerprintCount++;
    size = sizeof(Boolean*)*inFingerprint->subfingerprintCount;
    inFingerprint->subfingerprints = (Boolean**)realloc(inFingerprint->subfingerprints, size
        );
    inFingerprint->subfingerprints[inFingerprint->subfingerprintCount-1] = newSubfingerprint
        ;
}

#pragma mark -
#pragma mark Other Methods

Boolean LBAudioDetectiveFingerprintEqualToFingerprint(LBAudioDetectiveFingerprintRef
    inFingerprint1, LBAudioDetectiveFingerprintRef inFingerprint2) {
    if (inFingerprint1->subfingerprintCount != inFingerprint2->subfingerprintCount ||
        inFingerprint1->subfingerprintLength != inFingerprint2->subfingerprintLength) {
        return FALSE;
    }

    for (UInt32 i = 0; i < inFingerprint1->subfingerprintCount; i++) {
        if (memcmp(inFingerprint1->subfingerprints[i], inFingerprint2->subfingerprints[i],
            sizeof(Boolean)*inFingerprint1->subfingerprintLength) != 0) {
            return FALSE;
        }
    }

    return TRUE;
}

Float32 LBAudioDetectiveFingerprintCompareToFingerprint(LBAudioDetectiveFingerprintRef
    inFingerprint1, LBAudioDetectiveFingerprintRef inFingerprint2, UInt32 inRange) {
    UInt32 subfingerprintCount1 = inFingerprint1->subfingerprintCount;
    UInt32 subfingerprintCount2 = inFingerprint2->subfingerprintCount;
    if (inFingerprint1->subfingerprintCount < inFingerprint2->subfingerprintCount) {
        LBAudioDetectiveFingerprintRef tmpFingerprint = inFingerprint1;
        inFingerprint1 = inFingerprint2;
        inFingerprint2 = tmpFingerprint;

        UInt32 tmpSubfingerprintCount = subfingerprintCount1;
        subfingerprintCount1 = subfingerprintCount2;
        subfingerprintCount2 = tmpSubfingerprintCount;
    }

    Float32 match = 0.0f;
    UInt32 offset = 0;

    while (offset <= subfingerprintCount1-subfingerprintCount2) {
        Float32 matchesSum = 0.0f;

        for (UInt32 i = 0; i < subfingerprintCount2; i++) {
            Float32 currentMatch = LBAudioDetectiveFingerprintCompareSubfingerprints(
                inFingerprint1, inFingerprint1->subfingerprints[i+offset], inFingerprint2->
                subfingerprints[i], inRange);
            matchesSum += currentMatch;
        }

        match = MAX(match, matchesSum/(Float32)subfingerprintCount2);
        offset++;
    }

    return match;
}

Float32 LBAudioDetectiveFingerprintCompareSubfingerprints(LBAudioDetectiveFingerprintRef
    inFingerprint, Boolean* inSubfingerprint1, Boolean* inSubfingerprint2, UInt32 inRange) {
    UInt32 possibleHits = 0;
    UInt32 hits = 0;

    for (UInt32 i = 0; i < MIN(inRange, inFingerprint->subfingerprintLength); i += 2) {
        Boolean sf1s1 = inSubfingerprint1[i];
        Boolean sf1s2 = inSubfingerprint1[i+1];

        if (sf1s1 || sf1s2) {
            possibleHits++;

            Boolean sf2s1 = inSubfingerprint2[i];
            Boolean sf2s2 = inSubfingerprint2[i+1];

            if ((sf1s1 == sf2s1) && (sf1s2 == sf2s2)) {
                hits++;
            }
        }
    }

    if (possibleHits <= 0) {
        return 0.0f;
    }

    return (Float32)hits/(Float32)possibleHits;
}

#pragma mark -
```

E.4 WHWhistlesClient.m

```
//
//  WHWhistlesClient.m
//  Whistles
//
//  Created by Laurin Brandner on 11.08.13.
//  Copyright (c) 2013 Laurin Brandner. All rights reserved.
//

#import "WHWhistlesClient.h"

NSString* const WHWhistlesBaseURL = @"http://whistles.herokuapp.com/";
NSString* const WHWhistlesAPIIdentifiyURL = @"/birds/identify.json";
NSString* const WHWhistlesAPIBugURL = @"/bug.json";

@interface WHWhistlesClient ()

+(NSString*)stringFromRecording:(NSURL*)path;

-(void)identifyEncodedFingerprint:(NSString*)string completion:(void (^)(NSDictionary *,
    NSError *))completion;

@end
@implementation WHWhistlesClient

#pragma mark Initialization

+(instancetype)sharedClient {
    static WHWhistlesClient* sharedClient = nil;
    static dispatch_once_t onceToken;
    dispatch_once(&onceToken, ^{
        sharedClient = [WHWhistlesClient new];
    });

    return sharedClient;
}

-(id)init {
    self = [super initWithBaseURL:[self.class defaultWhistlesURL]];
    if (self) {
        WHWhistlesClientInitialize(self);
    }

    return self;
}
```

```
-(id)initWithCoder:(NSCoder *)aDecoder {
    self = [super initWithCoder:aDecoder];
    if (self) {
        WHWhistlesClientInitialize(self);
    }

    return self;
}

-(id)initWithBaseURL:(NSURL *)url {
    self = [super initWithBaseURL:url];
    if (self) {
        WHWhistlesClientInitialize(self);
    }

    return self;
}

void WHWhistlesClientInitialize(WHWhistlesClient* client) {
    client.parameterEncoding = AFJSONParameterEncoding;
}

#pragma mark -
#pragma mark Other Methods

+(NSURL*)defaultWhistlesURL {
#if TARGET_IPHONE_SIMULATOR
    NSURL* URL = [NSURL URLWithString:@"http://localhost:3000"];
#else
    NSURL* URL = [NSURL URLWithString:WHWhistlesBaseURL];
#endif
    return URL;
}

+(NSString*)stringFromFingerprint:(LBAudioDetectiveFingerprintRef)fingerprint {
    NSMutableArray* array = [NSMutableArray new];
    NSUInteger subfingerprintLength = LBAudioDetectiveFingerprintGetSubfingerprintLength(
        fingerprint);
    for (NSUInteger i = 0; i < LBAudioDetectiveFingerprintGetNumberOfSubfingerprints(
        fingerprint); i++) {
        Boolean subfingerprint[subfingerprintLength];
        LBAudioDetectiveFingerprintGetSubfingerprintAtIndex(fingerprint, i, subfingerprint);
        NSMutableString* subfingerprintString = [NSMutableString new];
        for (NSUInteger j = 0; j < subfingerprintLength; j++) {
            [subfingerprintString appendString:[NSString stringWithFormat:@"%i",
                subfingerprint[j]]];
        }
        [array addObject:subfingerprintString];
    }

    return [array componentsJoinedByString:@"+"];
}

+(NSString*)stringFromAudio:(NSURL *)URL {
    NSData* data = [NSData dataWithContentsOfURL:URL];
    return [data base64EncodedStringWithOptions:0];
}

+(NSString*)stringFromRecording:(NSURL*)path {
    LBAudioDetectiveRef detective = LBAudioDetectiveNew();
    OSStatus error = LBAudioDetectiveProcessAudioURL(detective, path);
    if (error == noErr) {
        LBAudioDetectiveFingerprintRef fingerprint = LBAudioDetectiveGetFingerprint(
            detective);
        NSString* string = [self.class stringFromFingerprint:fingerprint];
        LBAudioDetectiveDispose(detective);

        return string;
    }
    return nil;
}

-(void)identifyRecording:(NSURL *)path completion:(void (^)(NSDictionary *, NSError *))
    completion {
    [self identifyEncodedFingerprint:[WHWhistlesClient stringFromRecording:path] completion:
        completion];
}

-(void)identifyRecordedFingerprint:(LBAudioDetectiveFingerprintRef)fingerprint completion:(
    void (^)(NSDictionary *, NSError *))completion {
    [self identifyEncodedFingerprint:[self.class stringFromFingerprint:fingerprint]
        completion:completion];
}

-(void)identifyEncodedFingerprint:(NSString*)string completion:(void (^)(NSDictionary *,
    NSError *))completion {
    if (!string.length) {
        string = [NSString string];
    }

    NSDictionary* parameters = @{@"fingerprint": string};
    NSMutableURLRequest* request = [self requestWithMethod:@"POST" path:
        WHWhistlesAPIIdentifiyURL parameters:parameters];
    AFJSONRequestOperation* operation = [AFJSONRequestOperation
        JSONRequestOperationWithRequest:request success:nil failure:nil];
    if (completion) {
        [operation setCompletionBlockWithSuccess:^(AFHTTPRequestOperation *operation,
            NSDictionary* JSON) {
            completion(JSON, nil);
        } failure:^(AFHTTPRequestOperation *operation, NSError *error) {
            completion(nil, error);
        }];
    }
    [self enqueueHTTPRequestOperation:operation];
}

-(void)reportBugForRecording:(NSURL *)path description:(NSString *)description completion:(
    void (^)(NSError *))completion {
    NSString* audioString = [WHWhistlesClient stringFromAudio:path];
    if (!audioString.length) {
        audioString = [NSString string];
    }

    NSMutableDictionary* parameters = [NSMutableDictionary dictionaryWithObject:audioString
        forKey:@"audio"];
    if (description) {
        [parameters setObject:description forKey:@"description"];
    }

    NSMutableURLRequest* request = [self requestWithMethod:@"POST" path:WHWhistlesAPIBugURL
        parameters:parameters];
    AFJSONRequestOperation* operation = [AFJSONRequestOperation
        JSONRequestOperationWithRequest:request success:nil failure:nil];
    if (completion) {
        [operation setCompletionBlockWithSuccess:^(AFHTTPRequestOperation *operation, id
            JSON) {
            completion(nil);
        } failure:^(AFHTTPRequestOperation *operation, NSError *error) {
            completion(error);
        }];
    }
    [self enqueueHTTPRequestOperation:operation];
}

#pragma mark -

@end
```

E.5 WHBirdsViewController.m

```
//
//  WHViewController.m
//  Whistles
//
//  Created by Laurin Brandner on 30.07.13.
//  Copyright (c) 2013 Laurin Brandner. All rights reserved.
//

#import <AVFoundation/AVFoundation.h>
#import "WHBirdsViewController.h"
#import "WHBirdTableViewCell.h"
#import "WHBirdViewController.h"
#import "WHBugViewController.h"
#import "WHToolbar.h"
#import "WHWhistlesClient.h"

NSString* const WHBirdTableViewCellIdentifier = @"WHTableViewCellIdentifier";
const NSUInteger WHBirdsViewControllerMaxBirdCount = 50;

struct _WHBirdsViewControllerGeometry WHBirdsViewControllerGeometry = {
    .recordButtonRadius = 30.0f,
    .inputImageInset = 2.0f
};

@interface WHBirdsViewController () <UIActionSheetDelegate>

@property (nonatomic) NSTimeInterval recordingTimestamp;
@property (nonatomic, readonly) CLLocationCoordinate2D currentLocation;

@property (nonatomic, strong) WHRecordButton* recordButton;
@property (nonatomic, strong) UIActionSheet* actionSheet;

-(NSURL*)URLForRecording:(NSTimeInterval)timeStamp;
-(void)startRecording:(id)sender;
-(void)stopRecording:(id)sender;

-(void)reidentifyUnknownBirds;

-(void)activateAudioSession:(BOOL)activate;
-(void)availableInputPortsHaveChanged:(NSNotification*)notification;

+(NSDictionary*)defaultRecordSettings;

@end
@implementation WHBirdsViewController
```

```
#pragma mark Accessors

-(CLLocationCoordinate2D)currentLocation {
    return self.locationManager.location.coordinate;
}

-(NSFetchedResultsController*)controller {
    if (_controller) {
        return _controller;
    }

    NSFetchRequest* request = [NSFetchRequest fetchRequestWithEntityName:NSStringFromClass([
        WHBird class])];
    request.sortDescriptors = @[[NSSortDescriptor sortDescriptorWithKey:@"recordTimestamp"
        ascending:NO]];
    self.controller = [[NSFetchedResultsController alloc] initWithFetchRequest:request
        managedObjectContext:[NSManagedObjectContext sharedContext] sectionNameKeyPath:@"
        recordDate.day" cacheName:nil];
    _controller.delegate = self;
    [_controller performFetch:nil];

    return _controller;
}

#pragma mark -
#pragma mark Initialization

-(id)init {
    self = [super init];
    if (self) {
        WHBirdsViewControllerInitialize(self);
    }

    return self;
}

-(id)initWithCoder:(NSCoder *)aDecoder {
    self = [super initWithCoder:aDecoder];
    if (self) {
        WHBirdsViewControllerInitialize(self);
    }

    return self;
}

void WHBirdsViewControllerInitialize(WHBirdsViewController* controller) {
    controller.title = @"Whistles";

    controller.locationManager = [CLLocationManager new];
    controller.locationManager.distanceFilter = kCLDistanceFilterNone;
    controller.locationManager.desiredAccuracy = kCLLocationAccuracyHundredMeters;
    [controller.locationManager startUpdatingLocation];

    WHWhistlesClient* client = [WHWhistlesClient sharedClient];
    __weak WHBirdsViewController* weakController = controller;
    [client setReachabilityStatusChangeBlock:^(AFNetworkReachabilityStatus status) {
        if (status == AFNetworkReachabilityStatusReachableViaWWAN || status ==
            AFNetworkReachabilityStatusReachableViaWiFi) {
            [weakController reidentifyUnknownBirds];
        }
    }];

    NSNotificationCenter* center = [NSNotificationCenter defaultCenter];
    [center addObserver:controller selector:@selector(availableInputPortsHaveChanged:) name:
        AVAudioSessionRouteChangeNotification object:nil];
    [center addObserver:controller selector:@selector(abortRecording:) name:
        UIApplicationWillTerminateNotification object:nil];
}

-(void)dealloc {
    [[NSNotificationCenter defaultCenter] removeObserver:self];
}

#pragma mark -
#pragma mark View Lifecycle

-(void)loadView {
    UITableView* newTableView = [UITableView new];
    newTableView.autoresizingMask = UIViewAutoresizingFlexibleHeight|
        UIViewAutoresizingFlexibleWidth;
    newTableView.delegate = self;
    newTableView.dataSource = self;
    newTableView.rowHeight = WHBirdTableViewCellGeometry.height;
    self.tableView = newTableView;
}

-(void)viewDidLoad {
    [super viewDidLoad];

    UIBarButtonItem* bugItem = [[UIBarButtonItem alloc] initWithTitle:NSLocalizedString(@"
        Report a bug", nil) style:UIBarButtonItemStylePlain target:self action:@selector(
        showBugReporter:)];
    bugItem.enabled = ([self tableView:self.tableView numberOfRowsInSection:0] > 0);
    self.navigationItem.leftBarButtonItem = bugItem;

    self.recordButton = [WHRecordButton new];
    self.recordButton.radius = WHBirdsViewControllerGeometry.recordButtonRadius;
    self.recordButton.delegate = self;
    self.recordButton.animationDuration = 4.0f;
    self.recordButton.color = self.navigationController.toolbar.tintColor;
    UIBarButtonItem* recordingItem = [[UIBarButtonItem alloc] initWithCustomView:self.
        recordButton];

    UIBarButtonItem* inputItem = [[UIBarButtonItem alloc] initWithImage:[UIImage
        birdsViewControllerInputImage] style:UIBarButtonItemStylePlain target:self action:
        @selector(showChannelSelection:)];
    UIBarButtonItem* revertItem = [[UIBarButtonItem alloc] initWithBarButtonSystemItem:
        UIBarButtonSystemItemTrash target:self action:@selector(abortRecording:)];

    self.toolbarItems = @[inputItem, recordingItem, revertItem];
    ((WHToolbar*)self.navigationController.toolbar).centeredCustomView = recordingItem.
        customView;
    [(WHToolbar*)self.navigationController.toolbar setOffset:CGPointMake(0.0f, -
        WHBirdsViewControllerGeometry.inputImageInset) forItemAtIndex:0];
    self.navigationController.toolbarHidden = NO;
}

#pragma mark -
#pragma mark NSFetchedResultsController

-(void)controllerDidChangeContent:(NSFetchedResultsController *)controller {
    [self.tableView reloadData];
    self.navigationItem.leftBarButtonItem.enabled = ([self tableView:self.tableView
        numberOfRowsInSection:0] > 0);
}

#pragma mark -
#pragma mark UITableViewDataSource

-(NSInteger)numberOfSectionsInTableView:(UITableView *)tableView {
    return self.controller.sections.count;
}

-(NSInteger)tableView:(UITableView *)tableView numberOfRowsInSection:(NSInteger)section {
    NSArray* sections = self.controller.sections;
    if (sections.count > 0) {
        id <NSFetchedResultsSectionInfo> sectionInfo = [sections objectAtIndex:section];
        return [sectionInfo numberOfObjects];
    }
    return 0;
```

```
}

-(CGFloat)tableView:(UITableView *)tableView heightForRowAtIndexPath:(NSIndexPath *)
    indexPath {
    return WHBirdTableViewCellGeometry.height;
}

-(UITableViewCell*)tableView:(UITableView *)tableView cellForRowAtIndexPath:(NSIndexPath *)
    indexPath {
    WHBird* bird = [self.controller objectAtIndexPath:indexPath];
    WHBirdTableViewCell* cell = [tableView dequeueReusableCellWithIdentifier:
        WHBirdTableViewCellIdentifier];
    if (!cell) {
        cell = [WHBirdTableViewCell cellWithIdentifier:WHBirdTableViewCellIdentifier];
    }

    cell.bird = bird;

    return cell;
}

-(NSString*)tableView:(UITableView *)tableView titleForHeaderInSection:(NSInteger)section {
    NSIndexPath* indexPath = [NSIndexPath indexPathForRow:0 inSection:section];
    WHBird* bird = [self.controller objectAtIndexPath:indexPath];

    NSDateFormatter* formatter = [NSDateFormatter new];
    formatter.dateStyle = NSDateFormatterLongStyle;
    formatter.timeStyle = NSDateFormatterNoStyle;

    return [formatter stringFromDate:bird.recordDate];
}

#pragma mark -
#pragma mark UITableViewDelegate

-(void)tableView:(UITableView *)tableView didSelectRowAtIndexPath:(NSIndexPath *)indexPath {
    WHBird* bird = [self.controller objectAtIndexPath:indexPath];
    WHBirdViewController* controller = [[WHBirdViewController alloc] initWithBird:bird];
    [self.navigationController pushViewController:controller animated:YES];

    [self.tableView deselectRowAtIndexPath:indexPath animated:YES];
}

-(BOOL)tableView:(UITableView *)tableView canEditRowAtIndexPath:(NSIndexPath *)indexPath {
    return YES;
}
```

```
-(UITableViewCellEditingStyle)tableView:(UITableView *)tableView
    editingStyleForRowAtIndexPath:(NSIndexPath *)indexPath {
    return UITableViewCellEditingStyleDelete;
}

-(BOOL)tableView:(UITableView *)tableView shouldIndentWhileEditingRowAtIndexPath:(
    NSIndexPath *)indexPath {
    return YES;
}

-(void)tableView:(UITableView *)tableView commitEditingStyle:(UITableViewCellEditingStyle)
    editingStyle forRowAtIndexPath:(NSIndexPath *)indexPath {
    if (editingStyle == UITableViewCellEditingStyleDelete) {
        NSManagedObjectContext* context = [NSManagedObjectContext sharedContext];
        WHBird* bird = [self.controller objectAtIndexPath:indexPath];
        if (bird) {
            [context deleteObject:bird];
            [context save];
        }
    }
}

#pragma mark -
#pragma mark WHRecordButtonDelegate

-(void)recordButtonDidStartAnimating:(WHRecordButton *)recordButton {
    [self startRecording:recordButton];
}

-(void)recordButtonDidPauseAnimating:(WHRecordButton *)recordButton {
    [self pauseRecording:recordButton];
}

-(void)recordButtonDidResumeAnimating:(WHRecordButton *)recordButton {
    [self resumeRecording:recordButton];
}

-(void)recordButtonDidStopAnimating:(WHRecordButton *)recordButton {
    [self stopRecording:recordButton];
}

#pragma mark -
#pragma mark UIActionSheetDelegate

-(void)actionSheet:(UIActionSheet *)actionSheet willDismissWithButtonIndex:(NSInteger)
    buttonIndex {
    self.actionSheet = nil;
```

```
}

-(void)actionSheet:(UIActionSheet *)actionSheet didDismissWithButtonIndex:(NSInteger)
   buttonIndex {
    if (buttonIndex != actionSheet.cancelButtonIndex) {
        AVAudioSessionPortDescription* selectedInput = nil;
        AVAudioSession* session = [AVAudioSession sharedInstance];
        NSArray* availableInputPorts = session.availableInputs;

        if (availableInputPorts.count > buttonIndex) {
            selectedInput = [availableInputPorts objectAtIndex:buttonIndex];
        }

        if (selectedInput) {
            NSError* error = nil;
            [session setPreferredInput:selectedInput error:&error];
            if (error) {
                [[UIApplication sharedApplication] showErrorMessage:NSLocalizedString(@"
                   Couldn’t select this input", nil)];
                return;
            }
        }
    }
}

#pragma mark -
#pragma mark Other Methods

+(NSDictionary*)defaultRecordSettings {
    AudioStreamBasicDescription asbd = LBAudioDetectiveDefaultRecordingFormat();
    NSMutableDictionary* settings = [NSMutableDictionary new];

    [settings setObject:@(kAudioFormatLinearPCM) forKey:AVFormatIDKey];
    [settings setObject:@(asbd.mSampleRate) forKey:AVSampleRateKey];
    [settings setObject:@(asbd.mChannelsPerFrame) forKey:AVNumberOfChannelsKey];
    [settings setObject:@(asbd.mBitsPerChannel) forKey:AVLinearPCMBitDepthKey];
    [settings setObject:@(NO) forKey:AVLinearPCMIsBigEndianKey];
    [settings setObject:@(NO) forKey:AVLinearPCMIsFloatKey];

    return settings;
}

-(NSURL*)URLForRecording:(NSTimeInterval)timeStamp {
    NSArray* paths = NSSearchPathForDirectoriesInDomains(NSDocumentDirectory,
       NSUserDomainMask, YES);
    NSString* basePath = ([paths count] > 0) ? [paths objectAtIndex:0] : nil;
    NSString* path = [basePath stringByAppendingPathComponent:[NSString stringWithFormat:@"
        recording-%.0f.caf", timeStamp]];

    return [NSURL fileURLWithPath:path];
}

-(void)startRecording:(id)sender {
    if (self.recorder.isRecording) {
        [self abortRecording:sender];
    }

    NSError* error = nil;
    self.recordingTimestamp = [[NSDate date] timeIntervalSince1970];
    self.recorder = [[AVAudioRecorder alloc] initWithURL:[self URLForRecording:self.
        recordingTimestamp] settings:[self.class defaultRecordSettings] error:&error];
    self.recorder.meteringEnabled = YES;

    [self activateAudioSession:YES];
    [self.recorder record];
}

-(void)resumeRecording:(id)sender {
    [self.recorder record];
}

-(void)pauseRecording:(id)sender {
    [self.recorder pause];
}

-(void)stopRecording:(id)sender {
    self.recordButton.progress = 0.0f;
    [self.recorder stop];
    [self activateAudioSession:NO];

    NSManagedObjectContext* context = [NSManagedObjectContext sharedContext];
    if (self.controller.fetchedObjects.count > WHBirdsViewControllerMaxBirdCount) {
        NSError* error = nil;
        NSFetchRequest* request = [NSFetchRequest fetchRequestWithEntityName:
            NSStringFromClass([WHBird class])];
        request.sortDescriptors = @[[NSSortDescriptor sortDescriptorWithKey:@"
            recordTimestamp" ascending:YES]];
        request.fetchLimit = 1;
        NSArray* birds = [context executeFetchRequest:request error:&error];
        if (error) {
            [[UIApplication sharedApplication] showErrorMessage:NSLocalizedString(@"Couldn't
                keep the archive limit. The oldest birds cannot be deleted anymore.", nil)
                ];
        }
        else {
            WHBird* oldestBird = birds.lastObject;
            [context deleteObject:oldestBird];
        }
    }

    WHBird* newBird = [[WHBird alloc] initWithEntity:[WHBird entityInContext:context]
        insertIntoManagedObjectContext:context];
    newBird.recordTimestamp = self.recordingTimestamp;
    newBird.recordURL = [self URLForRecording:self.recordingTimestamp];
    newBird.recordLocation = self.currentLocation;
    [context save];

    [[WHWhistlesClient sharedClient] identifyRecording:newBird.recordURL completion:^(
        NSDictionary* result, NSError* error) {
        if (error) {
            [[UIApplication sharedApplication] showErrorMessage:NSLocalizedString(@"Couldn’t
                 communicate with the server. Make sure your internet connection is working
                .", nil)];
        }

        [newBird mapAttributes:result];
        [context save];

        if (newBird.identified) {
            WHBirdViewController* controller = (WHBirdViewController*)self.
                navigationController.topViewController;
            if ([controller isEqual:self]) {
                WHBirdViewController* controller = [[WHBirdViewController alloc]
                    initWithBird:newBird];
                [self.navigationController pushViewController:controller animated:YES];
            }
            else if ([controller isKindOfClass:[WHBirdViewController class]]) {
                if ([controller.bird isEqual:newBird]) {
                    [controller reloadBirdData];
                }
            }
        }

        NSString* message = [result objectForKey:@"message"];
        if (message) {
            [[UIApplication sharedApplication] showErrorMessage:NSLocalizedString(message,
                nil)];
        }
    }];
```

```
    self.recordingTimestamp = 0.0;
}

-(void)abortRecording:(id)sender {
    self.recordButton.progress = 0.0f;
    [self.recorder stop];

    NSURL* recordingURL = [self URLForRecording:self.recordingTimestamp];
    self.recordingTimestamp = 0.0;
    [self.recordButton stopAnimating];
    self.recordButton.progress = 0.0f;

    NSFileManager* manager = [NSFileManager new];
    [manager removeItemAtURL:recordingURL error:nil];
}

-(void)reidentifyUnknownBirds {
    NSManagedObjectContext* context = [NSManagedObjectContext sharedContext];
    NSFetchRequest* request = self.controller.fetchRequest.copy;
    request.predicate = [NSPredicate predicateWithFormat:@"name = nil"];
    NSError* error = nil;
    NSArray* unidentifiedBirds = [context executeFetchRequest:request error:&error];
    if (error) {
        return;
    }

    [unidentifiedBirds enumerateObjectsUsingBlock:^(WHBird* bird, NSUInteger idx, BOOL *stop
        ) {
        [[WHWhistlesClient sharedClient] identifyRecording:bird.recordURL completion:^(
            NSDictionary* result, NSError* error) {
            [bird mapAttributes:result];

            if (unidentifiedBirds.count-1 == idx) {
                [context save];
            }
        }];
    }];
}

-(void)showChannelSelection:(id)sender {
    AVAudioSession* session = [AVAudioSession sharedInstance];
    self.actionSheet = [[UIActionSheet alloc] initWithTitle:NSLocalizedString(@"Available
        Input Sources", nil) delegate:self cancelButtonTitle:nil destructiveButtonTitle:nil
        otherButtonTitles:nil];

    [session.availableInputs enumerateObjectsUsingBlock:^(AVAudioSessionPortDescription*
        input, NSUInteger idx, BOOL *stop) {
```

```
        [self.actionSheet addButtonWithTitle:input.portName];
    }];

    [self.actionSheet addButtonWithTitle:NSLocalizedString(@"Cancel", nil)];
    self.actionSheet.cancelButtonIndex = self.actionSheet.numberOfButtons-1;

    [self.actionSheet showFromBarButtonItem:sender animated:YES];
}

-(void)showBugReporter:(id)sender {
    UINavigationController* controller = [[UINavigationController alloc]
        initWithRootViewController:[WHBugViewController new]];
    [self presentViewController:controller animated:YES completion:nil];
}

-(void)activateAudioSession:(BOOL)activate {
    NSString* message = NSLocalizedString(@"An error occured while setting up the audio
        input.", nil);
    NSError* error = nil;

    AVAudioSession* session = [AVAudioSession sharedInstance];

    [session setActive:activate error:&error];
    if (error) {
        [[UIApplication sharedApplication] showErrorMessage:message];
    }
}

-(void)availableInputPortsHaveChanged:(NSNotification*)notification {
    AVAudioSession* session = [AVAudioSession sharedInstance];
    NSArray* availableInputPorts = session.availableInputs;

    if (self.actionSheet && (self.actionSheet.numberOfButtons != availableInputPorts.count
        +1)) {
        [self.actionSheet dismissWithClickedButtonIndex:self.actionSheet.cancelButtonIndex
            animated:YES];
        [self showChannelSelection:self];
    }

    if ([notification.userInfo[AVAudioSessionRouteChangeReasonKey] unsignedIntegerValue] ==
        AVAudioSessionRouteChangeReasonOldDeviceUnavailable) {
        [availableInputPorts enumerateObjectsUsingBlock:^(AVAudioSessionPortDescription*
            inputPort, NSUInteger idx, BOOL *stop) {
            if ([inputPort.portType isEqualToString:AVAudioSessionPortBuiltInMic]) {
                NSError* error = nil;
                [session setPreferredInput:inputPort error:&error];
                if (error) {
                    [[UIApplication sharedApplication] showErrorMessage:NSLocalizedString(@"
                        Couldn’t set the default input microphone.", nil)];
                }

                *stop = YES;
            }
        }];
    }
}

#pragma mark -

@end
@implementation UIImage (WHBirdsViewControllerStyle)

+(UIImage*)birdsViewControllerInputImage {
    return [UIImage imageNamed:@"WHToolbar-input"];
}

@end
```

E.6 WHBirdViewController.m

```
//
//  WHBirdViewController.m
//  Whistles
//
//  Created by Laurin Brandner on 03.08.13.
//  Copyright (c) 2013 Laurin Brandner. All rights reserved.
//

#import "WHBirdViewController.h"
#import "WHWhistlesClient.h"
#import "WHBugViewController.h"

struct _WHBirdViewControllerGeometry WHBirdViewControllerGeometry = {
    .contentInsets = {15.0f, 15.0f, 15.0f, 15.0f},
    .imageViewSize = {80.0f, 80.0f},
    .playButtonSize = {80.0f, 44.0f},
    .titleLabelHeight = 21.0f,
    .textViewInset = 16.0f,
    .textViewOriginOffset = {-4.0f, -14.0f},
    .viewBorderWidth = 1.0f,
    .viewCornerRadius = 6.0f,
};

@interface WHBirdViewController ()

@property (nonatomic, strong) AVAudioPlayer* audioPlayer;

-(NSString*)titleForBird:(WHBird*)bird;

-(void)fadeAudioOut;

@end
@implementation WHBirdViewController

#pragma mark Accessors

-(void)setBird:(WHBird *)bird {
    if (![_bird isEqual:bird]) {
        _bird = bird;

        [self reloadBirdData];
    }
}

#pragma mark -
```

```
#pragma mark Initialization

-(id)initWithBird:(WHBird *)bird {
    self = [super init];
    if (self) {
        self.bird = bird;
    }

    return self;
}

#pragma mark -
#pragma mark View Lifecycle

-(void)loadView {
    self.view = [UIView new];
    self.view.autoresizingMask = UIViewAutoresizingFlexibleHeight|
       UIViewAutoresizingFlexibleWidth;

    self.scrollView = [UIScrollView new];
    self.scrollView.alwaysBounceVertical = YES;
    self.scrollView.backgroundColor = [UIColor whiteColor];
    [self.view addSubview:self.scrollView];

    CGColorRef borderColor = [UIColor colorWithWhite:0.1f alpha:0.2].CGColor;
    self.mapView = [MKMapView new];
    self.mapView.scrollEnabled = NO;
    self.mapView.layer.borderWidth = WHBirdViewControllerGeometry.viewBorderWidth;
    self.mapView.layer.borderColor = borderColor;
    [self.scrollView addSubview:self.mapView];

    self.imageView = [UIImageView new];
    self.imageView.contentMode = UIViewContentModeScaleAspectFill;
    self.imageView.image = [UIImage defaultBirdImage];
    self.imageView.layer.borderWidth = WHBirdViewControllerGeometry.viewBorderWidth;
    self.imageView.layer.borderColor = borderColor;
    self.imageView.layer.masksToBounds = YES;
    self.imageView.layer.cornerRadius = WHBirdViewControllerGeometry.viewCornerRadius;
    [self.scrollView addSubview:self.imageView];

    self.playButton = [UIButton new];
    [self.playButton addTarget:self action:@selector(toggleRecordPlayback:) forControlEvents
       :UIControlEventTouchUpInside];
    [self.playButton setTitle:NSLocalizedString(@"Play", nil) forState:UIControlStateNormal
       ];
    [self.playButton setTitle:NSLocalizedString(@"Pause", nil) forState:
       UIControlStateSelected];
    [self.playButton setTitleColor:[UIColor defaultControlColor] forState:
        UIControlStateNormal];
    [self.scrollView addSubview:self.playButton];

    self.titleLabel = [UILabel new];
    self.titleLabel.font = [UIFont boldSystemFontOfSize:17.0f];
    [self.scrollView addSubview:self.titleLabel];

    self.textView = [UITextView new];
    self.textView.textColor = [UIColor colorWithWhite:0.5f alpha:1.0f];
    self.textView.editable = NO;
    self.textView.scrollEnabled = NO;
    [self.scrollView addSubview:self.textView];

    [self reloadBirdData];
}

-(void)viewDidLoad {
    UIBarButtonItem* actionItem = [[UIBarButtonItem alloc] initWithBarButtonSystemItem:
        UIBarButtonSystemItemAction target:self action:@selector(showActions:)];
    self.navigationItem.rightBarButtonItem = actionItem;
}

-(void)viewDidLayoutSubviews {
    CGRect bounds = self.view.bounds;
    UIEdgeInsets contentInsets = WHBirdViewControllerGeometry.contentInsets;
    CGPoint origin = CGPointZero;

    self.scrollView.frame = bounds;

    CGFloat mapViewOffset = CGRectGetHeight(bounds);
    origin.y = -mapViewOffset;
    CGFloat visibleHeight = floorf(CGRectGetHeight(bounds)/4.0f);
    self.mapView.frame = (CGRect){origin, {CGRectGetWidth(bounds), floorf(visibleHeight+
        mapViewOffset)}};

    CGPoint point = [self.mapView convertCoordinate:self.bird.recordLocation toPointToView:
        self.mapView];
    point.y -= floorf(CGRectGetHeight(bounds)/2.0f);
    CLLocationCoordinate2D center = [self.mapView convertPoint:point toCoordinateFromView:
        self.mapView];
    self.mapView.centerCoordinate = center;

    origin = CGPointMake(contentInsets.left, CGRectGetMaxY(self.mapView.frame)+contentInsets
        .top);
    self.imageView.frame = (CGRect){origin, WHBirdViewControllerGeometry.imageViewSize};
    origin.y = CGRectGetMaxY(self.imageView.frame)+floorf(contentInsets.top/2.0f);
    self.playButton.frame = (CGRect){origin, WHBirdViewControllerGeometry.playButtonSize};

    origin = CGPointMake(CGRectGetMaxX(self.imageView.frame)+contentInsets.right,
        CGRectGetMinY(self.imageView.frame));
    self.titleLabel.frame = (CGRect){origin, {CGRectGetWidth(bounds)-origin.x-contentInsets.
        right, WHBirdViewControllerGeometry.titleLabelHeight}};

    origin.x += WHBirdViewControllerGeometry.textViewOriginOffset.x;
    origin.y = CGRectGetMaxY(self.titleLabel.frame)+contentInsets.top+
        WHBirdViewControllerGeometry.textViewOriginOffset.y;
    CGFloat textViewInset = WHBirdViewControllerGeometry.textViewInset;
    CGFloat textViewHeight = ceilf([self.textView.text sizeWithFont:self.textView.font
        constrainedToSize:CGSizeMake(CGRectGetWidth(self.titleLabel.frame)-textViewInset,
        CGFLOAT_MAX)].height+textViewInset);
    self.textView.frame = (CGRect){origin, {CGRectGetWidth(self.titleLabel.frame),
        textViewHeight}};

    CGSize contentSize = CGSizeZero;
    CGFloat neededHeight = CGRectGetMaxY(self.textView.frame)+contentInsets.bottom;
    if (neededHeight > CGRectGetHeight(bounds)) {
        contentSize.height = neededHeight;
    }
    self.scrollView.contentSize = contentSize;
}

-(void)viewWillDisappear:(BOOL)animated {
    [self fadeAudioOut];
}

#pragma mark -
#pragma mark Other Methods

-(NSString*)titleForBird:(WHBird *)bird {
    NSDateFormatter* formatter = [NSDateFormatter new];
    formatter.dateStyle = NSDateFormatterShortStyle;
    formatter.timeStyle = NSDateFormatterNoStyle;
    return [NSString stringWithFormat:@"Recording from %@", [formatter stringFromDate:bird.
        recordDate]];
}

-(void)reloadBirdData {
    self.title = [self titleForBird:self.bird];

    CLLocationCoordinate2D coordinate = self.bird.recordLocation;
    self.mapView.centerCoordinate = coordinate;
    [self.mapView removeAnnotations:self.mapView.annotations];
    MKPointAnnotation* annotation = [MKPointAnnotation new];
    annotation.coordinate = coordinate;
    [self.mapView addAnnotation:annotation];

    [self.imageView setImageWithURL:self.bird.pictureURL placeholderImage:[UIImage
        defaultBirdImage]];

    if (self.bird.identified) {
        NSString* title = [NSString stringWithFormat:@"%@ (%.1f%%)", self.bird.name, self.
            bird.accuracy];
        self.titleLabel.text = title;
    }
    else {
        self.titleLabel.text = NSLocalizedString(@"Unknown", nil);
    }
    self.textView.text = self.bird.text ?: NSLocalizedString(@"Whistles couldn’t identify
        the recorded bird. File a bug if you think Whistles is supposed to be able to know
        this bird.", nil);

    [self.view setNeedsLayout];
}

-(void)reidentifiy:(id)sender {
    [[WHWhistlesClient sharedClient] identifyRecording:self.bird.recordURL completion:^(
        NSDictionary* result, NSError* error) {
        if (error) {
            [[UIApplication sharedApplication] showErrorMessage:NSLocalizedString(@"Couldn’t
                 communicate with the server. Make sure your internet connection is working
                .", nil)];
        }

        [self.bird mapAttributes:result];
        [[NSManagedObjectContext sharedContext] save];
        [self reloadBirdData];

        NSString* message = [result objectForKey:@"message"];
        if (message) {
            [[UIApplication sharedApplication] showErrorMessage:NSLocalizedString(message,
                nil)];
        }
    }];
}

-(void)share:(id)sender {
    NSData* audio = [NSData dataWithContentsOfURL:self.bird.recordURL];
    NSString* text;
    if (self.bird.identified) {
        text = [NSString stringWithFormat:NSLocalizedString(@"I’ve found a %@. Have you seen
             one too?", nil), self.bird.name];
    }
    else {
        text = NSLocalizedString(@"Do you know what bird this is?", nil);
    }

    NSMutableArray* activityItems = [NSMutableArray arrayWithObjects:text, audio, self.bird.
        pictureURL, nil];

    UIActivityViewController* controller = [[UIActivityViewController alloc]
        initWithActivityItems:activityItems applicationActivities:nil];
    controller.excludedActivityTypes = @[UIActivityTypeAssignToContact, UIActivityTypePrint
        ];
    [self presentViewController:controller animated:YES completion:nil];
}

-(void)reportBug:(id)sender {
    WHBugViewController* bugController = [WHBugViewController bugViewControllerForRecording:
        self.bird];
    bugController.selectionEnabled = NO;
    UINavigationController* controller = [[UINavigationController alloc]
        initWithRootViewController:bugController];
    [self.navigationController presentViewController:controller animated:YES completion:nil
        ];
}

-(void)showActions:(id)sender {
    UIActionSheet* actionSheet = [[UIActionSheet alloc] initWithTitle:nil delegate:self
        cancelButtonTitle:NSLocalizedString(@"Cancel", nil) destructiveButtonTitle:nil
        otherButtonTitles:NSLocalizedString(@"Reidentify bird", nil), NSLocalizedString(@"
        Share", nil), NSLocalizedString(@"Report a bug", nil), nil];
    [actionSheet showFromBarButtonItem:sender animated:YES];
}

-(void)toggleRecordPlayback:(id)sender {
    if (self.playButton.selected) {
        [self stopRecording:sender];
    }
    else {
        [self playRecording:sender];
    }
}

-(void)playRecording:(id)sender {
    self.playButton.selected = YES;
    NSError* error = nil;
    self.audioPlayer = [[AVAudioPlayer alloc] initWithContentsOfURL:self.bird.recordURL
        error:&error];
    if (error) {
        [[UIApplication sharedApplication] showErrorMessage:NSLocalizedString(@"An error
            occured while playing the audio.", nil)];
        self.audioPlayer = nil;
        return;
    }

    self.audioPlayer.delegate = self;
    [self.audioPlayer play];
}

-(void)stopRecording:(id)sender {
    self.playButton.selected = NO;
    [self.audioPlayer stop];
    self.audioPlayer = nil;
}

-(void)fadeAudioOut {
    if (!self.audioPlayer) {
        return;
    }

    if (self.audioPlayer.volume > 0.1f) {
        self.audioPlayer.volume = self.audioPlayer.volume-0.1f;
        [self performSelector:@selector(fadeAudioOut) withObject:nil afterDelay:0.05];
    } else {
        [self stopRecording:self];
    }
}

#pragma mark -
#pragma mark UIActionSheetDelegate

-(void)actionSheet:(UIActionSheet *)actionSheet didDismissWithButtonIndex:(NSInteger)
    buttonIndex {
    if (buttonIndex != actionSheet.cancelButtonIndex) {
        if (buttonIndex == 0) {
            [self reidentifiy:self];
        }
        else if (buttonIndex == 1) {
            [self share:self];
        }
        else {
            [self reportBug:self];
        }
    }
}

#pragma mark -
#pragma mark AVAudioPlayerDelegate

-(void)audioPlayerDidFinishPlaying:(AVAudioPlayer *)player successfully:(BOOL)flag {
    self.playButton.selected = NO;
}

#pragma mark -

@end
@implementation UIImage (WHBirdViewControllerStyle)

+(UIImage*)defaultBirdImage {
    // Copyright Dragoart. This is property of Dragoart. http://www.dragoart.com/tuts
       /3312/1/1/how-to-draw-a-robin.htm

    return [UIImage imageNamed:@"WHBirdViewController-bird"];
}

@end
```

E.7 WHBugViewController.m

```
//
//  WHBugViewController.m
//  Whistles
//
//  Created by Laurin Brandner on 09.08.13.
//  Copyright (c) 2013 Laurin Brandner. All rights reserved.
//

#import "WHBugViewController.h"
#import "WHWhistlesClient.h"

struct _WHBugViewControllerGeometry WHBugViewControllerGeometry = {
    .contentInsets = {15.0f, 15.0f, 15.0f, 15.0f},
    .selectionButtonHeight = 50.0f,
    .titleLabelHeight = 21.0f
};

@interface WHBugViewController ()

@property (nonatomic, strong) NSFetchedResultsController* controller;
@property (nonatomic, strong) WHPickerView* pickerView;

@property (nonatomic) CGFloat keyboardHeight;

-(NSString*)titleForRecording:(WHBird*)bird addBirdNameIfKnown:(BOOL)addBirdName;

-(void)loadSelectionButton;

-(void)keyboardWillChangeFrame:(NSNotification*)notification;
-(void)layoutTextViewWithOrigin:(CGPoint)origin animated:(BOOL)animated;

@end
@implementation WHBugViewController

#pragma mark Accessor

-(void)setRecording:(WHBird *)recording {
    if (![_recording isEqual:recording]) {
        _recording = recording;

        self.title = [self titleForRecording:recording addBirdNameIfKnown:NO];
    }
}

-(void)setSelectionEnabled:(BOOL)selectionEnabled {
    if (_selectionEnabled != selectionEnabled) {
        _selectionEnabled = selectionEnabled;

        if (self.isViewLoaded) {
            if (_selectionEnabled) {
                [self loadSelectionButton];
            }
            else {
                [self.selectionButton removeFromSuperview];
                self.selectionButton = nil;
            }
        }
    }
}

-(NSFetchedResultsController*)controller {
    if (_controller) {
        return _controller;
    }

    NSFetchRequest* request = [NSFetchRequest fetchRequestWithEntityName:NSStringFromClass([
        WHBird class])];
    request.sortDescriptors = @[[NSSortDescriptor sortDescriptorWithKey:@"recordTimestamp"
        ascending:NO]];
    self.controller = [[NSFetchedResultsController alloc] initWithFetchRequest:request
        managedObjectContext:[NSManagedObjectContext sharedContext] sectionNameKeyPath:nil
        cacheName:nil];
    _controller.delegate = self;
    [_controller performFetch:nil];

    return _controller;
}

#pragma mark -
#pragma mark Initialization

+(instancetype)bugViewControllerForRecording:(WHBird *)recording {
    WHBugViewController* controller = [WHBugViewController new];
    controller.recording = recording;

    return controller;
}

-(id)init {
    self = [super init];
    if (self) {
        WHBugViewControllerInitialize(self);
    }

    return self;
}

-(id)initWithCoder:(NSCoder *)aDecoder {
    self = [super initWithCoder:aDecoder];
    if (self) {
        WHBugViewControllerInitialize(self);
    }

    return self;
}

void WHBugViewControllerInitialize(WHBugViewController* controller) {
    controller.title = [controller titleForRecording:controller.recording addBirdNameIfKnown
        :NO];
    controller.selectionEnabled = YES;

    [[NSNotificationCenter defaultCenter] addObserver:controller selector:@selector(
        keyboardWillChangeFrame:) name:UIKeyboardWillChangeFrameNotification object:nil];
}

-(void)dealloc {
    [[NSNotificationCenter defaultCenter] removeObserver:self];
}

#pragma mark -
#pragma mark View Lifecycle

-(void)loadView {
    self.view = [UIView new];
    self.view.autoresizingMask = UIViewAutoresizingFlexibleHeight|
        UIViewAutoresizingFlexibleWidth;

    self.scrollView = [UIScrollView new];
    self.scrollView.alwaysBounceVertical = YES;
    self.scrollView.backgroundColor = [UIColor whiteColor];
    [self.view addSubview:self.scrollView];

    if (self.selectionEnabled) {
        [self loadSelectionButton];
    }

    self.titleLabel = [UILabel new];
    self.titleLabel.font = [UIFont boldSystemFontOfSize:17.0f];
    self.titleLabel.text = NSLocalizedString(@"Description", nil);
    [self.scrollView addSubview:self.titleLabel];

    UIColor* placeholderColor = [UIColor textViewPlaceholderColor];
    self.textView = [WHTextView new];
    self.textView.placeholder = NSLocalizedString(@"Explain what went wrong with the
        identification and the result you’ve expected. Make sure to mention any special
        circumstances if any.", nil);
    self.textView.layer.borderWidth = 1.0f;
    self.textView.layer.borderColor = [placeholderColor colorWithAlphaComponent:0.2f].
        CGColor;
    self.textView.placeholderTextColor = placeholderColor;
    self.textView.font = [UIFont systemFontOfSize:13.0f];
    self.textView.delegate = self;
    [self.scrollView addSubview:self.textView];
}

-(void)loadSelectionButton {
    if (self.selectionButton) {
        return;
    }

    self.selectionButton = [UIButton new];
    [self.selectionButton setTitle:NSLocalizedString(@"Select the recording", nil) forState:
        UIControlStateNormal];
    [self.selectionButton addTarget:self action:@selector(selectRecording:) forControlEvents
        :UIControlEventTouchUpInside];
    [self.selectionButton setTitleColor:[UIColor defaultControlColor] forState:
        UIControlStateNormal];
    [self.scrollView addSubview:self.selectionButton];
}

-(void)viewDidLoad {
    UIBarButtonItem* sendItem = [[UIBarButtonItem alloc] initWithTitle:NSLocalizedString(@"
        Report", nil) style:UIBarButtonItemStylePlain target:self action:@selector(
        sendReport:)];
    self.navigationItem.rightBarButtonItem = sendItem;

    UIBarButtonItem* cancelItem = [[UIBarButtonItem alloc] initWithTitle:NSLocalizedString(@
        "Cancel", nil) style:UIBarButtonItemStylePlain target:self action:@selector(cancel:)
        ];
    self.navigationItem.leftBarButtonItem = cancelItem;
}

#pragma mark -
#pragma mark Other Methods

-(NSString*)titleForRecording:(WHBird *)bird addBirdNameIfKnown:(BOOL)addBirdName {
    if (bird) {
        NSDateFormatter* formatter = [NSDateFormatter new];
        formatter.dateStyle = NSDateFormatterShortStyle;
        formatter.timeStyle = NSDateFormatterNoStyle;
        if (addBirdName && bird.identified) {
            return [NSString stringWithFormat:@"%@ recorded on %@", bird.name, [formatter
                stringFromDate:bird.recordDate]];
        }
        return [NSString stringWithFormat:@"Recording from %@", [formatter stringFromDate:
            bird.recordDate]];
    }

    return NSLocalizedString(@"Bug Report", nil);
}

-(void)selectRecording:(id)sender {
    [self showPickerView:YES];
}

-(void)cancel:(id)sender {
    [self.presentingViewController dismissViewControllerAnimated:YES completion:nil];
}

-(void)sendReport:(id)sender {
    [[WHWhistlesClient sharedClient] reportBugForRecording:self.recording.recordURL
        description:self.textView.text completion:^(NSError* error) {
        if (error) {
            [[UIApplication sharedApplication] showErrorMessage:NSLocalizedString(@"An error
                occured while reporting the bug. Please try again later.", nil)];
        }
    }];
    [self cancel:sender];
}

-(void)showPickerView:(BOOL)animated {
    [self.textView endEditing:YES];

    if (!self.pickerView) {
        self.pickerView = [[WHPickerView alloc] initWithTitle:@"Recordings"];
        self.pickerView.delegate = self;
        if (self.recording) {
            self.pickerView.selectedRow = [self.controller indexPathForObject:self.recording
                ].row;
        }

        [self.pickerView showInView:self.view animated:YES];
    }
```

```
}

-(void)keyboardWillChangeFrame:(NSNotification *)notification {
    self.keyboardHeight = CGRectGetHeight(self.view.bounds)-CGRectGetMinY(((NSValue*)
        notification.userInfo[UIKeyboardFrameEndUserInfoKey]).CGRectValue);
    [self layoutTextViewWithOrigin:self.textView.frame.origin animated:YES];
}

#pragma mark -
#pragma mark WHPickerViewDelegate

-(NSUInteger)numberOfTitlesInPickerView:(WHPickerView *)pickerView {
    id <NSFetchedResultsSectionInfo> sectionInfo = [self.controller.sections objectAtIndex
        :0];
    return [sectionInfo numberOfObjects];
}

-(NSString*)pickerView:(WHPickerView *)pickerView titleForRow:(NSInteger)row {
    NSIndexPath* indexPath = [NSIndexPath indexPathForRow:row inSection:0];
    WHBird* bird = [self.controller objectAtIndexPath:indexPath];

    return [self titleForRecording:bird addBirdNameIfKnown:YES];
}

-(void)pickerView:(WHPickerView *)pickerView didSelectRow:(NSUInteger)row {
    NSIndexPath* indexPath = [NSIndexPath indexPathForRow:row inSection:0];
    self.recording = [self.controller objectAtIndexPath:indexPath];
}

-(void)pickerViewDidDismiss:(WHPickerView *)pickerView {
    [self.pickerView hide:YES];
    self.pickerView = nil;
}

#pragma mark -
#pragma mark UITextViewDelegate

-(void)textViewDidBeginEditing:(UITextView *)textView {
    [self.pickerView hide:YES];
}

-(void)textViewDidChange:(UITextView *)textView {
    [self.textView scrollRangeToVisible:NSMakeRange(self.textView.text.length, 0)];
}

#pragma mark -
#pragma mark Layout
```

```
-(void)viewDidLayoutSubviews {
    UIEdgeInsets insets = WHBugViewControllerGeometry.contentInsets;
    CGRect bounds = self.view.bounds;

    self.scrollView.frame = bounds;

    bounds = UIEdgeInsetsInsetRect(bounds, insets);
    CGFloat width = CGRectGetWidth(bounds);

    CGPoint origin = CGPointMake(CGRectGetMinX(bounds), CGRectGetMinY(bounds));
    self.selectionButton.frame = (CGRect){origin, {width, WHBugViewControllerGeometry.
        selectionButtonHeight}};

    origin.y += CGRectGetMaxY(self.selectionButton.frame);

    self.titleLabel.frame = (CGRect){origin, {width, WHBugViewControllerGeometry.
        titleLabelHeight}};

    origin.y = CGRectGetMaxY(self.titleLabel.frame)+insets.top;
    [self layoutTextViewWithOrigin:origin animated:NO];
}

-(void)layoutTextViewWithOrigin:(CGPoint)origin animated:(BOOL)animated {
    CGRect bounds = UIEdgeInsetsInsetRect(self.view.bounds, WHBugViewControllerGeometry.
        contentInsets);
    CGFloat textViewHeight = CGRectGetHeight(bounds)-origin.y-self.scrollView.contentInset.
        top-self.scrollView.contentInset.bottom-self.keyboardHeight;
    [UIView animateWithDuration:0.2f animations:^{
        self.textView.frame = (CGRect){origin, {CGRectGetWidth(bounds), textViewHeight}};
    }];
}

#pragma mark -

@end
@implementation UIColor (WHBugViewControllerStyle)

+(UIColor*)textViewPlaceholderColor {
    return [UIColor colorWithWhite:0.1f alpha:0.4];
}

@end
```

E.8 application_controller.rb

```
require "base64"

class ApplicationController < ActionController::Base
  protect_from_forgery with: :null_session

  # POST bug.json
  def report_bug
    dir = "#{Rails.root}/reports/#{Time.now.to_i}"

    unless File.directory? dir
      FileUtils.mkdir_p dir
    end

    desc_dir = dir + "/description.txt"
    File.open(desc_dir, "w") do |file|
      file.write report_params[:description]
    end

    audio_data = Base64.decode64(report_params[:audio])
    audio_dir = dir + "/audio.caf"
    File.open(audio_dir, "wb") do |file|
      file.write audio_data
    end

    response = {status: 200, message:"Bug successfully reported."}
    respond_to do |format|
      format.xml { render :xml => response }
      format.any { render :json => response }
    end
  end

  private

    def report_params
      params.permit(:description, :audio)
    end

end
```

E.9 birds_controller.rb

```
require "fingerprint"

class BirdsController < ApplicationController
  skip_before_filter :verify_authenticity_token

  # GET /birds/all.json
  def all
    birds = Bird.all
    respond_to do |format|
      format.xml { render :xml => birds }
      format.any { render :json => birds }
    end
  end

  # POST /birds/new.json
  def new
    bird = Bird.new
    bird.name = bird_params[:name]
    bird.fingerprint = bird_params[:fingerprint]

    respond_to do |format|
      if bird.save
        format.xml { render xml: bird, status: :created }
        format.any { render json: bird, status: :created }
      else
        format.xml { render xml: bird.errors, status: :unprocessable_entity }
        format.any { render json: bird.errors, status: :unprocessable_entity }
      end
    end
  end

  # PUT /birds/update.json
  def update
    bird = Bird.where("name = ?", bird_params[:name]).first

    if (bird.kind_of? Array) then
      raise ActionController::RoutingError.new("Internal Error")
    else
      bird.fingerprint = bird_params[:fingerprint]

      respond_to do |format|
        if bird.save
          format.xml { render xml: bird, status: :created }
          format.any { render json: bird, status: :created }
        else
```

```
          format.xml { render xml: bird.errors, status: :unprocessable_entity }
          format.any { render json: bird.errors, status: :unprocessable_entity }
        end
      end
    end
  end

  # GET /birds/identify.json
  def identify
    fingerprint = Fingerprint.new params[:fingerprint]
    if (fingerprint.nil? || !fingerprint.valid?) then
      raise ActionController::RoutingError.new("Internal Error")
    else
      bird, percentage = Bird.identify fingerprint
      response;
      if bird.nil? then
        response = {status: 200, message: "Couldn’t find a matching bird."}
      else
        response = ret_params(bird, percentage)
      end

      respond_to do |format|
        format.xml { render :xml => response }
        format.any { render :json => response }
      end
    end
  end

  private

    def bird_params
      params.permit(:name, :fingerprint)
    end

    # The description attribute of the birds is a paragraph of their corresponding wikipedia
        website. It only serves as a proof of concept.

    def ret_params(bird, accuracy)
      domain = "#{request.protocol}#{request.host_with_port}"
      return {name:bird.name, description:bird.description, accuracy:accuracy, image:bird.
          image_url_with_domain(domain)}
    end

end
```

E.10 bird.rb

```
require "fingerprint"

class Bird < ActiveRecord::Base

  validates :name, presence:true, uniqueness:true
  validates :fingerprint, presence:true

# Accessors

  def fingerprint
    Fingerprint.new read_attribute(:fingerprint)
  end

# Class Methods

  def self.identify (fingerprint1)
    birds = Bird.all

    matches = Hash.new

    birds.each do |bird|
      fingerprint2 = bird.fingerprint
      matches[bird] = fingerprint2.similarity_to fingerprint1
    end

    # puts "MATCHES"
    # matches.each_pair do |key, value|
    #   puts "#{key.name}:#{value}"
    # end

    bird = nil
    match = 0

    matches.each_pair do |key, value|
      if (value > match) then
        match = value
        bird = key
      end
    end

    return bird, match*100.0
  end

# Other Methods
```

```
  def to_s
    self.name
  end

  def image_url_with_domain domain
    file_name = self.name.downcase.sub(" ", "_")
    "#{domain}/bird/#{file_name}.jpg"
  end

end
```

E.11 fingerprint.rb

```
class Fingerprint

# Accessors

        attr_accessor :subfingerprints

# Initialization

        def initialize(data)
                if (data.nil? || data.empty?) then
                        return nil
                end
                @subfingerprints = data.split "+"
        end

# Comparison

        def similarity_to(fingerprint)
                match = 0.0
                offset = 0

                while (offset < @subfingerprints.count-fingerprint.subfingerprints.count) do
                        matchesSum = 0.0

                        fingerprint.subfingerprints.each_with_index do |subfingerprint2,
                            index|
                                subfingerprint1 = @subfingerprints[index]
                                current_match = self.compare_subfingerprints(subfingerprint1
                                    , subfingerprint2)

                                matchesSum += current_match
                        end

                        average_match = matchesSum/fingerprint.subfingerprints.count
                        if average_match > match then
                                match = average_match
                        end

                        offset += 1
                end

                return match
        end

        def compare_subfingerprints(subfingerprint1, subfingerprint2)
```

```
                possible_hits = 0
                hits = 0

                for i in (0..subfingerprint1.length).step(2)
                        on = "1"
                        sf1s1 = subfingerprint1[i]
                        sf1s2 = subfingerprint1[i+1]

                        if (sf1s1 == on || sf1s2 == on) then
                                        possible_hits += 1

                                        sf2s1 = subfingerprint2[i]
                                        sf2s2 = subfingerprint2[i+1]

                                        if ((sf1s1 == sf2s1) && (sf1s2 == sf2s2)) then
                                                        hits += 1
                                        end
                        end
                end

                if (possible_hits <= 0.0) then
                                return 0.0
                end

                return hits.to_f/possible_hits.to_f
        end

        def ==(fingerprint)
                return (@subfingerprints == fingerprint.subfingerprints)
        end

        def ===(fingerprint)
                return (self == fingerprint)
        end

# Other Methods

        def valid?
                @subfingerprints.count > 0
        end

end
```

F Anti-Plagiat-Erklärung

Ich erkläre hiermit, dass

- diese Arbeit weder abgeschrieben noch kopiert oder aus dem Internet übernommen wurde.
- der Quellennachweis korrekt und vollständig ist.
- die dargestellten Daten und Resultate selber und korrekt erhoben und verarbeitet wurden.

Name:	Ort:	Datum:	Unterschrift: