Saarland University
Faculty of Natural Sciences and Technology I
Department of Computer Science

Bachelor's Thesis

Short-time Audio Event Classification with Applications to the Evaluation of Sport Experiments

submitted by
Mark Simkin

submitted
September 24, 2012

Supervisor / Advisor
Prof. Dr. Meinard Müller

Reviewers
Prof. Dr. Meinard Müller
Prof. Dr. Michael Clausen

Eidesstattliche Erklärung

Ich erkläre hiermit an Eides Statt, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Statement under Oath

I confirm under oath that I have written this thesis on my own and that I have not used any other media or materials than the ones referred to in this thesis.

Einverständniserklärung

Ich bin damit einverstanden, dass meine (bestandene) Arbeit in beiden Versionen in die Bibliothek der Informatik aufgenommen und damit veröffentlicht wird.

Declaration of Consent

I agree to make both versions of my thesis (with a passing grade) accessible to the public by having them added to the library of the Computer Science Department.

Saarbrücken, _____ _____

(Datum / Date) (Unterschrift / Signature)

Acknowledgements

I would like to thank Priv.-Doz. Meinard Müller for providing me with this interesting topic as well as all the experience that came along with it. Furthermore I would like to thank Peter Grosche for being supportive throughout, helping me with all kinds of problems. Also I would also like to express my gratefulness to Thomas Helten. His expertise and knowledge on how to conduct and prepare the experiments data was of great value.

In general I would like to thank the whole group for providing such a nice working atmosphere. Furthermore there is the Sport Psychology Unit of the Paderborn University without which we could have not conducted the experiments in the first place.

Last but not least I would like to thank my family and everybody else who supported me during this time.

Contents

Abstract

After conducting some experiment one always faces the tedious work of manually annotating and examining it. This is time consuming and error prone, especially if the experiment contains a large amount of relevant events that need to be annotated. In this thesis we tackle this problem for a real world experiment from the field of sport psychology, where two players have to pass a ball to each other as quickly as possible by bouncing it over a target zone. The experiments evaluation consists of the number of passes that bounced on the target zone in relation to the number of bounces that did not. Our main idea was to cover the target zone with a surface different from the one outside of it and use audio classification techniques to classify the bounces. To obtain the required audio data we made minor changes to the experiments setup by adding two microphones to it and changing the surface of the target zone. We conducted a number of experiments with this setup, extracted and analyzed all bounces from the audio data we retrieved and showed, that an automatic classification is very possible and feasible.

Chapter 1

Introduction

In our everyday life we perceive and process a multitude of different sounds without much effort. When listening we can have very different goals in mind. Sometimes we just want to understand the words spoken to us by somebody else. On another occasion we want to identify a piece of music by only listening to a small extract of it. When fulfilling these tasks, we extract various properties from the processed sound which we intuitively consider to be more significant for a certain goal. As an example we might pay close attention to the rhythm of a sound when trying to recognize a song, but we would not pay any attention to this property when somebody is talking to us.

The task of *audio classification* is to automatically extract and analyze such audio properties from a given audio recording in order to fulfill goals similar to the ones mentioned above. To successfully tackle such tasks we have to extract the most characteristic audio properties, so called *audio features*, for each of them. When, for example, analyzing the *danceability* of a given song, we might consider rhythmic properties such as tempo and beat to be important.

In this thesis we will deal with a typical classifciation problem, where our algorithm has to assign labels to certain events in the audio recording from a given set of labels. Typically such a problem is tackled in a two-stage approach. In the first stage we extract more meaningful information from each event in the given audio recording by reducing it to a vector of different features, a so called *feature vector*. This mainly involves techniques known from signal processing. In the next stage we then analyze each feature vector with methods known from machine learning in order to decide how the different features should be weighted and combined. Ideally we want to minimize the difference between feature vectors that belong to different events of the same label, while maximizing this difference for events belonging to different labels.

In this thesis we will analyze and compare a set audio features in regards to their usability for short time audio events based on the task of evaluating an experiment from the field of sports psychology. These short time audio events, which occur in the experiment, are usually shorter than a second and thus we are somewhat limited in the set of properties we can use to analyze them. We will mainly rely on timbral properties rather than on melodic or temporal information.

The experiment consists of two players passing a ball to each other by bouncing it on a target zone of variable size as fast as possible in a given time interval. The purpose of it is to backup the so called *Interpersonal Fitts' law*, which formulates the time needed to reach a target zone as a function of the distance to and the size of the target zone. We will describe this law in more detail in a later chapter. The evaluation of such an experiment consists of the number of passes that bounce on the target zone. As we will see later there are about 135 passes on average in each experiment. It would be very tedious to annotate each pass for each experiment by hand and therefore we will be using audio classification techniques to ease our lives. Here our main idea is that different surfaces sound different. We will coat the target zone with a surface different from the one surrounding it in order to hear when a bounce hits the target zone. The experiments execution will then be recorded with a microphone in order to obtain an audio recording, which we can analyze with our classification algorithm. Audio classification has several advantages here. Recording a video requires a very good and expensive camera, due to the high speed of the ball as we might not be able to see where the ball hits the ground otherwise. In addition one camera might even not be enough to classify a bounce with certainty in our concrete experimental setup. For a human, this task is not only very tedious, but also slow and error prone, thus rendering this solution infeasible for larger sets of experimental recordings.

With audio classification, on the other hand, we completely avoid those problems. An average microphone is cheap, easy to acquire and still suffices for our purposes. Moreover we simplify the experimental setup when using a microphone rather than a set of cameras. In case the algorithm is not able to classify a bounce with certainty, we can still fall back on a manual annotation by a human, resulting in a semi-automatic classification approach. For this purpose we developed a small tool to simplify the manual annotation which we will describe in a later chapter. As already mentioned, we assume that bounces on the target zone sound different from bounces outside of it. We conducted the experiment in a gym and by covering the target zone with a carpet we obtained two different sounding surfaces.

Strictly speaking our algorithm consists of two pipelines. One for the learning phase and one for the classification phase. In the learning phase the algorithm receives already labelled training data as its input and outputs a classifier, that is then used in the second pipeline to classify unlabelled data. These two pipelines are mostly similar as they only differ in their last step. In Figure 1.1 we can see the classification pipeline. First the algorithm receives an audio recording of an experiment execution as a waveform, depicted in blue. We want to classify each bounce with our classifier on its own, thus we first need to detect the relevant events in the audio recording. After detecting the events we extract each bounce by cutting out a 0.5 seconds long window containing each of them. These extracts are shown as grey boxes in the Figure. This is done in the *event detection* step. Now we iterate over the set of obtained waveform extracts and analyze each of them by itself. For each extract we compute our choosen audio features and build a feature vector in the *feature extraction* step. By reducing each extract to a feature vector we accentuate the meaningful information while ignoring properties that we consider to not be important for our purpose. In the last step, the so called *classification* step, we use our classifier to determine which label is most likely for each bounce based on its feature vector. The classifier uses a *distance function* to calculate the distance of a feature vector

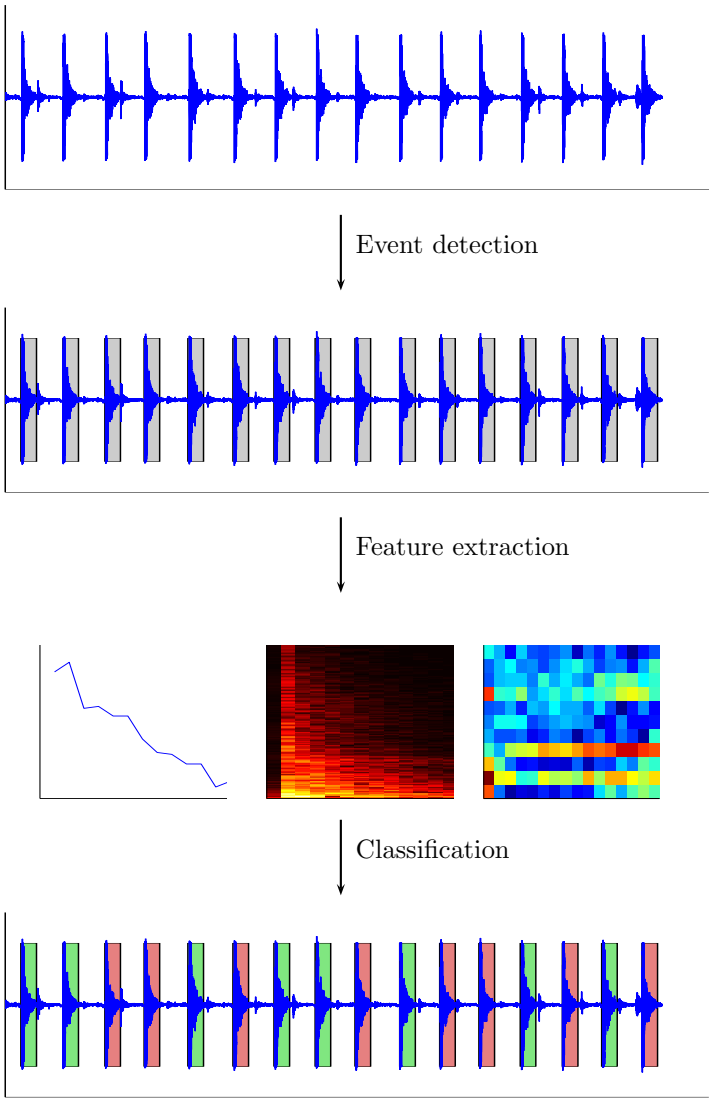

Figure 1.1. Outline of the classification algorithm showing all major processing steps from top to bottom that are needed to classify bounces from a given waveform.

to a template of a feature vector for a certain class. After doing this for each class the classifier outputs the label of the class that has minimal distance to the given feature vector. We will investigate distance functions more closely in Chapter 4.

The only difference to the learning pipeline is that there we would learn the classifier in the last step rather than classifying.

Thesis Outline

The content of this thesis is organized as follows:

Chapter 2 describes Fitts' law, the experimental setup as well as the resulting dataset.

Chapter 3 introduces basic concepts and notions from the field of signal processing. Further we will examine the choosen features for this audio classification task.

Chapter 4 presents several classification methods for unlabelled audio events

Chapter 5 exhibits an evaluation of the techniques we used to accomplish our classification task.

Chapter 6 concludes the thesis with a summary of our results.

Chapter 2

Experimental Setup

Ever since Fitts' law was proposed by Paul Fitts in 1954 it established itself as a cornerstone for evaluating human movements in regards to the speed they are fulfilled with. In this chapter we will deal with this law as well as the experimental setup that was used to investigate its soundness for interpersonal human movements. We will start by examining *Fitts' law* in Section 2.1 in more detail. Afterwards we will move on to the experimental setup that was used to conduct the experiment in Section 2.2.

After conducting the experiment we acquired a large amount of raw data which had to be edited and preprocessed. Details on this can be found in Section 2.3 and Section 2.4.

2.1 Fitts' Law

In 1954 Paul Fitts was researching the human motor system and its limiting factors. His aim was to find a model for human movement that could predict the speed to fulfill a given movement towards a certain target based on the difficulty of the movement. Having such a model could give further insights into a variety of different areas such as *human-computer interaction*, ergonomics or sports science. For instance, both a soccer player kicking and a basketball player throwing a ball are trying to fulfill such a movement and Fitts' Law can give us indications about their performances. Especially for the rather new field of human-computer interaction this law is very important as the majority of this kind of interaction require pointing or dragging towards a certain target zone, like dragging files from one window to another or clicking a button on an interface. Fitts' law was a rather novel approach at its time as it introduced the information metric bits to measure human movements, thus it enabled us to measure, evaluate and compare such movements and the devices that require the movement in a mathematically sound way.

To make all these movements comparable in regards to their difficulty Fitts defined

$$I_{\mathrm{D}} := \log_2(2 \cdot \frac{A}{W}) \tag{2.1}$$

to be the *index of difficulty* where A is the amplitude, i. e. the distance to the target and W its width. This formula is derived from the *Shannon-Hartley theorem*, which is defined

as follows.

Theorem 2.1 *Let a channel be given with bandwidth B, signal power S and noise N, the upper bound for the transmission rate is defined as*

$$C := B \cdot \log_2(1 + \frac{S}{N}) \tag{2.2}$$

Apparently this theorem cannot directly be applied to human movements, because we are lacking a definition of capacity, channels, bandwidth and noise for human beings. Therefore Fitts introduces the *index of performance* defined as $I_\mathrm{P} := I_\mathrm{D}/MT$ with MT being the movement time to complete a given task. Given the movement time MT in seconds, I_P and the Shannon-Hartley theorem correspond directly with MT being $1/B$. The interpretation for this analogy is that electronic signals can be seen as movement distances and the noise of a channel as the tolerance, i. e. the width of the target zone. Subsequently Fitts' Law was formulated as

$$t = a + b \cdot \log_2(\frac{2A}{W}) \tag{2.3}$$

which describes a regression line with the index of difficulty on the x-axis and the movement time on the y-axis.

The above version of Fitts' law only applies to simple movement tasks with one hand. Consequently it seemed natural to try to extend this law to movement tasks that involve more than one hand. Experiments showed that our brain accesses functional groups of muscles rather than each motor unit individually. Motivated by these results the following experiment is now investigating movement tasks that involve two people to see what kind of dynamic interaction might exist between them. Further details on Fitts' law and its variations can be found at [1, 3, 2].

2.2 Experimental Setup

Each experiment consisted of two players and nine different *scenes*, where a scene denotes a distance-target size combination. In Figure 2.1 we can see a top down view of experiments setup. The light brown square with side length W indicates the target zone. The players stand on the left, respectively right side of the target zone facing each other so that their finger tips have a distance of A from the center of the target zone if they strech out their arms in front of them. Next to the target zone two microphones are placed symmetrically indicated as a circle with the label *mic*. Additionally a camera, indicated as a circle with the label *cam*, is placed next to the target zone to facilitate manual annotation in case the algorithm cannot classify a given bounce with certainty. The target size W and the distance A, both shown as dashed lines, are variables with three possible values each, resulting in nine possible scenes, hence resulting in nine different $I_\mathrm{D}s$.

In our setup we covered the blue area with a carpet, while leaving the target zone to be the underlying gym floor. The blue area, which indicates a miss, was chosen sufficiently large in order to render bounces outside of it highly unlikely. In the rare event of a bounce outside the miss zone, that scene had to be repeated.

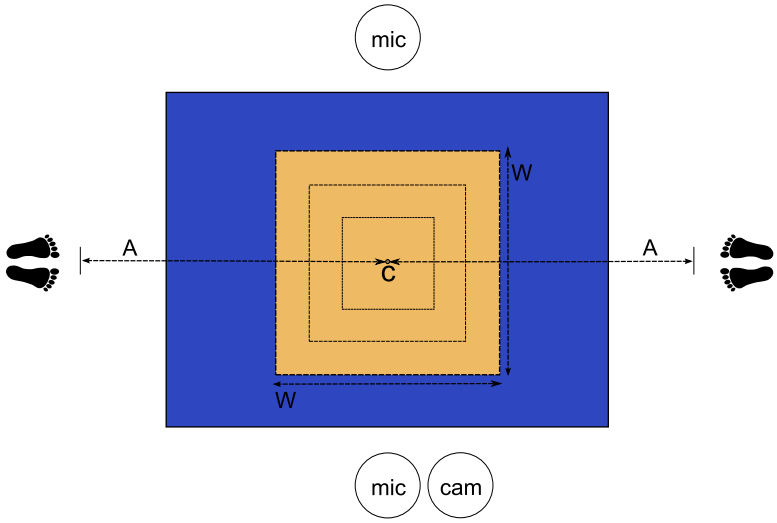

Figure 2.1. Sketch of experimental setup. Players stand on the left, respectively right side of the light brown target zone with a variable distance of A between the target zone and the finger tips of the players streched out hands. The blue surrounding of the target zone indicates the miss zone.

For each of these scenes the players were given a 20 seconds time window in which they had to pass a basketball to each other by bouncing it on the target zone as quick as possible. We will call one execution of such a 20 second window for a certain scene a *take*. The players were instructed to maximize the amount of passes that bounce over the target zone.

Apart from nine scenes mentioned above six other scenes were used. These were used for training the algorithm. Three scenes did not contain any miss zone and the other three did not contain any target zone. This way we could obtain recordings that only contained bounces on one surface.

Each of the training scenes was conducted twice in our experiment. After finishing the training phase the actual experiment was executed. Here each one of the nine scenes was executed four times in a row before moving on to the next one.

In Figure 2.2 we can see a photograph of an experiment execution. On the left and right we see the players bouncing the ball towards each other. The person in the middle of the picture keeps track of the hits/misses count for each execution in order to facilitate a later evaluation of the algorithms results. Between the players we can see the target surrounded by the blue miss zone. Next to it the two microphones and the camera are mounted on their stands as described in Figure 2.1.

2.3 Dataset and Annotations

In cooperation with the sports psychology unit of the Paderborn University the experiment was conducted with nine different pairs of people. Audio and video were recorded for each one of these experiments. The audio and video footage was annotated and then cut into 21 seconds long parts, each containing one take of an experimental scene, padded with half a second before and half a second after execution. As a result we obtained about 36 minutes of video footage for each pair. Since we obtained two additional audio recordings for each training scene for every pair, we have 42 minutes of audio recordings per pair, thus making it a total of 6 hours and 18 minutes of audio recordings that had to be processed. As mentioned above, we kept track of the number of hits and misses for each execution. Furthermore one annotation file per pair was created manually by looking through the videos and listening to the audio recordings, containing one bitstring for each take of each scene. These bitstrings describe the class of each bounce in one take to facilitate the evaluation of the algorithms output. This way we could obtain more precise insights into when and why the algorithm failed when it did.

Figure 2.2. Photograph of the experiment during its execution.

2.4 User interface for efficient manual annotation

Annotating these experiments by hand is very tedious and thus we developed a simple tool to simplify this process. A screenshot of it can be seen in Figure 2.3. Given a audio and video recording of a scene this tool detects the onsets as well as the takes inside the scene. Each detected onset is matched with its video extract inside the video recording and can be viewed by clicking onto the according rectangle. When manually annotating, one can now just click through the detected onsets one by one and view their video extracts instead of watching the whole video recording. Often it is neccessary to review a certain bounce multiple times to classify it correctly. With this tool this is possible in a convenient way and through the waveform representation one can easily navigate back and forth through the experiment without being unsure about what onset one is looking at, as it would be the case when watching the video itself.

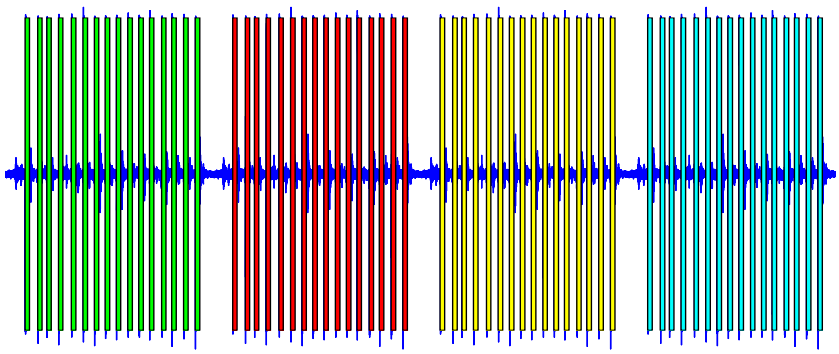

Figure 2.3. Annotation interface for manual annotation showing the recorded waveform for one scene. Each rectangle represents a detected onsets and by clicking it one can see the desired video extract for that detected onset. Each take of this scene is recognized and highlighed in a separate color.

Chapter 3

Feature Extraction

In this chapter we will discuss the first steps of our classification algorithm from figure 1.1, namely the *onset detection* and the *feature extraction*. Choosing the right features when extracting meaningful information from an audio signal is essential as these features have to represent information that varies significantly between classses, while varying only slightly between different audio signals belonging to the same class. The transformation of an audio signal into a more meaningful feature vector is called feature extraction. As we will see in this chapter sound can be decomposed into a spectrum of simple oscillating waves with different frequencies. This insight builds the foundation for many of the techniques used in audio signal processing as many sounds that appear to be similar to the human ear often also have visible similarities in frequency spectrum based representations.

We begin with introducing notions from signal processing and basic mathematical concepts that are necessary for understanding and dealing with audio features. Next we will look at time domain based features in Section 3.3, where we extract meaningful information directly from the audio signal. From Section 3.4 onwards we start looking at more complex frequency domain based features.

3.1 Basic notions and definitions

Before discussing audio signals we should first of all define what exactly we mean by that. In physics sound is described as a oscillating wave of pressures propagating through a medium like air, which can be described by a function $f : \mathbb{R} \to \mathbb{R}$ mapping a given point in $t \in \mathbb{R}$ in time to an air pressure. As we know $\mathbb{R}$ is an uncountable set, thus it is not suitable for computations. Functions representing sound in the real domain are generally said to be in the *continuous-time* (CT) domain in contrast to functions in a countable domain which are said to be in a *discrete-time* (DT) domain. Before being able to work with functions from the CT domain we have to convert them to a DT domain, which is suitable for computations. For this purpose we use a *sampling rate* s_f which describes each second in time in a function from the CT domain by s_f equidistant chosen points as indicated in Figure 3.1. The sampling rate is usually given in sample per second and its unit is *Hertz* (Hz). We define an *audio signal* of duration D with sampling rate s_f

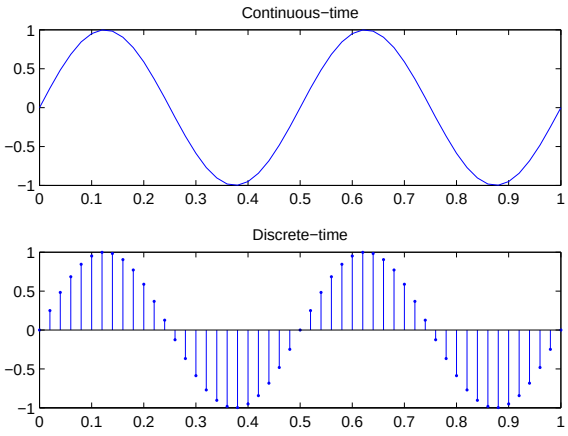

Figure 3.1. A CT function at the top and its DT conversion at the bottom showing the discrete points in time chosen on the x-axis and their respective time pressures on the y-axis

to be the DT conversion $f : [0, D \cdot s_{\mathrm{f}}] \to \mathbb{N}$ of a real world sound function restricted to an arbitrary constant interval of length D. The time-pressure plot representing an audio signal will be referred to as the *waveform*.

Often we will only refer to an extract of a whole audio signal. As a shorthand notation we will use

$$f_{[\mathrm{a,b}]}(t) := f(t), a \leq t \leq b \tag{3.1}$$

to address the function f on the interval $[a, b]$

Given an audio recording our aim is to compute a set of features in order to reduce the amount of data that represents the recording. By only extracting certain audio properties from the audio recording we convert it into a more meaningful representation, which we will call a *feature vector*. Formally it is defined as follows

$$X := (x_1, x_2, \ldots, x_{\mathrm{n}}), x_{\mathrm{i}} \in \mathbb{R}^{d_{\mathrm{i}}} \tag{3.2}$$

where x_{i} is some d_{i}-dimensional feature.

3.2 Spectrogram

In many cases the waveform representation does not offer all the information that might be needed. Determining the pitch or timbre of a given audio signal cannot be done without further ado by just using the waveform, because these properties require insights into the frequency composition of an audio signal. To acquire this information we will utilize a mathematical transformation known as *fourier transformation*, that transforms our time domain function which represents the audio signal to a frequency domain, i. e. it transforms

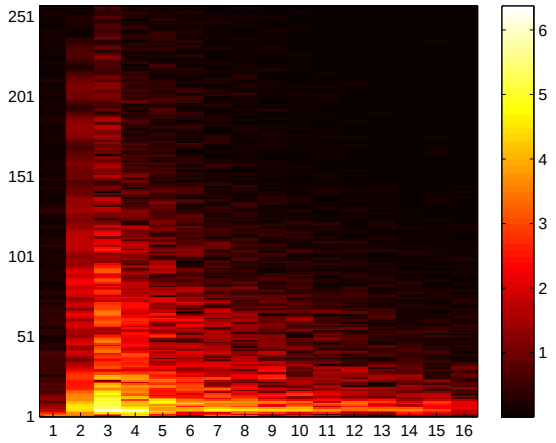

Figure 3.2. Exemplary spectrogram with the time on the x-axis, the frequencies on the y-axis and the amplitude color coded inside. The bar on the right maps the colors to their actual values.

our function that maps points in time to air pressures to a function that maps frequencies to amplitudes that indicate how strong the frequency is present in the transformed signal. Intuitively the basic idea is as follows. To calculate how strong a certain frequency ω is present in our audio signal we multiply the audio signal with a function that oscillates with a frequency of ω around zero and then integrate the result. In case the two functions oscillate with approximately the same frequency then they will have the same sign most of the time, thus multiplying them will most often result in a positive value which will make the integral bigger. If, on the other hand ω is hardly present in the signal then we will often multiply values with different signs, thus resulting in many negative values, hence the integral will be smaller.

This explanation is simplified to facilitate understanding the main points behind the fourier transformation by leaving out certain aspects which are not relevant for general understanding. For a detailed introduction to the fourier transform please refer to [5].

The fourier transform of a signal $f : D \to \mathbb{R}$ is given by

$$\widehat{f}(\omega) = \int_{t \in D} f(t) e^{-2\pi i \omega t} dt \tag{3.3}$$

with ω being the frequency. The magnitude $|\widehat{f}(\omega)|$ describes how strong ω is present in f. The complex exponential function is a periodic oscillating function with frequency ω as it follows from *Euler's Formula*:

$$e^{-2\pi i \omega t} = \cos(-2\pi i \omega t) + i \cdot \sin(-2\pi i \omega t) \tag{3.4}$$

Usually we are much more interested in the developement of the spectrum over time,

rather than the absolute amount of a certain frequency in the signal. Therefore we use a *window function* $w_{[a,b]} : \mathbb{R} \to \mathbb{R}$ with *window size* $b - a$. The function is defined to be zero for all values outside $[a, b]$ and bigger than zero inside that interval. By multiplying such a window function with the signal before applying the fourier transformation we can now examine the spectral decomposition of the signal at a certain interval, due to the fact that after the multiplication our audio signal will be zero everywhere outside of the interval $[a, b]$. Consequently we now segment the time axis into possibly overlapping intervals by multiplying with the desired window function for each interval and then apply the fourier transformation to each of these intervals separately. We call the distance between the center points of two consequent intervals the *hop-size*. The method of first segmenting the time axis and then applying the fourier transformation on each of these segments separately is called the *Short-time Fourier transformation* (STFT) and from its resulting three dimensional output we can now determine the amount, so called *magnitude*, of each frequency at each point in time. It should be noted though that we have to consider a trade-off when choosing the window size, because a smaller window size results in a better time axis resolution, albeit a worse frequency resolution. This also means, that we have frequency bins containing a range of frequencies on the frequency axis of the resulting transform rather than each frequency value by itself. Same holds for the time axis. In the following we will not work directly on the STFT, but rather on its squared absolute value and we refer to this as the *spectrogram*. Our auditory system perceives the intensity of a sound on a logarithmic scale, therefore we will be working with the logarithm of the intensities in our spectrogram. In Figure 3.2 we can see an exemplary spectrogram that was calculated using 512ms half overlapping big windows.

3.3 Zero crossing rate (ZCR) & short-time energy (STE)

The *zero crossing rate* (ZCR) is a very simple yet interesting feature for our purposes. It is widely used in speech recognition tasks or for classifying percussive sounds. It describes the rate of sign changes in our audio signal, i. e. the number of consecutive samples from our audio signal with different signs divided by the total amount of samples regarded, thus it can be calculated very easily by just going through the signal sequentially and counting these changes [4]. This feature gives us indications about which frequency dominates the audio signal. Often this feature is used in conjuction with the *short time energy* (STE) of an audio signal, which is defined as

$$\mathrm{STE}(f_{[a,b]}(t)) := \sum_{i=a}^{b} f^2(i) \tag{3.5}$$

where f is the audio signal and $[a, b]$ is the desired interval. Both of these features are calculated for each segment after segmenting the audio signal with a window function. Speech for example has a high variance in ZCR and STE values, while in music these values are normally much more constant.

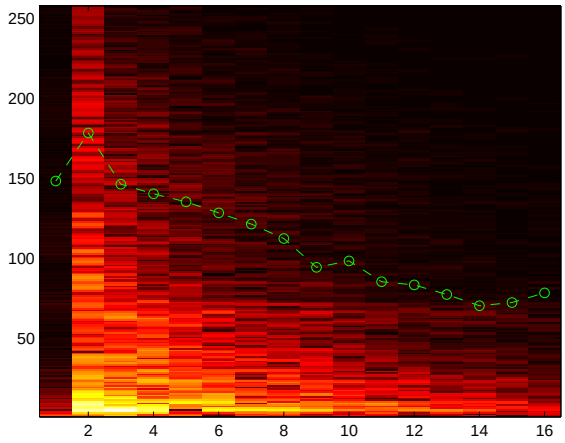

Figure 3.3. A spectrogram and its spectral rolloff function with p=80% indicated by the green graph

3.4 Spectral rolloff

When looking at a spectrogram we can see a very distinctive slope. Rather than looking at the whole specrogram by itself, we are more interested in the pathway of the energy, which brings up characteristic decay behaviour over time. The motivation behind the *spectral rolloff* is to reduce the spectrogram to a function, parameterized by a parameter precentage p, that represents this slope. For each point in time in the spectrogram the spectral rolloff is defined as the frequency at which p percent of the total energy that was present at that time is gone. An illustration of this can be seen in Figure 3.3. In our classification algorithm we will use the parameter $p = 0.95$ which we determined to be the best for our purposes.

3.5 Spectral centroid

In analogy to the visual brightness which we all know, there is also an audio *brightness*, which describes the high frequency content of a given audio signal. This feature co-determines the way we perceive the timbre of an audio signal and it directly correlates with our intuitive feeling of brightness. In signal processing this is measured by the *spectral centroid*, which is calculated from the spectrogram by the following formula.

$$\mathrm{SC_t} := \frac{\sum_{k=1}^{n} F(k) \cdot x_\mathrm{t}(k)}{\sum_{k=1}^{n} x_\mathrm{t}(k)} \tag{3.6}$$

Here $F(k)$ denotes the center frequency of the frequency range of bin k of the spectrogram and $x_{\mathrm{t}}(k)$ denotes the amplitude of bin k at time t. This formula corresponds to a weighted mean of the frequencies with the amplitudes being the weights.

3.6 Mel frequency cepstral coefficients (MFCC)

Mel frequency cepstral coefficients (MFCC) have initially been developed for speech recognition. There a speaker is modelled as a source that is generating the sound and a filter that the sound passes through. This filter then gives the speaker his characteristic timbre by filtering or boosting certain frequencies. The underlying idea of MFCCs is to separate the source and the filter and thus enabling us to only consider the filter while ignoring the source. Practice has shown that this separation of source and filter for describing the timbre of a sound also works reliably for other audio signals apart from speech. When adopting this feature to music, then one could for example separate the timbre of an instrument body like the one of a guitar from the actual string that was played. As a result from the MFCC feature extraction we will get a number of coefficients that will describe this timbre. The steps of calculation can be seen in Figure 3.6.

As we can see in the pipeline, given an audio signal, we first compute its spectrogram using the STFT. In the next step we address the problem that the frequency bins of our spectrogram are spaced in a linear fashion. This might be problematic, because our auditory system does not perceive pitch on a linear scale, meaning that when given two tones where one sounds doubled in pitch compared to the other one, then this does not imply that the frequency of one of them is doubled compared to the other one.

To avoid this discrepancy between our perception and the spectrogram scale the *mel scale* was introduced [6]. This scale correlates with our perception of pitch and given a frequency f in hertz we can convert it into mel using the following formula:

$$m = 2595 \log_{10}\left(1 + \frac{f}{700}\right) \tag{3.7}$$

The mel scale is approximately linear below 1kHz and logarithmic above.

After mapping our frequency axis to 40 linearly spaced bins on the mel scale, we take the logarithm of each value to take into account that our perception of sound intensity is logarithmic. In the last step we finally want to separate the source from the filter at each point in time. This can be achieved by applying the *discrete cosine transform* (DCT), which is similar to the fourier transform. At each point in time we can regard the spectral vector as the sum of a source and a filter component and with this transform we represent each vector as a linear combination of our DCT basis vectors. We assume that

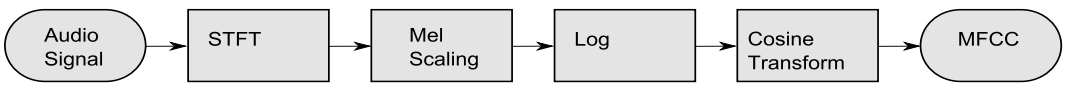

Figure 3.4. MFCC feature extraction pipeline

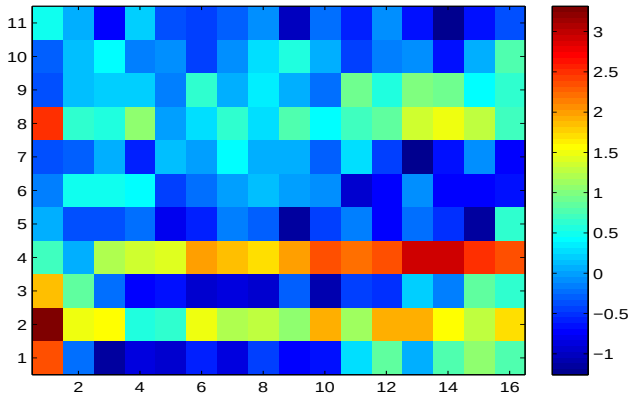

Figure 3.5. MFCC feature sequence showing $3^{rd} - 14^{th}$ coefficients of the calculated cepstral coefficients for each point in time.

the source is represented in the upper coefficients, while the filter is represented by the lower coefficients. The first coefficient is strongly correlated to the loudness and thus we disregard this as well as the higher coefficients, because they represent the source which we are not interested in. As a result we only consider the $3^{rd} - 14^{th}$ coefficients. In Figure 3.5 you can see the MFCC feature sequence for a detected bounce on carpet ground from the experiments.

Chapter 4

Distance measures

After extracting a feature vector for an unclassified bounce we need to determine which surface is most likely. We need to be able to measure the difference between our calculated feature vector and the given labelled training data. Commonly this is done by learning one representative feature vector together with some additional information for each label and then using a *distance measure* to calculate the difference between the unclassified feature vector and each of the representative feature vectors in order to determine the label with the smallest distance. In this chapter we will discuss several different distance measures as well as their assets and drawbacks. All of them have to fulfill the following properties, which are just mathematical formulations of our intuitive demands for a distance measures.

For a distance measure d and any feature vectors x, y and z it holds that:

(i) $d(x, y) \geq 0$

(ii) $d(x, y) = 0 \Leftrightarrow x = y$

(iii) $d(x, y) = d(y, x)$

(iv) $d(x, z) \leq d(x, y) + d(y, z)$

4.1 Euclidean distance

The *euclidean distance* is a very commonly used distance measure for points on a plane or a higher dimensional space. For two p-dimensional feature vectors x and y it is defined as

$$d(x, y) := \sqrt{\sum_{i=0}^{p} |x_i - y_i|^2} \tag{4.1}$$

It simply measures the difference for each feature separately and then sums these differences up. Small differences between a feature of two different feature vectors is normally

expected, while big differences indicated that these features belong to different labels. In order to punish bigger differences more than smaller ones they are being squared in the formula.

This simple measure works well if all the vectors components come from the same space. Unfortunately this is not true for our feature vectors. Each component describes a certain distinct audio property. These properties might come from completely different value spaces and thus have very different scales, making them not comparable. The euclidean distance does not take this issue into account and as a consequence features from spaces with larger values are automatically weighted to be more important than others, because they are likely to have a bigger mean and variance.

4.2 Normalized euclidean distance

As mentioned before we now want to make features from different spaces comparable, thus we need to eliminate the unit of measurement. This operation is commonly known as *normalization*. The *normalized euclidean distance* is a straight forward approach of implementing a simple normalization directly into the euclidean distance and it is defined as follows.

$$d(x, y) := \sqrt{\sum_{i=0}^{p} \frac{|x_i - y_i|^2}{\sigma_i}} \tag{4.2}$$

where σ_i is the standard deviation of the i^{th} feature.

As we can see this function is very similiar to the euclidean distance however here we divide each components difference by its standard deviation σ_i. We recall that the standard deviation expresses the dispersion of a random variable around its mean. By dividing each component with its standard deviation we ensure that the standard deviation of the resulting new random variable will be one, thus making it unit free and comparable, meaning that we can now compare it with other features that have been measured on other scales [8].

4.3 Mahalanobis distance

The last two discussed distance measures treated each feature on its own assuming, that all features in a feature vector are independent and that there exists no correlation between them. In contrast to the last two distance measures, the *mahalanobis distance* takes such possible correlations in our feature vectors into account.

To undestand the advantages of the mahalanobis distance let us consider a simple example where our feature vector has only two components. In Figure 4.3 we can see the plot of a given set of these two dimensional feature vectors, depicted in blue, where feature f_2 is plotted onto the x and feature f_1 onto the y-axis.

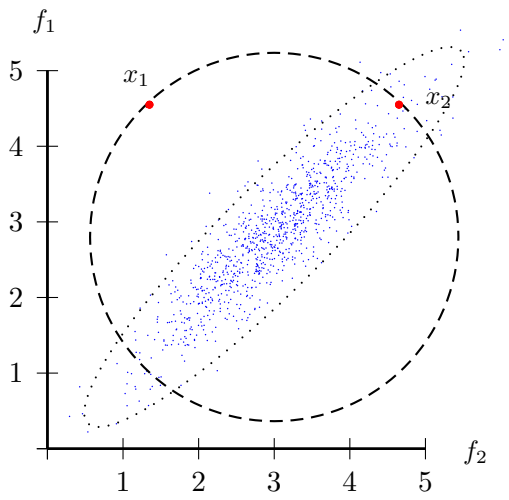

Figure 4.1. plot showing a set of two-dimensional feature vectors, with a feature f_1 on the y-axis and f_2 on the x-axis. We can see that the data points are distributed in a elliptical fashion, indicating that these two features are correlated.

Two exemplary data points x_1 and x_2 are highlighted in red. The center of the dashed circle is the mean of all the blue data points. We can see that x_1 and x_2 have the same distance according to the euclidean distance and thus both of these points are equally likely to belong to this point cloud.

Yet by just looking at the plot this contradicts our intuition as one would normally consider x_2 to be much more likely than x_1. Here we see another problem with the euclidean distance, namely the fact that it only takes into account the distances of points, but not their distribution. The elliptical shape of the point cloud suggests that there is a certain correlation between f_1 and f_2. To be more precise, it suggests that higher values of f_1 correspond with higher values of f_2 and vice versa.

When thinking of multivariate random vectors like our feature vector, where some components might be correlated, it seems natural to express these correlations through a covariance matrix. Like the distance measures described above, the mahalanobis distance is based on top of the euclidean distance as well and it is mathematically expressed as follows.

$$d(x, y) := \sqrt{\sum_{i=0}^{p} |x - y|^T S^{-1} |x - y|} \tag{4.3}$$

where S is the covariance matrix of the feature vectors.

In case all components of our feature vector are uncorrelated, then the covariance matrix will be the identity matrix and the resulting distance measure will be the euclidean distance [7].

Chapter 5

Experiment evaluation

The following chapter evaluates our classification algorithm in detail based on its classification results for the data acquired during the sports experiments.

In Chapter 3 we introduced a set of features that we used to transform our audio events into more meaningful feature vectors. Afterwards we discussed several approaches to determine the similarity of two given feature vectors in Chapter 4.

For evaluating our algorithm we need some statistical measures, which will measure its performance. These will be introduced in Section 5.1. Furthermore we will discuss and interpret the classification results for various combinations of features and distance measures in Section 5.2. To investigate the usefulness of individual features we assigned weights to them in order to examine whether certain features are more suitable than others for distinguishing between our possible labels.

5.1 Statistical measures

When evaluating our algorithm we are naturally interested in its overall performance which we will refer to as its *accuracy*

$$A := \frac{\#\text{correct classifications}}{\#\text{total classifications}} \tag{5.1}$$

This measure however can be elusive as it might not represent the actual quality of the classification algorithm. Consider a binary classification task with labels L_1 and L_2, where the probability for L_1 is 95%. Now we simply build a classification algorithm that outputs L_1 for any input. Such an algorithm would have an expected accuracy of 95%, but it would still be a very bad one. Therefore we introduce two other statistical measures, which will give us further insights into the performance for each label. These two measures are commonly known as *precision* and *recall*[9].

The precision for a label L_i is defined as

$$P_{L_i} := \frac{\#\text{events correctly predicted as } L_i}{\#\text{total events predicted to be } L_i} \tag{5.2}$$

This measure reflects how likely it is, that an event actually has the label L_i given that it is labelled as such.

The recall is defined as

$$R_{L_i} := \frac{\#\text{events correctly predicted as } L_i}{\#\text{total events that are actually } L_i} \tag{5.3}$$

It reflects how likely it is, that an event that should be labelled with L_i will be labelled as such.

There is a trade-off between these two measures. Let us reconsider our example of a primitive classificator from above. That classificator would have a perfect recall for L_1 as it would label all events that are L_1 as such. The precision for L_1 would be rather good due to the fact that almost all input would be L_1 anyway. In contrast to that the recall and precision for L_2 would be zero, because not a single event would be labelled correctly as L_2.

Finally we will use the so called *F-measure* to obtain an overall score for a label. It is defined as the harmonic mean of recall and precision:

$$Fm_{L_i} := \frac{P_{L_i} \cdot R_{L_i}}{P_{L_i} + R_{L_i}} \tag{5.4}$$

5.2 Classification results and discussion

In order to determine the best classification parameters we combined different feature vectors with different distance measures and calculated the statistical measures from Section 5.1 for each of these combinations. For each experiment we obtained one table per distance measure. All tables can be found in Appendix A.

One of these tables can be seen in Table 5.2. The first four columns on the left indicate which features were used, where $\mathrm{w}_m, \mathrm{w}_z, \mathrm{w}_e$ and w_c are the weights for mfcc, zcr, energy rolloff and spectral centroid respectively, meaning features with a weight of one are being used, while the others are not. The middle bar of columns represents our performance measures for the labels C (carpet) and G (ground), what we discussed in Section 5.1. On the right hand side we see the f-measures for our two possible labels. The best results for each column are highlighted through a bold font.

12% of all events in Table 5.2 have been carpet bounces. As we can see we have a peak F-measure of 0.98 and 0.90 and an accuracy of 0.97 when using a feature vector consisting of the energy rolloff and the spectral centroid. With an accuracy of 0.97 we are better than a trivial classification algorithm, that would only produce 0.88 as we discussed above. In other words we would classify 3 out of 100 bounces wrong. This is by no means worse than

w_m	w_z	w_e	w_c	Acc	P_C	R_C	P_G	R_G	Fm_G	Fm_C
0	0	0	1	0.9726	**0.9492**	0.8235	0.9755	**0.9938**	0.9846	0.8818
0	0	1	0	0.9635	0.8333	**0.8824**	**0.9832**	0.975	0.979	0.8572
0	0	1	1	**0.9763**	0.9365	0.8676	0.9814	0.9917	**0.9866**	**0.9008**
0	1	0	0	0.9416	0.7368	0.8235	0.9746	0.9583	0.9664	0.7778
0	1	0	1	0.9726	**0.9492**	0.8235	0.9755	**0.9938**	0.9846	0.8818
0	1	1	0	0.9653	0.8551	0.8676	0.9812	0.9792	0.9802	0.8614
0	1	1	1	**0.9763**	0.9365	0.8676	0.9814	0.9917	**0.9866**	**0.9008**
1	0	0	0	0.9416	0.75	0.7941	0.9706	0.9625	0.9666	0.7714
1	0	0	1	0.969	0.9322	0.8088	0.9734	0.9917	0.9824	0.8662
1	0	1	0	0.9599	0.8485	0.8235	0.9751	0.9792	0.9772	0.8358
1	0	1	1	0.9708	0.9333	0.8235	0.9754	0.9917	0.9834	0.875
1	1	0	0	0.9489	0.7857	0.8088	0.9728	0.9688	0.9708	0.797
1	1	0	1	0.969	0.9322	0.8088	0.9734	0.9917	0.9824	0.8662
1	1	1	0	0.9599	0.8485	0.8235	0.9751	0.9792	0.9772	0.8358
1	1	1	1	0.9726	**0.9492**	0.8235	0.9755	**0.9938**	0.9846	0.8818

Table 5.1. Evaluation of the algorithms performance using the normalized euclidean distance with all possible combinations of features for Group 1. The best value of each column is highlighted. 12% of all events in this experiment were bounces on carpet.

a human could do and therefore we can conclude that for group 1 automated classification would be a feasible and efficient solution. We can also observe that for group 1 the zero crossing rate and the MFCCs perform rather poorly when compared to all other results. For these two features and their combination the Fm_C even reaches its lowest values.

From the whole dataset and its evaluation we were able to observe several things. Most of what we described above for group 1 also holds for the other groups. After analyzing the results we concluded that our set of training recordings was not sufficiently large to be able to compute representative values for statistical values like the standard deviation or the covariance. As a consequence the normalized euclidean distance and the mahalanobis distance did not perform as well as they could have with a larger training dataset. The normalized euclidean distance was not affected too much and still performed equally well as the euclidean distance. The mahalanobis distance on the other hand performed very poorly, because the covariance matrix did not represent the correlations between the individual features correctly and therefore its output became unpredictable to a certain extend.

Besides that we could also conclude that the spectral centroid and the energy rolloff function were the most distinctive features yielding the the best Fm_C, which often is a deficiency of the other features, which tend to perform worse for the carpet surface.

When inspecting the features individually we noticed that MFCCs and ZCR tend to perform worse, which is surprising in the case of MFCCs, because our initial approach was to choose features that represent timbre properties, since these are the ones we as humans are using when classifying these events, and therefore MFCC should be very characteristic.

In most of the cases combining features yields in a better performance. For all experiments,

the classification results with all features used, tend to be between the best results. Further data can be found in Appendix A.

Chapter 6

Conclusions

In this thesis we investigated the usability of information retrieval techniques with applications to the evaluation of sport experiments. First we started with introducing the motivation and concepts behind the experiment. We have seen that through Fitts' law we can describe the time needed to reach a target as a function of the distance to and the size of the target. Next we examined the experiments details as well as its modifications that were needed to obtain the audio recordings. We have seen that only minor changes to the experiments setup were needed in order to enable an automated classification. To evaluate our classification approach we had to cut and annotate all recorded data by hand before beginning with our actual algorithm evaluation.

In Chapter 3 we tackled the signal processing site of our algorithm, by investigating several features as well as methods for extracting them from a given waveform. We paid special attention to features that described timbre properties of our audio events, as we assumed these properties might be the most suitable ones for our task. We have seen how to efficiently convert the high dimensional waveform extracts that describe the events we are interested in, into small feature vectors.

In order to compare these different vectors we discussed several measurements, namely the euclidean, normalized euclidean and mahalanobis distance. Upfront we expected the mahalanobis distance to perform best, but this turned out to be wrong as the training set was not sufficiently large to generate meaningful covariance matrices. Nonetheless we obtained good F-measures in Chapter 5 for our two other distance measures combined with different feature combinations. We could conclude that the spectral centroid and the energy rolloff were the most distinctive features and always contributed to a high F-measure.

We have shown that it is indeed possible to use audio classification techniques to automate the evaluation process for such experiments. This opens up a whole new field of applications for audio classification and drastically reduces the amount of tedious work such an experiment brings along.

Appendix A

Full classification results

This chapter contains all evaluations of the algorithms classification results, which have been discussed in Chapter 5.

Table A.1. Group 1 using Euclidean distance measure

w_m	w_z	w_e	w_c	Acc	P_C	R_C	P_G	R_G	Fm_G	Fm_C
0	0	0	1	**0.9763**	0.8873	**0.9265**	**0.9895**	0.9833	**0.9864**	**0.9064**
0	0	1	0	0.9434	0.9512	0.5735	0.9428	**0.9958**	0.9686	0.7156
0	0	1	1	0.9745	0.8857	0.9118	0.9874	0.9833	0.9854	0.8986
0	1	0	0	0.9361	0.7037	0.8382	0.9764	0.95	0.963	0.765
0	1	0	1	0.9745	0.8857	0.9118	0.9874	0.9833	0.9854	0.8986
0	1	1	0	0.9708	**0.9643**	0.7941	0.9715	**0.9958**	0.9834	0.871
0	1	1	1	0.9745	0.8857	0.9118	0.9874	0.9833	0.9854	0.8986
1	0	0	0	0.9343	0.7581	0.6912	0.9568	0.9688	0.9628	0.7232
1	0	0	1	**0.9763**	0.8873	**0.9265**	**0.9895**	0.9833	**0.9864**	**0.9064**
1	0	1	0	0.9434	0.9512	0.5735	0.9428	**0.9958**	0.9686	0.7156
1	0	1	1	0.9745	0.8857	0.9118	0.9874	0.9833	0.9854	0.8986
1	1	0	0	0.9361	0.7037	0.8382	0.9764	0.95	0.963	0.765
1	1	0	1	0.9745	0.8857	0.9118	0.9874	0.9833	0.9854	0.8986
1	1	1	0	0.9708	**0.9643**	0.7941	0.9715	**0.9958**	0.9834	0.871
1	1	1	1	0.9745	0.8857	0.9118	0.9874	0.9833	0.9854	0.8986

Table A.2. Group 1 using Normalized euclidean distance measure

w_m	w_z	w_e	w_c	Acc	P_C	R_C	P_G	R_G	Fm_G	Fm_C
0	0	0	1	0.9726	**0.9492**	0.8235	0.9755	**0.9938**	0.9846	0.8818
0	0	1	0	0.9635	0.8333	**0.8824**	**0.9832**	0.975	0.979	0.8572
0	0	1	1	**0.9763**	0.9365	0.8676	0.9814	0.9917	**0.9866**	**0.9008**
0	1	0	0	0.9416	0.7368	0.8235	0.9746	0.9583	0.9664	0.7778
0	1	0	1	0.9726	**0.9492**	0.8235	0.9755	**0.9938**	0.9846	0.8818
0	1	1	0	0.9653	0.8551	0.8676	0.9812	0.9792	0.9802	0.8614
0	1	1	1	**0.9763**	0.9365	0.8676	0.9814	0.9917	**0.9866**	**0.9008**
1	0	0	0	0.9416	0.75	0.7941	0.9706	0.9625	0.9666	0.7714
1	0	0	1	0.969	0.9322	0.8088	0.9734	0.9917	0.9824	0.8662
1	0	1	0	0.9599	0.8485	0.8235	0.9751	0.9792	0.9772	0.8358
1	0	1	1	0.9708	0.9333	0.8235	0.9754	0.9917	0.9834	0.875
1	1	0	0	0.9489	0.7857	0.8088	0.9728	0.9688	0.9708	0.797
1	1	0	1	0.969	0.9322	0.8088	0.9734	0.9917	0.9824	0.8662
1	1	1	0	0.9599	0.8485	0.8235	0.9751	0.9792	0.9772	0.8358
1	1	1	1	0.9726	**0.9492**	0.8235	0.9755	**0.9938**	0.9846	0.8818

Table A.3. Group 1 using Mahalanobis distance measure

w_s	w_z	w_e	w_c	Acc	P_C	R_C	P_G	R_G	Fm_G	Fm_C
0	0	0	1	0.7682	0.219	0.3382	0.8984	0.8292	0.8624	0.2658
0	0	1	0	0.885	0.5333	0.5882	0.9408	0.9271	0.9338	0.5594
0	0	1	1	0.6277	0.2405	0.9265	0.9825	0.5854	0.7336	0.3818
0	1	0	0	0.9416	0.7368	0.8235	0.9746	0.9583	0.9664	0.7778
0	1	0	1	0.8887	0.6842	0.1912	0.896	**0.9875**	0.9396	0.2988
0	1	1	0	0.6825	0.25	0.7794	0.9554	0.6687	0.7868	0.3786
0	1	1	1	0.9361	**0.7619**	0.7059	0.9588	0.9688	0.9638	0.7328
1	0	0	0	0.1241	0.1241	**1.0**		0.0	0.0	0.2208
1	0	0	1	0.7573	0.2358	0.4265	0.9082	0.8042	0.853	0.3036
1	0	1	0	0.8887	0.5402	0.6912	0.9544	0.9167	0.9352	0.6064
1	0	1	1	0.7281	0.3116	0.9853	0.997	0.6917	0.8168	0.4734
1	1	0	0	0.9343	0.7162	0.7794	0.9684	0.9563	0.9624	0.7464
1	1	0	1	0.9124	0.6852	0.5441	0.9372	0.9646	0.9508	0.6066
1	1	1	0	0.4526	0.1723	0.8971	0.9639	0.3896	0.555	0.289
1	1	1	1	**0.9507**	0.7595	0.8824	**0.9829**	0.9604	**0.9716**	**0.8164**

Table A.4. Group 2 using Euclidean distance measure

w_m	w_z	w_e	w_c	Acc	P_C	R_C	P_G	R_G	Fm_G	Fm_C
0	0	0	1	0.9664	0.5854	0.8571	0.9932	0.9715	0.9822	0.6956
0	0	1	0	**0.9856**	0.8065	**0.8929**	**0.9949**	0.9899	**0.9924**	**0.8476**
0	0	1	1	0.9664	0.5854	0.8571	0.9932	0.9715	0.9822	0.6956
0	1	0	0	0.7424	0.1243	0.7857	0.9866	0.7404	0.846	0.2146
0	1	0	1	0.9664	0.5854	0.8571	0.9932	0.9715	0.9822	0.6956
0	1	1	0	**0.9856**	**0.8276**	0.8571	0.9933	**0.9916**	**0.9924**	0.842
0	1	1	1	0.9664	0.5854	0.8571	0.9932	0.9715	0.9822	0.6956
1	0	0	0	0.9584	0.5217	0.8571	0.9931	0.9631	0.9778	0.6486
1	0	0	1	0.9664	0.5854	0.8571	0.9932	0.9715	0.9822	0.6956
1	0	1	0	**0.9856**	0.8065	**0.8929**	**0.9949**	0.9899	**0.9924**	**0.8476**
1	0	1	1	0.9664	0.5854	0.8571	0.9932	0.9715	0.9822	0.6956
1	1	0	0	0.7664	0.1358	0.7857	0.987	0.7655	0.8622	0.2316
1	1	0	1	0.9664	0.5854	0.8571	0.9932	0.9715	0.9822	0.6956
1	1	1	0	**0.9856**	**0.8276**	0.8571	0.9933	**0.9916**	**0.9924**	0.842
1	1	1	1	0.9664	0.5854	0.8571	0.9932	0.9715	0.9822	0.6956

Table A.5. Group 2 using Normalized euclidean distance measure

w_m	w_z	w_e	w_c	Acc	P_C	R_C	P_G	R_G	Fm_G	Fm_C
0	0	0	1	0.9744	0.65	0.9286	0.9966	0.9765	0.9864	0.7648
0	0	1	0	0.9648	0.56	**1.0**	**1.0**	0.9631	0.9812	0.718
0	0	1	1	0.9728	0.6279	0.9643	0.9983	0.9732	0.9856	0.7606
0	1	0	0	0.68	0.1019	0.7857	0.9853	0.675	0.8012	0.1804
0	1	0	1	0.9744	0.65	0.9286	0.9966	0.9765	0.9864	0.7648
0	1	1	0	0.9664	0.5714	**1.0**	**1.0**	0.9648	0.982	0.7272
0	1	1	1	0.9728	0.6279	0.9643	0.9983	0.9732	0.9856	0.7606
1	0	0	0	0.96	0.5283	**1.0**	**1.0**	0.9581	0.9786	0.6914
1	0	0	1	**0.9808**	**0.7105**	0.9643	0.9983	**0.9816**	**0.9898**	**0.8182**
1	0	1	0	0.9616	0.5385	**1.0**	**1.0**	0.9598	0.9794	0.7
1	0	1	1	0.976	0.6585	0.9643	0.9983	0.9765	0.9872	0.7826
1	1	0	0	0.9616	0.54	0.9643	0.9983	0.9615	0.9796	0.6924
1	1	0	1	**0.9808**	**0.7105**	0.9643	0.9983	**0.9816**	**0.9898**	**0.8182**
1	1	1	0	0.9616	0.5385	**1.0**	**1.0**	0.9598	0.9794	0.7
1	1	1	1	0.976	0.6585	0.9643	0.9983	0.9765	0.9872	0.7826

Table A.6. Group 2 using Mahalanobis distance measure

w_s	w_z	w_e	w_c	Acc	P_C	R_C	P_G	R_G	Fm_G	Fm_C
0	0	0	1	0.912	0.2632	0.5357	0.9771	0.9296	0.9528	0.353
0	0	1	0	0.896	0.3011	**1.0**	**1.0**	0.8911	0.9424	**0.4628**
0	0	1	1	0.2128	0.0521	0.9643	0.9907	0.1776	0.3012	0.0988
0	1	0	0	0.68	0.1019	0.7857	0.9853	0.675	0.8012	0.1804
0	1	0	1	0.808	0.1515	0.7143	0.9838	0.8124	0.89	0.25
0	1	1	0	0.664	0.1144	0.9643	0.9974	0.6499	0.787	0.2046
0	1	1	1	0.5168	0.0848	**1.0**	**1.0**	0.4941	0.6614	0.1564
1	0	0	0	0.3424	0.0102	0.1429	0.8974	0.3518	0.5054	0.019
1	0	0	1	0.8768	0.2247	0.7143	0.9851	0.8844	0.932	0.3418
1	0	1	0	0.8832	0.2772	**1.0**	**1.0**	0.8777	0.9348	0.434
1	0	1	1	0.064	0.0457	**1.0**	**1.0**	0.0201	0.0394	0.0874
1	1	0	0	0.6016	0.0952	0.9286	0.9943	0.5863	0.7376	0.1726
1	1	0	1	**0.9216**	**0.3019**	0.5714	0.979	**0.938**	**0.958**	0.395
1	1	1	0	0.776	0.1627	0.9643	0.9978	0.7672	0.8674	0.2784
1	1	1	1	0.5728	0.0949	**1.0**	**1.0**	0.5528	0.712	0.1734

Table A.7. Group 3 using Euclidean distance measure

w_m	w_z	w_e	w_c	Acc	P_C	R_C	P_G	R_G	Fm_G	Fm_C
0	0	0	1	0.9747	0.7647	**0.7879**	**0.9875**	0.9857	0.9866	**0.7762**
0	0	1	0	0.9764	0.88	0.6667	0.9807	0.9947	0.9876	0.7586
0	0	1	1	0.9747	0.7647	**0.7879**	**0.9875**	0.9857	0.9866	**0.7762**
0	1	0	0	0.8519	0.2043	0.5758	0.9721	0.8681	0.9172	0.3016
0	1	0	1	0.9747	0.7647	**0.7879**	**0.9875**	0.9857	0.9866	**0.7762**
0	1	1	0	**0.9781**	**0.9167**	0.6667	0.9807	**0.9964**	**0.9884**	0.772
0	1	1	1	0.9747	0.7647	**0.7879**	**0.9875**	0.9857	0.9866	**0.7762**
1	0	0	0	0.9562	0.5778	**0.7879**	**0.9872**	0.9661	0.9766	0.6666
1	0	0	1	0.9747	0.7647	**0.7879**	**0.9875**	0.9857	0.9866	**0.7762**
1	0	1	0	0.9764	0.88	0.6667	0.9807	0.9947	0.9876	0.7586
1	0	1	1	0.9747	0.7647	**0.7879**	**0.9875**	0.9857	0.9866	**0.7762**
1	1	0	0	0.8552	0.2088	0.5758	0.9722	0.8717	0.9192	0.3064
1	1	0	1	0.9747	0.7647	**0.7879**	**0.9875**	0.9857	0.9866	**0.7762**
1	1	1	0	**0.9781**	**0.9167**	0.6667	0.9807	**0.9964**	**0.9884**	0.772
1	1	1	1	0.9747	0.7647	**0.7879**	**0.9875**	0.9857	0.9866	**0.7762**

Table A.8. Group 3 using Normalized euclidean distance measure

w_m	w_z	w_e	w_c	Acc	P_C	R_C	P_G	R_G	Fm_G	Fm_C
0	0	0	1	0.9815	0.8438	0.8182	0.9893	0.9911	0.9902	0.8308
0	0	1	0	0.9882	0.8421	0.9697	**0.9982**	0.9893	0.9938	0.9014
0	0	1	1	0.9848	0.8158	0.9394	0.9964	0.9875	0.992	0.8732
0	1	0	0	0.8283	0.1892	0.6364	0.9752	0.8396	0.9024	0.2916
0	1	0	1	0.9798	0.8387	0.7879	0.9876	0.9911	0.9894	0.8126
0	1	1	0	0.9865	0.8378	0.9394	0.9964	0.9893	0.9928	0.8856
0	1	1	1	0.9848	0.8158	0.9394	0.9964	0.9875	0.992	0.8732
1	0	0	0	0.9714	0.7105	0.8182	0.9892	0.9804	0.9848	0.7606
1	0	0	1	0.9899	**0.9091**	0.9091	0.9947	**0.9947**	0.9948	0.9092
1	0	1	0	0.9899	0.8857	0.9394	0.9964	0.9929	0.9946	0.9118
1	0	1	1	**0.9916**	0.8889	0.9697	**0.9982**	0.9929	**0.9956**	**0.9276**
1	1	0	0	0.9714	0.7105	0.8182	0.9892	0.9804	0.9848	0.7606
1	1	0	1	0.9899	**0.9091**	0.9091	0.9947	**0.9947**	0.9948	0.9092
1	1	1	0	0.9899	0.8857	0.9394	0.9964	0.9929	0.9946	0.9118
1	1	1	1	**0.9916**	0.8889	0.9697	**0.9982**	0.9929	**0.9956**	**0.9276**

Table A.9. Group 3 using Mahalanobis distance measure

w_s	w_z	w_e	w_c	Acc	P_C	R_C	P_G	R_G	Fm_G	Fm_C
0	0	0	1	0.6633	0.0966	0.6061	0.9664	0.6667	0.789	0.1666
0	0	1	0	0.7172	0.1053	0.5455	0.9645	0.7273	0.8292	0.1766
0	0	1	1	0.9512	0.5909	0.3939	0.965	0.984	0.9744	0.4726
0	1	0	0	0.8283	0.1892	0.6364	0.9752	0.8396	0.9024	0.2916
0	1	0	1	0.7037	0.096	0.5152	0.9616	0.7148	0.82	0.1618
0	1	1	0	0.601	0.1077	**0.8485**	0.985	0.5865	0.7352	0.1912
0	1	1	1	0.9343	0.4464	0.7576	0.9851	0.9447	0.9644	0.5618
1	0	0	0	0.3721	0.055	0.6364	0.9434	0.3565	0.5174	0.1012
1	0	0	1	0.8182	0.2047	0.7879	0.985	0.82	0.895	0.325
1	0	1	0	0.9259	0.3333	0.3333	0.9608	0.9608	0.9608	0.3332
1	0	1	1	0.9461	0.5294	0.2727	0.9584	**0.9857**	0.9718	0.36
1	1	0	0	0.8636	0.26	0.7879	**0.9858**	0.8681	0.9232	0.391
1	1	0	1	0.8283	0.1835	0.6061	0.9732	0.8414	0.9026	0.2818
1	1	1	0	**0.9596**	**0.6286**	0.6667	0.9803	0.9768	**0.9786**	0.647
1	1	1	1	0.9562	0.5814	0.7576	0.9855	0.9679	0.9766	**0.658**

Table A.10. Group 4 using Euclidean distance measure

w_m	w_z	w_e	w_c	Acc	P_C	R_C	P_G	R_G	Fm_G	Fm_C
0	0	0	1	0.9449	0.6024	**0.9259**	**0.9932**	0.9466	0.9694	0.73
0	0	1	0	**0.9836**	**0.9574**	0.8333	0.9856	**0.9968**	**0.9912**	**0.891**
0	0	1	1	0.9449	0.6024	**0.9259**	**0.9932**	0.9466	0.9694	0.73
0	1	0	0	0.7515	0.2011	0.7037	0.9669	0.7557	0.8484	0.3128
0	1	0	1	0.9449	0.6024	**0.9259**	**0.9932**	0.9466	0.9694	0.73
0	1	1	0	0.9792	0.9167	0.8148	0.984	0.9935	0.9888	0.8628
0	1	1	1	0.9449	0.6024	**0.9259**	**0.9932**	0.9466	0.9694	0.73
1	0	0	0	0.9033	0.4476	0.8704	0.9877	0.9061	0.9452	0.5912
1	0	0	1	0.9449	0.6024	**0.9259**	**0.9932**	0.9466	0.9694	0.73
1	0	1	0	**0.9836**	**0.9574**	0.8333	0.9856	**0.9968**	**0.9912**	**0.891**
1	0	1	1	0.9449	0.6024	**0.9259**	**0.9932**	0.9466	0.9694	0.73
1	1	0	0	0.7589	0.2065	0.7037	0.9672	0.7638	0.8536	0.3194
1	1	0	1	0.9449	0.6024	**0.9259**	**0.9932**	0.9466	0.9694	0.73
1	1	1	0	0.9792	0.9167	0.8148	0.984	0.9935	0.9888	0.8628
1	1	1	1	0.9449	0.6024	**0.9259**	**0.9932**	0.9466	0.9694	0.73

Table A.11. Group 4 using Normalized euclidean distance measure

w_m	w_z	w_e	w_c	Acc	P_C	R_C	P_G	R_G	Fm_G	Fm_C
0	0	0	1	0.939	0.5714	0.963	0.9966	0.9369	0.9658	0.7172
0	0	1	0	0.9539	0.642	0.963	0.9966	0.9531	0.9744	0.7704
0	0	1	1	**0.9568**	**0.6582**	0.963	0.9966	**0.9563**	0.976	0.782
0	1	0	0	0.6458	0.1618	0.8148	0.975	0.6311	0.7662	0.27
0	1	0	1	0.9375	0.5652	0.963	0.9966	0.9353	0.965	0.7124
0	1	1	0	0.9539	0.642	0.963	0.9966	0.9531	0.9744	0.7704
0	1	1	1	**0.9568**	**0.6582**	0.963	0.9966	**0.9563**	0.976	0.782
1	0	0	0	0.8988	0.4397	0.9444	0.9946	0.8948	0.942	0.6
1	0	0	1	0.942	0.5824	**0.9815**	**0.9983**	0.9385	0.9674	0.731
1	0	1	0	0.942	0.5843	0.963	0.9966	0.9401	0.9676	0.7274
1	0	1	1	0.9554	0.6463	**0.9815**	**0.9983**	0.9531	**0.9752**	**0.7794**
1	1	0	0	0.9018	0.4474	0.9444	0.9946	0.8981	0.9438	0.6072
1	1	0	1	0.942	0.5824	**0.9815**	**0.9983**	0.9385	0.9674	0.731
1	1	1	0	0.9435	0.5909	0.963	0.9966	0.9417	0.9684	0.7324
1	1	1	1	0.9539	0.6386	**0.9815**	**0.9983**	0.9515	0.9744	0.7738

Table A.12. Group 4 using Mahalanobis distance measure

w_s	w_z	w_e	w_c	Acc	P_C	R_C	P_G	R_G	Fm_G	Fm_C
0	0	0	1	0.5565	0.1474	0.9444	0.9908	0.5227	0.6844	0.255
0	0	1	0	0.311	0.0966	0.9074	0.9697	0.2589	0.4086	0.1746
0	0	1	1	0.3869	0.0968	0.7963	0.9518	0.3511	0.513	0.1726
0	1	0	0	0.6458	0.1618	0.8148	0.975	0.6311	0.7662	0.27
0	1	0	1	0.6935	0.186	0.8333	0.9791	0.6812	0.8034	0.3042
0	1	1	0	0.2515	0.0955	0.9815	0.9915	0.1877	0.3156	0.174
0	1	1	1	0.7887	0.2349	0.7222	0.9704	0.7945	0.8736	0.3544
1	0	0	0	0.1577	0.0803	0.9074	0.9194	0.0922	0.1676	0.1476
1	0	0	1	0.6131	0.1038	0.5	0.9345	0.623	0.7476	0.172
1	0	1	0	0.247	0.0964	**1.0**	**1.0**	0.1812	0.3068	0.1758
1	0	1	1	0.2188	0.0933	**1.0**	**1.0**	0.1505	0.2616	0.1706
1	1	0	0	0.8676	0.3529	0.7778	0.9783	0.8754	0.924	0.4856
1	1	0	1	0.5342	0.1331	0.8704	0.9781	0.5049	0.666	0.2308
1	1	1	0	0.2068	0.092	**1.0**	**1.0**	0.1375	0.2418	0.1684
1	1	1	1	**0.8929**	**0.4211**	0.8889	0.9892	**0.8932**	**0.9388**	**0.5714**

Table A.13. Group 5 using Euclidean distance measure

w_m	w_z	w_e	w_c	Acc	P_C	R_C	P_G	R_G	Fm_G	Fm_C
0	0	0	1	0.9665	0.6111	**1.0**	**1.0**	0.9646	0.982	0.7586
0	0	1	0	**0.9824**	**0.8929**	0.7576	0.9866	**0.9949**	**0.9908**	**0.8198**
0	0	1	1	0.9681	0.6226	**1.0**	**1.0**	0.9663	0.9828	0.7674
0	1	0	0	0.8738	0.287	0.9394	0.9961	0.8702	0.929	0.4396
0	1	0	1	0.9665	0.6111	**1.0**	**1.0**	0.9646	0.982	0.7586
0	1	1	0	0.9792	0.8125	0.7879	0.9882	0.9899	0.989	0.8
0	1	1	1	0.9681	0.6226	**1.0**	**1.0**	0.9663	0.9828	0.7674
1	0	0	0	0.9585	0.5814	0.7576	0.9863	0.9696	0.9778	0.658
1	0	0	1	0.9665	0.6111	**1.0**	**1.0**	0.9646	0.982	0.7586
1	0	1	0	**0.9824**	**0.8929**	0.7576	0.9866	**0.9949**	**0.9908**	**0.8198**
1	0	1	1	0.9681	0.6226	**1.0**	**1.0**	0.9663	0.9828	0.7674
1	1	0	0	0.877	0.2925	0.9394	0.9962	0.8735	0.9308	0.446
1	1	0	1	0.9665	0.6111	**1.0**	**1.0**	0.9646	0.982	0.7586
1	1	1	0	0.9776	0.7879	0.7879	0.9882	0.9882	0.9882	0.788
1	1	1	1	0.9681	0.6226	**1.0**	**1.0**	0.9663	0.9828	0.7674

Table A.14. Group 5 using Normalized euclidean distance measure

w_m	w_z	w_e	w_c	Acc	P_C	R_C	P_G	R_G	Fm_G	Fm_C
0	0	0	1	0.9553	0.541	**1.0**	**1.0**	0.9528	0.9758	0.7022
0	0	1	0	0.9569	0.5577	0.8788	0.993	0.9612	0.9768	0.6824
0	0	1	1	0.9665	0.62	0.9394	0.9965	0.968	0.982	0.747
0	1	0	0	0.9105	0.3647	0.9394	0.9963	0.9089	0.9506	0.5254
0	1	0	1	0.9537	0.5323	**1.0**	**1.0**	0.9511	0.975	0.6948
0	1	1	0	0.9585	0.5686	0.8788	0.993	0.9629	0.9778	0.6904
0	1	1	1	0.9665	0.62	0.9394	0.9965	0.968	0.982	0.747
1	0	0	0	0.9425	0.4727	0.7879	0.9877	0.9511	0.969	0.5908
1	0	0	1	**0.9712**	**0.6531**	0.9697	0.9983	**0.9713**	**0.9846**	**0.7806**
1	0	1	0	0.9537	0.5385	0.8485	0.9913	0.9595	0.9752	0.6588
1	0	1	1	0.9696	0.6458	0.9394	0.9965	**0.9713**	0.9838	0.7654
1	1	0	0	0.9457	0.4909	0.8182	0.9895	0.9528	0.9708	0.6136
1	1	0	1	**0.9712**	**0.6531**	0.9697	0.9983	**0.9713**	**0.9846**	**0.7806**
1	1	1	0	0.9553	0.549	0.8485	0.9913	0.9612	0.976	0.6666
1	1	1	1	0.9696	0.6458	0.9394	0.9965	**0.9713**	0.9838	0.7654

Table A.15. Group 5 using Mahalanobis distance measure

w_s	w_z	w_e	w_c	Acc	P_C	R_C	P_G	R_G	Fm_G	Fm_C
0	0	0	1	0.4393	0.0405	0.4242	0.9321	0.4401	0.5978	0.074
0	0	1	0	0.5288	0.0774	0.7273	0.9715	0.5177	0.6754	0.14
0	0	1	1	0.5799	0.0611	0.4848	0.9533	0.5852	0.7252	0.1086
0	1	0	0	**0.9105**	0.3647	0.9394	0.9963	**0.9089**	**0.9506**	**0.5254**
0	1	0	1	0.4377	0.0814	0.9394	0.9918	0.4098	0.58	0.1498
0	1	1	0	0.2636	0.0597	0.8788	0.9714	0.2293	0.371	0.1118
0	1	1	1	0.3834	0.0767	**0.9697**	0.9952	0.3508	0.5188	0.1422
1	0	0	0	0.0895	0.0532	**0.9697**	0.96	0.0405	0.0778	0.1008
1	0	0	1	0.4728	0.0745	0.7879	0.9747	0.4553	0.6206	0.1362
1	0	1	0	0.3387	0.06	0.7879	0.9637	0.3137	0.4734	0.1116
1	0	1	1	0.3275	0.0708	**0.9697**	0.9943	0.2917	0.451	0.132
1	1	0	0	0.7428	0.1667	**0.9697**	**0.9977**	0.7302	0.8432	0.2844
1	1	0	1	0.393	0.0758	0.9394	0.9908	0.3626	0.531	0.1402
1	1	1	0	0.3307	0.0711	**0.9697**	0.9943	0.2951	0.4552	0.1324
1	1	1	1	0.2284	0.0623	**0.9697**	0.9911	0.1872	0.315	0.117

Bibliography

[1] H. Drewes, *Only one fitts' law formula please!*, in Proceedings of the 28th of the international conference extended abstracts on Human factors in computing systems, CHI EA '10, New York, NY, USA, 2010, ACM, pp. 2813–2822.

[2] J. M. Fine and E. L. Amazeen, *Interpersonal Fitts law: when two perform as one*, Springer, 2011.

[3] P. M. Fitts, *The information capacity of the human motor system in controlling the amplitude of movement*, Journal of experimental psychology, 1954.

[4] F. Gouyon, F. Pachet, and O. Delerue, *On the use of zero-crossing rate for an application of classification of percussive sounds*, in Proceedings of the COST G-6 Conference on Digital Audio Effects (DAFX-00, 2000.

[5] M. Mller, *Information Retrieval for Music and Motion.*, Springer, 2007.

[6] S. S. Stevens, Je, and E. B. Newman, *A scale for the measurement of the psychological magnitude of pitch*, J. Acoust Soc Amer, (1937), pp. 185–190.

[7] G. Taguchi and R. Jugulum, *The Mahalanobis-Taguchi Strategy: A Pattern Technology System*, J. Wiley, 2002.

[8] H. Zaman, P. Robinson, and M. Petrou, *Visual Informatics: Bridging Research and Practice: First International Visual Informatics Conference, IVIC 2009 Kuala Lumpur, Malaysia, November 11-13, 2009 Proceedings*, Lecture Notes in Computer Science, Springer, 2009.

[9] M. Zhu, *Recall, precision and average precision*, (2004).