Master Thesis

Keyword Spotting for Emergency

conducted at the
Signal Processing and Speech Communication Lab
Graz University of Technology

by
Andreas Zehetner

Supervisors:
Assoc.Prof. Dipl.-Ing. Dr.mont. Franz Pernkopf
Dipl.-Ing. Dr. techn. Martin Hagmüller
Univ.-Prof. Dipl.-Ing. Dr.techn. Gernot Kubin

Assessor/Examiner:
Assoc.Prof. Dipl.-Ing. Dr.mont. Franz Pernkopf

WS 2013

Statutory Declaration

I declare that I have authored this thesis independently, that I have not used other than the declared sources / resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

Place, Date

Signature

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen / Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

Ort, Datum

Unterschrift

Abstract

Keyword spotting (KWS) represents an important component of speech signal processing. This thesis aims to detect the occurrence of individual keywords in a continuous audio signal. In doing so, two approaches can be distinguished. Some approaches use phonetic or syllabic models for their speech models based on Hidden Markov Models. These algorithms require a specific model for each language. Because the KWS algorithm should be independent of the language or dialect used, another type of algorithm was selected, namely pattern matching. In this process, the keyword is recorded once and serves as reference (template) in the subsequent detection process. The algorithm compares the current audio data to the template on the basis of different distance measures such as dynamic time warping or euclidan distance, among others. A keyword is considered detected if the distance between the template and the current audio data, represented by mel frequency cepstral coeffiecients, is below a certain threshold. Additionally, a rhythm detection algorithm that detects handclapping is developed. Like in the keyword spotting method, a reference rhythm is recorded and serves as template in the subsequent detection process. Both algorithms are implemented in Matlab and tested on recorded databases with regard to their recognition and false alarm rate. The KWS database was recorded with three different background noise levels, four different distances between speaker and recording device and ten different speakers. It consists of 480 keywords. The KWS algorithm is evaluated on this database and has a recognition rate of 54.4% and a precision of 97.8%. Additionally, a performance curve, i.e. recall vs. precision, with different settings of the thresholds is presented. The rhythm database was recorded with two different background noise levels and four different distances between speaker and device. It consists of 256 rhythms. The rhythm detection algorithm evaluated on this database has a recognition rate of 79.7% and precision of 99.5%.

Kurzfassung

Signalwortdetektion (keyword spotting) stellt einen wichtigen Bereich der Sprachsignalverarbeitung dar. Dabei soll das Auftreten von einzelnen Signalwörtern in einem kontinuierlichen Audiosignal detektiert werden. Hierbei können zwei verschiedene Herangehensweisen unterschieden werden. Zum einen Algorithmen, welche auf Phonem-, Triphon- oder Silbenmodellen als grundlegendes Sprachmodell basieren mit anschließender Verarbeitung mittels Hidden Markov Modellen. Diese Algorithmen benötigen aber für jede Sprache ein eigenes Modell. Da der KWS Algorithmus unabhängig von der gewählten Sprache oder Dialekt sein soll, wurde ein anderer Weg gewählt, der des Referenzmuster-Vergleichs. Dazu wird das zu detektierende Signal einmal aufgenommen und dient anschließend als Referenz. Der Algorithmus vergleicht nun mittels verschiedener Methoden, wie etwa Dynamic Time Warping, Euklidische Distanz u.a., das aktuelle Audiosignal mit der Referenz. Ein Signalwort gilt nun dann als detektiert, wenn mittels der eingesetzten Methoden eine ausreichende Übereinstimmung zwischen Referenz und aktuellem Audiosignal, repräsentiert durch mel frequency cepstral coefficients, errechnet worden ist. Zusätzlich wird ein Algorithmus entwickelt, welcher den Rhythmus von Händeklatschen detektiert. Auch hier wird der gewünschte Rhythmus einmal mittels Klatschen aufgenommen und dient anschließend als Referenz. Beide Algorithmen wurden in Matlab implementiert und anhand von erstellten Datenbanken hinsichtlich ihrer Erkennungs- und Fehlalarmrate ausgewertet. Die KWS Datenbank wurde mit drei verschiedenen Lautstärken des Hintergrundgeräusches, vier verschiedenen Distanzen zwischen Sprecher und Aufnahmegerät und zehn verschiedenen Sprechern aufgenommen. Sie beinhaltet 480 mal das Signalwort. Der KWS Algorithmus erzielt auf dieser Datenbank eine Erkennungsrate von 54.4% und eine Genauigkeit von 97.8%. Zusätzlich werden Performance-Kurven, d.h. Erkennungsrate vs. Genauigkeit, mit verschiedenen Schwellwerten, aufgenommen. Die Rhythmus Datenbank wurde mit zwei verschiedenen Hintergrundgeräuschszenarien und vier verschiedenen Distanzen erstellt und beinhaltet 256 mal den Rhythmus. Der Rhythmus Erkennungs Algorithmus erreicht eine Erkennungsrate von 79.7% und eine Genauigkeit von 99.5%.

Danksagung

Ein großer Dank gilt meinen beiden Betreuern Franz Pernkopf und Martin Hagmüller für die Betreuung und Unterstützung bei dieser Arbeit sowie den Besprechungen, bei denen sie wertvolle Beiträge leisteten. Weiters danken möchte ich Franz Stieger für die Kooperation bei der Erstellung der Datenbanken und die Zusammenarbeit mit Claus Zotter. Ein großer Dank geht an meine Freundin Khulan, die bei der Simulation der Datenbanken sehr geholfen hat. Vielen Dank an Annika Flaake für das unglaublich rasche Gegenlesen dieser Arbeit. Zum Abschluss gilt mein größter Dank meinen Eltern, Regina und Josef, die mir dieses Studium und eine wunderschöne Zeit in Graz ermöglicht haben.

Graz, Oktober 2013 Andreas Zehetner

Contents

List of Abbreviations

AAA	Average Absolute Amplitude
AAAD	Average Absolute Amplitude Difference
AAAR	Average Absolute Amplitude Ratio
ASR	Automatic Speech Recognition
AZCR	Amendatory Zero-Crossing Rate
BNC	Background Noise Classification
CC	Cross-Correlation
DCT	Discrete Cosine Transform
DFT	Discrete Fourier Transform
DTW	Dynamic Time Warping
ED	Euclidian Distance
FFT	Fast Fourier Transform
FIR	Finite Impulse Response
FN	False Negatives
FP	False Positives
FT	Fourier Transform
GUI	Graphical User Interface
HIST	Histogram of Amplitudes
HMM	Hidden Markov Model
KWS	Keyword Spotting
LPC	Linear Predictive Coding
MA	Moving Average
MFCC	Mel Frequency Cepstral Coefficients
NAD	Noise Activity Detection
ORC	Other Room with Closed door
ORO	Other Room with Open door

P ..... Precision
PLP ..... Perceptual Linear Predictive
R ..... Recall
SNR ..... Signal to Noise Ratio
TP ..... True Positives
VAD ..... Voice Activity Detection
VP ..... Voice Preemphasising
ZCR ..... Zero-Crossing Rate

1 Introduction

Automatic speech recognition (ASR) is the ability of a computer to convert an audio speech signal, captured by a microphone, to a set of words. One goal is to allow the interaction between humans and computers using speech. ASR is done in a vast number of applications including voice user interfaces such as voice dialling, call routing or speech-to-text processing. The motivation behind ASR is to improve the human-computer interaction or to build up systems that can process spoken language as efficiently as humans. Since the number of mobile devices like smartphones, tablets or laptops is increasing for years, more and more effort is put into development of ASR systems for these applications [1], [2].

The advantages of using speech instead of a graphical user interface (GUI) such as a mouse or a keyboard are numerous. Speech is natural for humans and needs no training. Specially elderly people have problems to tag along with new technologies. Moreover interacting with a computer using speech can be faster and more intuitive. Another advantage is that the user does not have to be in front of the device. The user can control the device such as a smartphone or a laptop from far distance using ASR [3].

An important branch of ASR systems is keyword spotting (KWS). KWS can be found in a variety of different applications that do not require the knowledge of the whole content of an utterance such as the detection of a keyword in conversational speech monitoring, i.e.

surveillance applications. Other examples are telecommunications services, i.e. automatic operator services, or voice command detection. Different methods for KWS were developd in the last decades. Although KWS is not novel, there are still many open research questions.

1.1 Keyword Spotting Methods

KWS can be defined as the task of identifying the occurrence of certain keywords in an arbitrary speech signal. While in speech recognition an occasional unknown word punctuates a stream of known words, in KWS a typically small number of keywords is to be recognized among a large number of words. KWS approaches try to detect these specific keywords within other words, sounds and noises without modelling the non-keywords. Therefore, some KWS approaches use individual models for the keywords and filler or garbage models for non-keywords. As depicted in Figure 1.1, these models are usually based on hidden Markov models (HMM), as can be seen for example in [4], [5] and [6]. If the vocabulary is small, the keyword models are typically based on word models. If there are many keywords as for instance in task independent models, the keyword models are based on subword models such as syllabic or phonetic models. Since the filler models aim to cover a wide variety of non-keywords, they are usually based on phonetic or syllabic models [7].

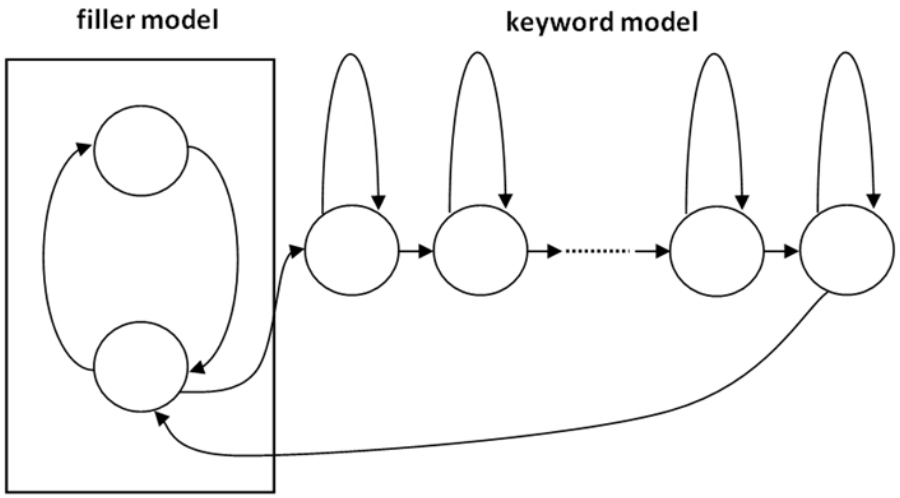


Figure 1.1: *KWS based on HMMs using keyword and filler models.*

Some KWS approaches try to determine the boundaries of the keywords, i.e. the beginning and the ending of the keyword, in a continuous audio stream in a first step. One way to do so is to build up a codebook. Each entry of the codebook should represent an acoustic characteristic of the signal. Therefore a vector quantization (VQ) is performed

over a training database to construct the codebook. A keyword is defined as a specific conjunction of these codebook entries, i.e. an ordered set of acoustic characteristics. The detection of the keyword boundaries is performed by the detection of these characteristics. Once the boundaries of the keyword are detected using VQ, a recognition process based on HMMs begins. Therefore, the acoustic representation returned from the VQ stage is aligned with subword units represented by HMMs representing the given keyword. If the probability of the keyword represented by HMMs exceeds a specific threshold an alarm is triggered [8].

Approaches based on HMMs are widely used in KWS and are accurate for a variety of tasks. However, HMMs are computational expensive and language specific training data is necessary to train the parameters of the model. An alternative approach for KWS is presented in [9]. The introduced method extend the notion of discriminative large margin and kernel methods to the task of KWS.

Another strategy is to analyze and search through the word lattice of a speech recognizer to detect the occurences of a keyword [10] [11]. Thereby the audio signal gets indexed using speech recognition to generate a phonetic representation of each audio file, i.e. a lattice of phoneme hypotheses. KWS means now rapidly locating all sub-paths in the lattice set that match the representation of the query string. If the confidence between the query string and the template exceeds a certain threshold, a keyword is detected. These systems rely on speech recognizers and suffer from high error rates, especially when the speech is not clean [11], [12].

However, all these methods usually need some representative amount of training data to train the parameters. To overcome this drawback, other techniques such as dynamic time warping (DTW) have been used [13]. DTW finds an optimal alignment between two time-dependent sequences, i.e. the reference and the test sequence. Therefore it is a widely used method for pattern matching [14]. The KWS algorithm presented in this thesis should be independent of the language or dialect used for KWS. Also the keyword should be selected by the user. Therefore HMMs are not optimal, since they require a specific model for each different language. On the other hand DTW is a simple pattern matching algorithm. For this reason, we execute DTW, among other methods, for KWS in this thesis.

1.2 App 112 - Motivation and Problem Statement

A new safety solution is going to be introduced by a German company. It consists of an app for a smartphone. The aim of the app is to recognize a specific keyword said by a specific person. In case of emergency, one can say this personalized keyword. If the keyword is detected by the KWS algorithm an alarm is triggerd and the app calls for help, for example calls a relative. Such systems are useful for elderly people and institutions such as the Red Cross. To set up this app, the user needs to record a personalized keyword once. This recording serves as reference token and is identified afterwards in the continuous audio recording via pattern matching. Therefore, an algorithm which detects this specific keyword in continuously uttered speech and in different situations without learning and with a single reference recording has to be developed. Moreover, an algorithm that detects a specific rhythm clapped by the user in case of emergency has been developed. The basic set-up is similar to KWS. One needs to clap and record the personalized rhythm. In daily life the algorithm detects this rhythm in the continuous audio signal.

Figure 1.2 shows the GUI of the so called App112. By pressing the red button with the microphone sign in the middle, the detection is activated and the algorithm starts to monitor the continuous audio stream for spotting the keyword/rhythm. In case one says the keyword/claps the rhythm, the algorithm detects it and an alarm is raised. The SOS button in the bottom right corner will activate the alarm immediately without any KWS or rhythm detection.

1.3 Requirements of the App112

The KWS algorithm is developed for a private company. This company already introduced other speech processing apps for smartphones or tablets. Hence, a basic framework exists. Therefore, there are some requirements or constraints from the company which had to be considered in the development process.

Speech model The private company wants to sell the App112 in different countries with different languages. Hence, the App112 should be able to process without adaptation to different languages. So the simplest approach is to work with keyword templates. In case of HMMs this is more difficult.

10s blocks The processing of the audio signal in 10s blocks and the classification of these 10s blocks with background noise classification (BNC) is a requirement, since the company introduced already other speech processing apps and these apps process with a 10s blocking and 2s overlap.

Figure 1.2: *GUI of the App112.*

MFCC The used framework of the private company already introduced MFCC with a frame- and hop-size of 32ms and 16ms, respectively.

Weighted performance The weighting of recall and precision of 1 : 3 to calculate the performance is determined by the private company, since false alarms are costly and should be avoided.

Databases Since the KWS algorithm should be tested on different keywords, own databases are recorded.

1.4 Organization and Aim of this Work

In this thesis, a KWS and a rhythm detection algorithm is presented. Since the algorithms have to run on a smartphone, emphasis is put on real-time processing and low energy consumption. Moreover the algorithms are used in the App112, which should be usable in different countries. Therefore, the KWS algorithm should work independently of the used language.

The thesis is organized in two main parts. In the first part the theory for the KWS and rhythm detection and the implementation is presented. This part consists of Chapter 2, 3 and 4. Chapter 2 introduces the overall system design of the App112. The system consists of different methods for KWS, BNC and quality measures for the reference

recording. These methods are introduced in Chapter 3. Moreover the combination of the single methods for KWS is described in this chapter. Chapter 4 presents the method and its implementation used for the rhythm detection algorithm, namely the average absolute amplitude ratio (AAAR).

The second block covers the experimental results of both algorithms. In Chapter 5 the used databases for the empirical evaluation of the KWS algorithm and the rhythm detection algorithm are presented. Moreover this chapter provides an introduction to the evaluation of both algorithms via recall and precision. Finally, performance results are presented. The algorithms are evaluated depending on different background noise scenarios and different distances between user and microphone. Moreover, the performance of the KWS algorithm is determined depending on the gender of the speaker and different keywords.

2
App112 - System Design

Figure 2.1 shows the overall system of the App112 for KWS and rhythm detection. Basically it consists of three main parts, namely, reference recording, KWS and rhythm detection. When launching App112, one needs to record a personalized keyword first (e.g. "Help emergency!"). After that, the quality of the recorded keyword and of the audio recording is measured automatically. Therefore, the following methods are used:

- average absolute amplitude difference (AAAD)
- noise activity detection (NAD)
- amendatory zero-crossing rate (AZCR)

Details about measuring the quality are provided in Section 3.1. If the quality is too bad due to background noise or the keyword does not meet certain requirements, one needs to repeat the reference recording until the quality is satisfactory. If the quality check is successful the reference is played to the user and the user has to confirm the reference for KWS. Then the audio recording of the keyword is saved and is used as reference for the KWS algorithm.

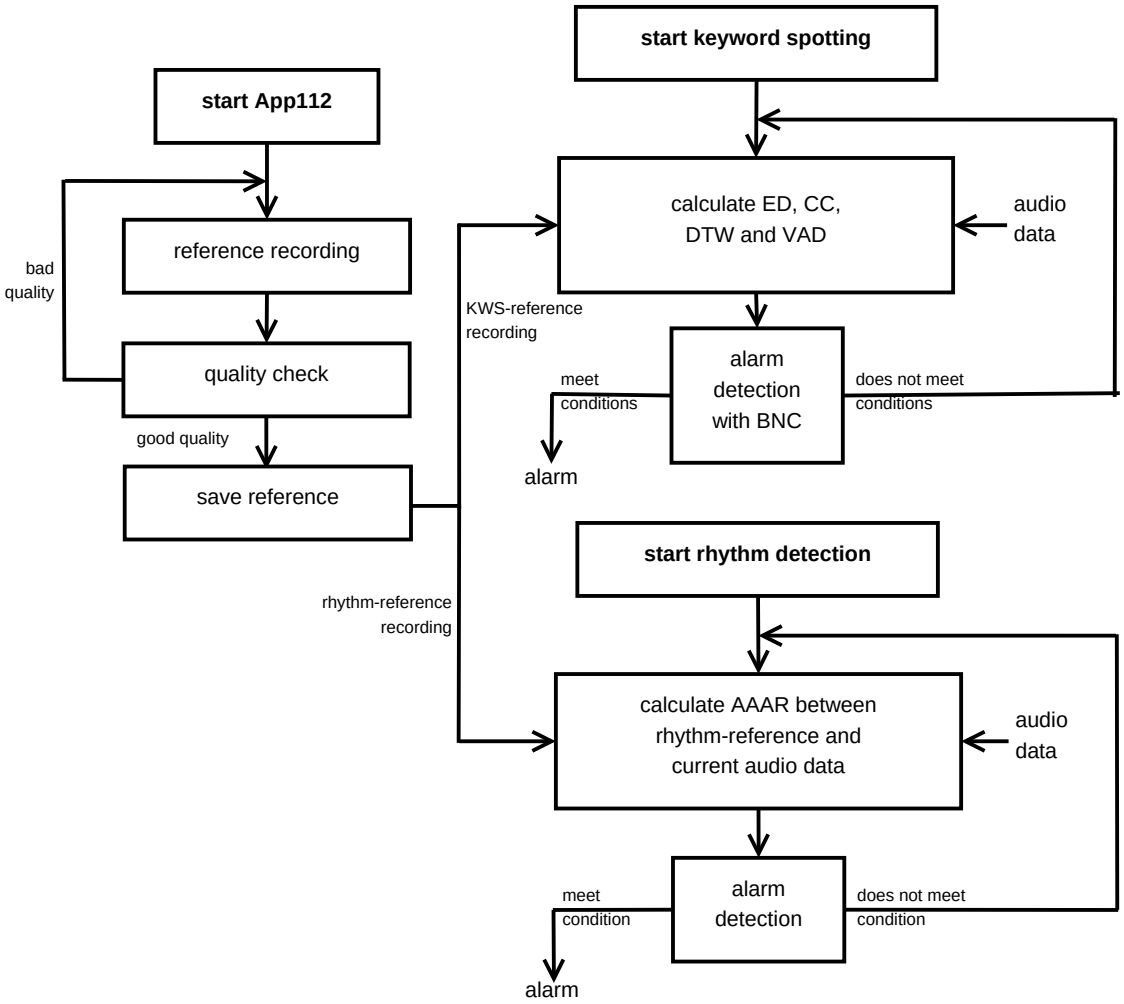


Figure 2.1: *Overall system of the App112.*

The KWS algorithm calculates the following three different criteria for measuring the distance between the recorded template and the current audio data:

- euclidian distance (ED)
- cross-correlation (CC)
- dynamic time warping (DTW)

Moreover a voice activity detection (VAD) is implemented to reduce the number of false alarms. We discovered that in different situations a different combination of these methods improves the performance. Therefore, we use BNC to classify the background noise into the three categories silence, noise or high noise. Depending on the classification a different combination of the criterias used for KWS is taken for the final limit check.

In case the used combination of the criteria meet certain requirements, i.e. exceed/is below a threshold, an alarm is triggered. Otherwise, the algorithm processes the next chunk of audio data. More details are in Chapter 3. However, only audio data chunks

with sufficient average amplitude level are analyzed. Therefore, the average absolute amplitude (AAA) of the audio data is calculated. If this AAA exceeds a threshold, the KWS is started, otherwise the App112 remains in a sleeping mode without KWS.

The rhythm detection algorithm works in a similar way. After recording a reference, i.e. clapping a certain rhythm, the recording of the rhythm is saved and used for the rhythm detection algorithm. The quality check procedure is similar as above. The rhythm detection algorithm calculates the AAAR of the current audio signal and compares it to the AAAR of the recorded rhythm reference. If the difference between both AAARs is below a certain threshold, an alarm is triggered. Otherwise, the algorithm processes the next chunk of audio data. Again, the AAA of the audio signal has to be above a certain level. Otherwise the AAAR is not calculated. More details for the rhythm detection are provided in Chapter 4.

3
Keyword Spotting Algorithm

This chapter shows the implementation of the KWS algorithm. First, the quality measures for the reference recording get introduced. If the reference recording satisfies the quality criterias, KWS is activated. Therefore, ED, CC, DTW and VAD are introduced. Depending on the BNC a different combination of these methods is taken for the final detection process.

3.1 Quality Check of Reference Recording

To set up the app, one needs to record a personalized keyword in a first step. This specifies the reference for KWS in the continuous audio record. To ensure a good recognition performance, the reference, i.e. the template, is of essential importance. On the one hand, the quality of the audio recording is crucial. Therefore, AAAD is calculated to detect possible transient noise sources such as the slamming of a door or a window. NAD is calculated to detect stationary noise such as a fan or traffic noise. On the other hand, the keyword itself is important. To ensure that the keyword has sufficient phonetic structure, AZCR is calculated.

3.1.1 AAAD

Figure 3.1 depicts the steps for calculating the AAAD. First, the AAA of the audio signal $x[n]$ is calculated, i.e. sum up the absolute value of the audio signal over a frame-size of 25ms. The hop-size is 5ms. The AAA is

$$AAA[n] = \frac{1}{N} \sum_{i=(n-1)HS+1}^{(n-1)HS+N} |x[i]|, \quad 1 \leq n \leq M, \tag{3.1}$$

where f_s is the sampling frequency, $N = 0.025 f_s$ is the number of samples per frame, $HS = 0.05 f_s$ is the hop-size in samples, and $M = \frac{SL-N}{HS} + 1$, with SL as the total number of samples in the utterance. In the following step, the absolute differences between adjacent absolute amplitude values are determined

$$AAAD^*[n] = |AAA[n+1] - AAA[n]|, \quad 1 \leq n < M. \tag{3.2}$$

Thereby transient signals become easily detectable. As the absolute amplitude of transients is changing rapidly, the difference between adjacent AAA values increases. In contrast, continuous signals such as speech have small changes in the AAA and therefore the differences are small. Finally, these differences are accumulated over five consecutive frames with step-size of three. The result is a vector consisting of the AAAD

$$AAAD[m] = \sum_{i=(m-1)SS+1}^{(m-1)SS+FS} AAAD^*[i], \quad 1 \leq m \leq L, \tag{3.3}$$

where $SS = 3$, $FS = 5$ and $L = \frac{M-FS}{SS}$.

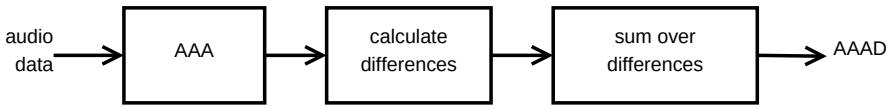


Figure 3.1: *Flow chart of the average absolute amplitude difference.*

Figure 3.2 shows the different stages of the AAAD. Figure 3.2 left column shows a signal without transient noise and in the right column a utterance with transient noise, i.e. the knocking of a pen onto a table, is presented. Figure 3.2 (a) shows the PCM audio signal of both signals, whereas in (b) the AAA of the audio signals are depicted. In (c) the differences between consecutive AAA values are presented. Finally, in (d) the accumulated differences AAAD are presented. While for the signal without transient noise the accumulated differences are roughly the same everywhere and do not have a peak, the AAAD for the utterance including a transient shows a significant peak at the

position where the transient is located. The algorithm uses a threshold, 0.4 here, for the accumulated differences to detect transients in the background. If transients are detected, the user needs to repeat the recording. Otherwise the other quality measures are calculated.

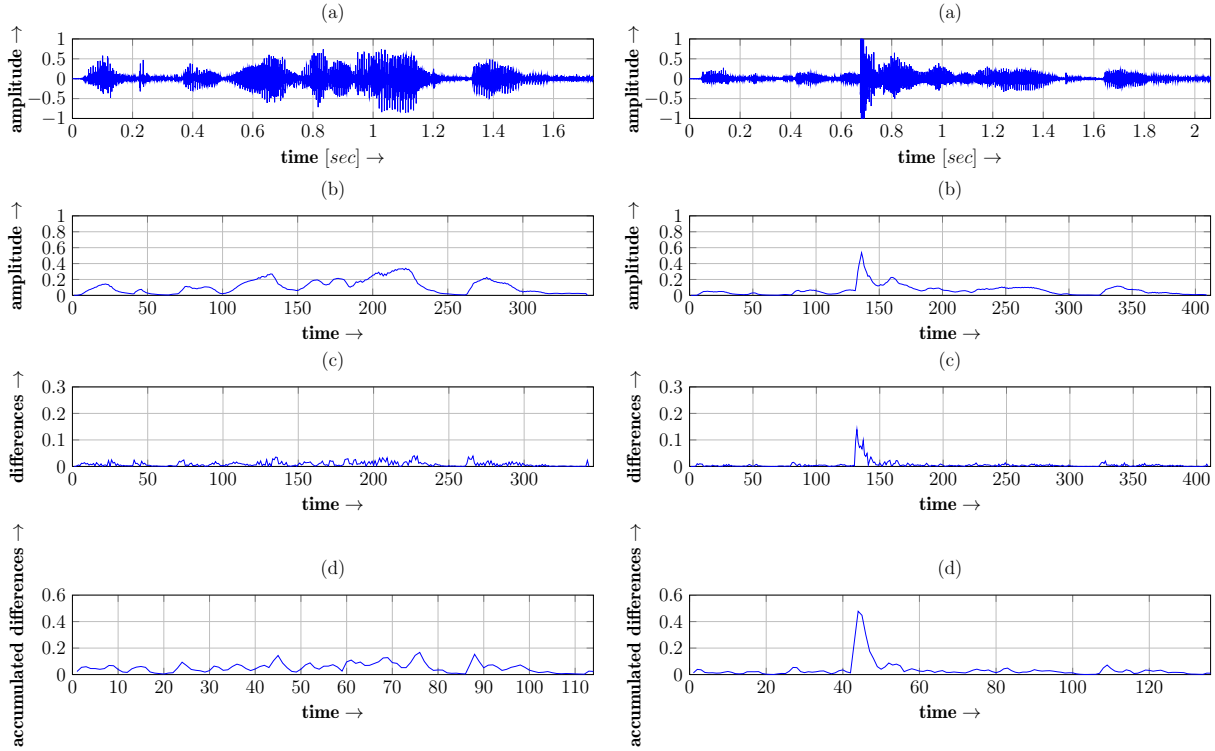


Figure 3.2: *AAAD for quality measure. Left column shows a signal without transient noise and right column shows a signal including transient noise. (a) PCM audio signal. (b) AAA of the audio signal. Frame-size is 25ms and hop-size is 5ms. (c) AAAD of consecutive AAA values. (d) Accumulated AAAD. Frame-size is 5 and hop-size is 3.*

3.1.2 NAD

To detect stationary noise in the reference recording NAD is implemented. Therefore the AAA directly before and after the keyword in the reference recording is calculated. Since the reference recording has to take place in a quiet environment, the AAA is expected to be very low. If the AAA exceeds a certain threshold, there is background noise in the recording and the user has to repeat the recording in a more quiet place.

For the reference recording the user has 5s to record the keyword. Afterwards the frontiers of the keyword are determined within these 5s by calculating the AAA. The starting point is defined as the first point, where the AAA exceeds a certain threshold with an additional safety gap of 50ms and the ending point is defined as the last point,

where the AAA exceeds this threshold with an additional safety gap of 160ms. Only the recording within these boundaries is saved as template for the KWS. To determine possible background noise in the reference recording, the signal before the starting point and after the ending point is analyzed. If the AAA of these parts of the recording exceeds a threshold, there is background noise and the speaker has to repeat the reference recording. Figure 3.3 shows a reference recording of 5s in length. The blue line represents the audio signal and the green line the AAA. The starting point of the keyword is the first point, where the AAA exceeds the threshold of 0.05, marked in red colour in the plot, minus a safety gap of 50ms. The ending point of the keyword is the last point where the AAA exceeds this threshold plus a safety gap of 160ms. NAD is averaging the AAA before the

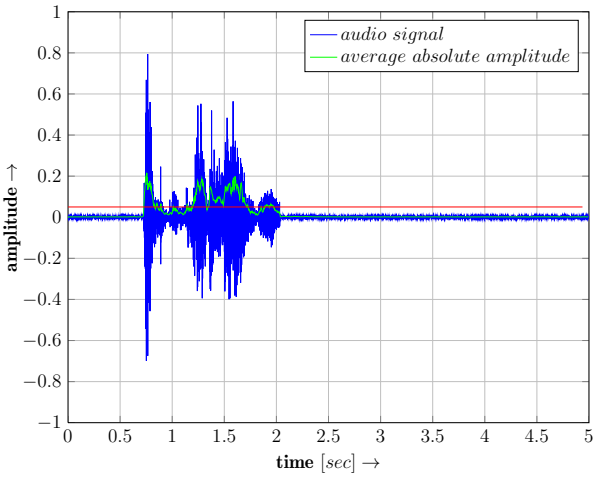


Figure 3.3: *NAD for reference recording.*

starting point and after the ending point. If the NAD exceeds a threshold, the user has to repeat the reference recording, otherwise the other quality measures are calculated. In case the NAD of the audio signal is always above the threshold, i.e. starting point and ending point are 0s and 5s, respectively, the noise is too large and the recording has to be repeated.

3.1.3 AZCR

The zero-crossing rate (ZCR) denotes the frequency of a signal crossing zero in a frame. As noise has a lower ZCR than voiceless but higher ZCR than voiced speech, this criterion is vulnerable to low-frequency noise. To find a remedy of this drawback, we use the AZCR for the quality check of the reference recording. AZCR is the rate of the signal crossing a threshold T and $-T$ per frame. This reduces the ZCR of noise in speech pauses and therefore avoids unreliable ZCR estimates. Figure 3.4 shows the AZCR with threshold T.

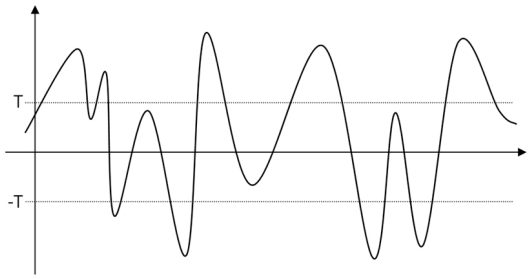


Figure 3.4: *Amendatory zero-crossing rate.*

AZCR is computed as

$$z[n] = \frac{1}{2} \sum_{i=-\infty}^{\infty} \{|sgn\,(x[i] - T) - sgn\,(x[i-1] - T)\,| + |sgn\,(x[i] + T) - sgn\,(x[i-1] + T)\,|\}w[n-i], \tag{3.4}$$

where $n = 1, 2, ..., N$, $w(\cdot)$ is a rectangular window and $sgn(\cdot)$ is the sign function [15]. We selected a window length for $w(\cdot)$ of 10ms.

The AZCR of the selected keyword, i.e. the sequence of vocals and plosives has to be sufficiently large. The idea of this quality measure is to disable the use of improper keywords, i.e. the use of a keyword with just one vocal and no consonants like "*Ah*". Such keywords would result in a high false alarm rate. Therefore the recorded template requires some phonetic content, which we measure by the AZCR.

Figure 3.5 shows ZCR and AZCR for two audio recordings. On the left hand side the speech sequence "*a-ma-ma*" and on the right hand side "*a-ka-ka*" is depicted. Figure 3.5 (a) shows both PCM audio signals. The ZCR of both sequences is presented in (b) whereas (c) shows the AZCR. The difference between ZCR and AZCR can be seen at the beginning and at the end of the audio recording, i.e. the parts in the recording where there is silence and no speech. As ZCR measures the signal's zero crossings, it measures the ZCR of white noise at silent parts, which provides a flowed impression of the audio signal. Therefore AZCR is implemented, so the zero crossings at silent parts are zero.

While "*a-ma-ma*" has no significant peaks in AZCR because all parts of the sequence are voiced, "*a-ka-ka*" shows evident peaks at the plosives. Immediately prior to the plosives, there is a short region where the AZCR becomes zero because there always is a short pause in speech before plosives. Thus, AZCR is a useful measure to measure to some extent the phonetic content of the keyword. Currently, the keyword requires to have at least two plosives, i.e. AZCR needs to exceed 40 at two different points with at least 0.2s time difference between them.

If the recorded keyword fulfills this requirement as well as the AAAD and NAD quality check, it can be checked by the user. In case of acceptance it is saved and used afterwards

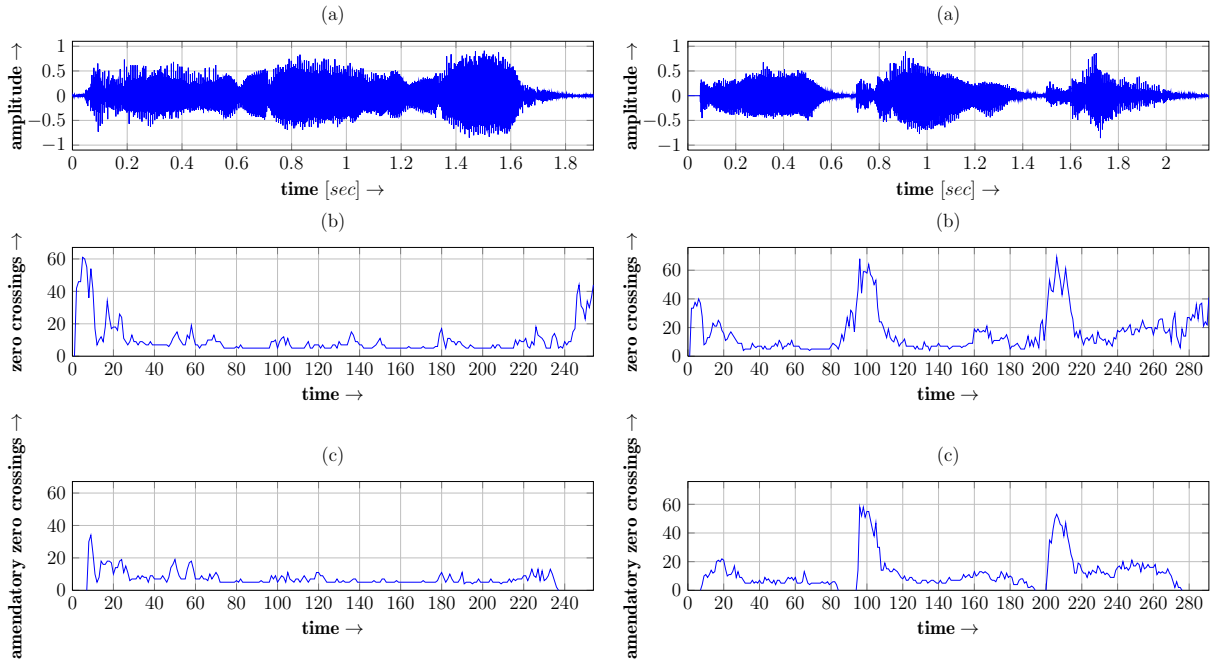


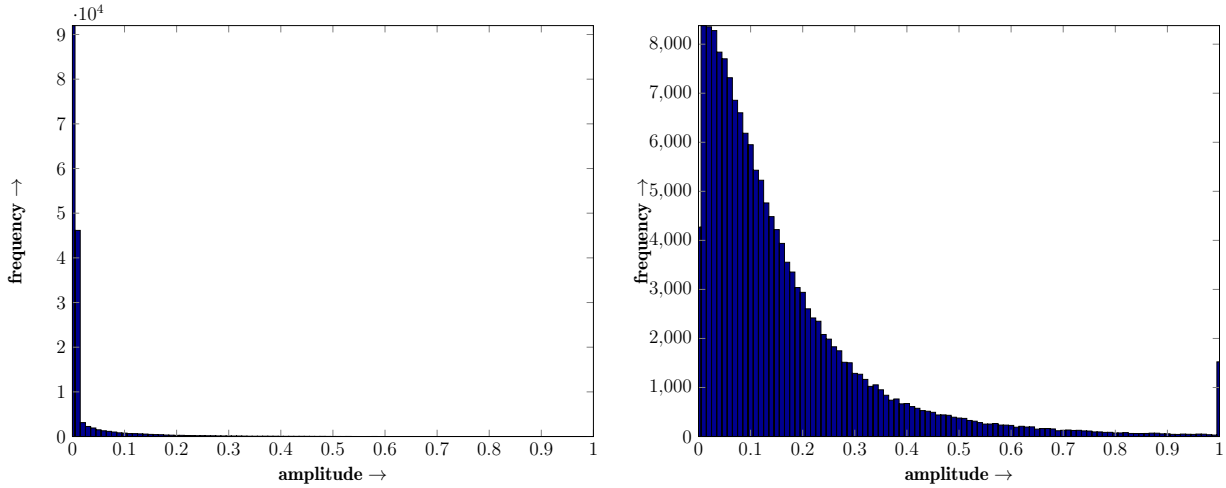
Figure 3.5: *ZCR and AZCR for quality measure. Left column represents (a) PCM audio signal of* "a-ma-ma"*. (b) ZCR of the audio signal. (c) AZCR of the audio signal. Threshold T is 0.05. Right column shows the same for speech signal* "a-ka-ka"*.*

as a template in the KWS algorithm. Otherwise the speaker needs to repeat the reference recording.

3.2 Background Noise Classification

During the process of KWS, the audio signal is processed in frames of 10s in length at a sampling rate of $f_s = 16kHz$. Each frame is classified into one of the following three categories: silence, noise or high noise. Depending on the category of the audio frame a different combination of ED, CC, DTW and VAD is used to detect the keyword. The classifier is based on the histogram of amplitudes (HIST) in the 10s frame.

For each of the 10s blocks a histogram is calculated, i.e. the distribution of the amplitudes is analyzed. Therefore, the absolute value of the amplitudes are classified into one of 100 uniformly distributed bins from 0 to 1. The result is a vector $HIST[n]$ with $n = 1, 2..., 100$ containing the frequency of observations of the different amplitudes. Figure 3.6 shows the resulting histograms of silence and high noise audio signals. Most of the data points have small amplitudes in silence as can be seen in Figure 3.6(a), hence the distribution showing in the histogram is concentrated at the first few bins. In contrast, in high noise the distribution of the amplitudes changes to higher values, see Figure 3.6(b).



(a) silence (b) high noise

Figure 3.6: *Histogram of amplitudes of silence and high noise.*

Thus, to classify the 10s audio signal the frequency count in the first bin is divided by the total number of audio samples in the 10s block, i.e. the probability of the first percentile $p_1(x)$ is used

$$p_1(x) = \frac{\#\ samples\ x[n] < 0.01}{total\ \#\ samples} = \frac{\#\ samples\ x[n] < 0.01}{160000}. \tag{3.5}$$

Depending on $p_1(x)$, the block is classified to:

silence: $if \quad 0.80 \leq p_1(x)$
noise: $if \quad 0.45 \leq p_1(x) < 0.80$
high noise: $if \quad p_1(x) < 0.45$

Figure 3.7 shows the $p_1(x)$ values of the single audio files of the KWS database.

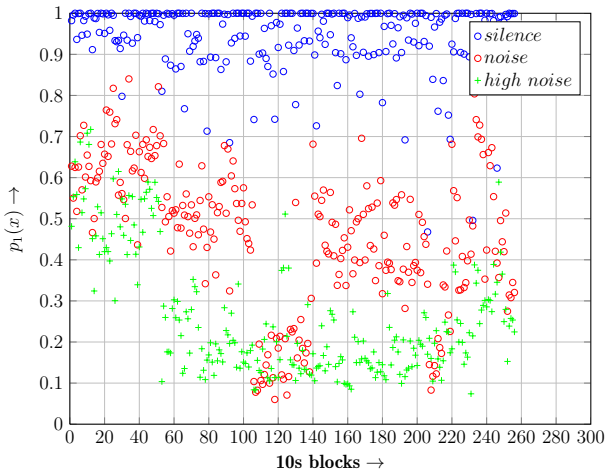


Figure 3.7: $p_1(x)$ *of the database used for KWS.*

3.3 Keyword Spotting Algorithm

The energy of the audio signal is calculated continuously in the energy detector in Figure 3.8. If the energy is below a threshold, no KWS is processed. If the energy exceeds this threshold, the continuous audio signal is blocked into frames of 10s in length with 2s overlap. Each of these frames is processed as depicted in Figure 3.8.

First, the digitized audio data is voice preemphasised (VP) to flatten the spectrum of the audio signal. This is done by filtering the signal with a first-order FIR filter. Equation (3.6) shows the filter, the value of a is 0.95 [16]

$$H(z) = 1 - az^{-1}. \tag{3.6}$$

The keyword spotting is performed via pattern matching of the current audio signal and the reference recording in the cepstral domain. Therefore the Mel Frequency Cepstral Coefficients (MFCC) are calculated for the current audio signal and the reference recording. Empirically we observed that 17 MFCC features are sufficient for good performance. The frame-length for calculating the MFCC is 32ms with 16ms overlap, as required (see Section 1.3). The n^{th} signal frame is represented by a MFCC feature vector x_n. Now ED, CC and DTW distances between the template and the current audio signal are determined and normed by dividing by moving average (MA) filter. This results in scaled ED, CC and DTW values. To reduce false alarms a voice activity detection (VAD) is introduced which classifies each frame into speech or non-speech. At the same time, the background noise is classified via HIST. Depending on the output of the classification, i.e. silence, noise or high noise, a different combination of the scaled criteria ED, CC or DTW and VAD is taken for alarm detection. This combination depends on the BNC using HIST. Therefore, the distance measures are thresholded and combined by Boolean operator.

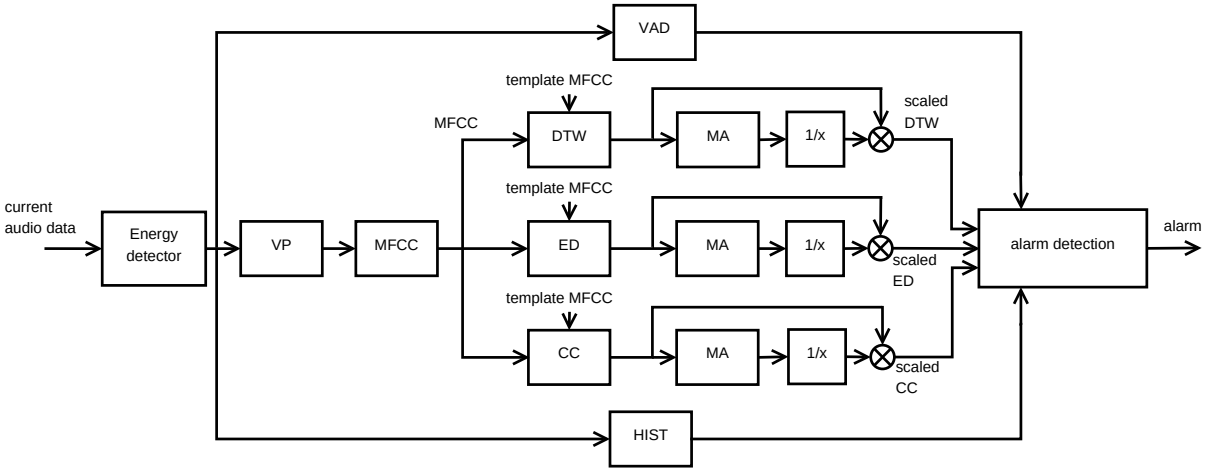


Figure 3.8: *Overall system for KWS.*

3.3.1 Mel Frequency Cepstral Coefficients

Calculating MFCCs is a common technique of feature extraction in speech recognition systems. In this thesis MFCCs are selected because they are less prone to noise and resembles the human auditory system, since human hearing is based on frequency analysis [16]. Human speech can be described by a so-called source-filter model. This model consists of two components:

- the *excitation source* is related to the glottis and is responsible for the fundamental frequency in voiced speech or provides white noise in unvoiced or mixed speech,
- the *filter* is related to the (time-varying) vocal tract consisting of mouth, tongue and nasal cavity. The vocal tract acts as a filter and shapes the excitation.

A frequency domain representation of a speech signal $x[n]$ is depicted in Figure 3.9. The bold line represents the spectral envelope due to filtering with the vocal tract. F_1, F_2 and F_3 are the formant frequencies of human speech. The position of these formants is an important cue for the human auditory system as well as for ASR.

F_0 is the fundamental frequency or spectral fine structure, i.e. the spectral harmonics, due to the glottis, i.e. the source excitation. F_0 is less important for speech recognition than the formant frequencies, since it only determines the pitch. The idea is to separate the source component from the filter component, since for speech recognition only the filter characteristic is of interest. One approach is to calculate MFCC [17].

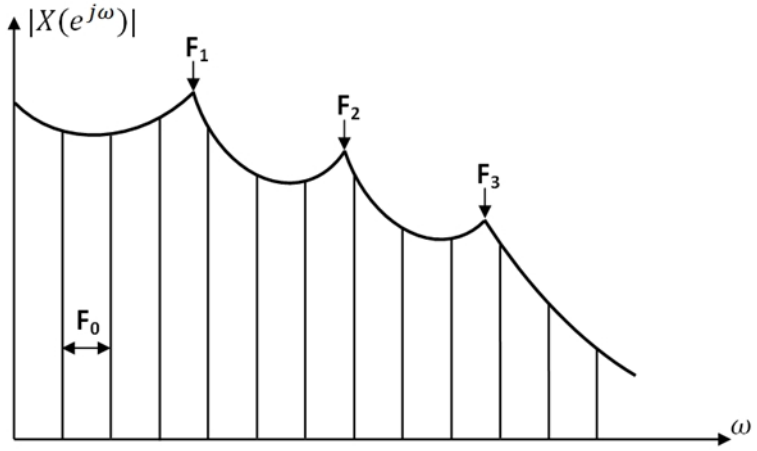


Figure 3.9: *Source-Filter model of speech signals.*

Figure 3.10 shows the calculation process for MFCCs. In a first step, the audio signal is divided into blocks of usually 20-30ms in length. Within this time period, speech can be assumed as stationary. There is a trade-off between time and frequency resolution. The longer the blocks, the better the frequency resolution. On the other hand, the blocks

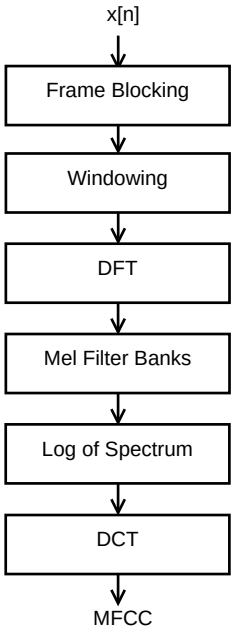


Figure 3.10: *MFCC calculation.*

should be short enough to capture the local spectra properties. To minimize edge effects caused by discontinuities in the signal at the beginning and ending of the blocks, each block is windowed, i.e. the time signal $x[n]$ is multiplied by a window function $w[n]$. This is usually done with a hamming window, see Equation (3.7) and Figure 3.11

$$w[n] = 0.54 - 0.46 \cos\left(\frac{2\pi n}{L-1}\right), \quad 0 \leq n < L. \tag{3.7}$$

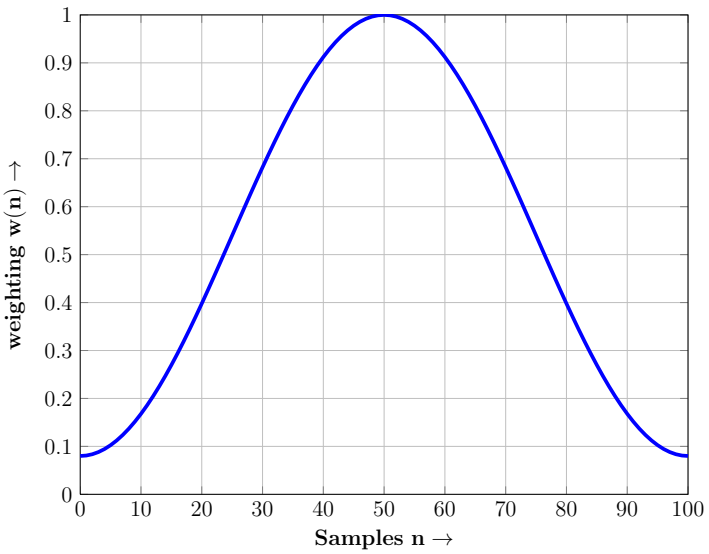


Figure 3.11: *Hamming window for MFCC calculation.*

After framing and windowing the time sequence, $s[n]$ is transformed to the frequency domain via discrete Fourier transform (DFT) efficiently implemented as fast Fourier transform (FFT), i.e.

$$S[k] = \sum_{n=0}^{K-1} s[n] e^{-2\pi j \frac{kn}{K}}, \quad k = 0, 1, ..., K-1, \tag{3.8}$$

where $S[k]$ is the K-point FFT spectrum of one windowed speech frame. Now the mel filter bank consisting of M triangular shaped band pass filters is applied to the spectrum, so that perceptual important frequency regions are merged together

$$X[m] = \sum_{k=0}^{\frac{K}{2}-1} |S[k]|^2 |H_m[k]|, \quad 1 \leq m \leq M, \tag{3.9}$$

where $|H_m[k]|$ denotes the frequency magnitude response of the m^{th} mel filter. This reduces the number of frequency bins. Here the number of summation points is just $\frac{K}{2}$ since half of the FFT spectrum is a mirror image of the other half.
Afterwards the so-called mel spectrum is compressed by a logarithmic function

$$X_{ln}[m] = \ln(X[m]). \tag{3.10}$$

Finally, the output of the logarithmic function is decorrelated using a Discrete Cosine Transform (DCT), i.e. converting the log mel spectrum back to time. The first few coefficients of this last stage are grouped together as a feature vector. The k^{th} MFCC can be expressed as

$$MFCC[k] = \sqrt{\frac{2}{M}} \sum_{m=1}^{M} X_{ln}[m] \cos\left(\frac{\pi k(m-0.5)}{M}\right), \quad 1 \leq k \leq p, \tag{3.11}$$

where M is the number of triangular band pass filters for the mel filter bank and p is the order of the mel scale cepstrum, typically between 12 and 20 coefficients [16], [17].

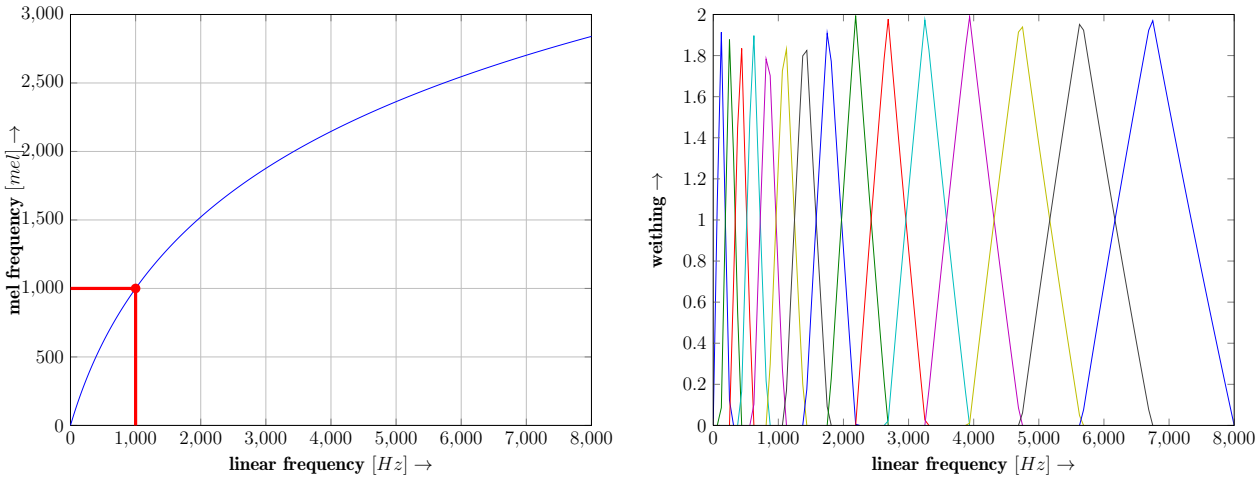
Mel Filter Bank

To calculate MFCCs, a mel filter bank is needed. This filter bank mimics the auditory system and is determined by listening tests. Therefore the proband has to judge the ratio between two tones with different pitches. Doubling or halving of the perceptual pitch means doubling or halving of the new frequency scale called mel scale (because this scale is based on the melody). The mel scale is a non-linear frequency scale, which has

approximately linear frequency spacing below 1000Hz and a logarithmic spacing above 1000Hz as shown in Figure 3.12(a). The conversion from linear frequency scale f in Hz to mel scale is done according to the following equation

$$mel(f) = 2595 \log\left(1 + \frac{f}{700}\right). \tag{3.12}$$

The filter bank is now realized with a series of triangular band pass filters as depicted in Figure 3.12(b). The filters overlap in such a way that the lower boundary of one filter is situated at the centre frequency of the previous filter and the upper boundary is located at the centre frequency of the next filter. The maximum response of the filter, i.e. the top vertex of a filter, equals the centre frequency of the filter.



(a) mel vs. linear frequency

(b) mel filter bank

Figure 3.12: *MFCC. (a) Relationship between linear and mel frequency. (b) Triangular band pass filters for MFCC calculation.*

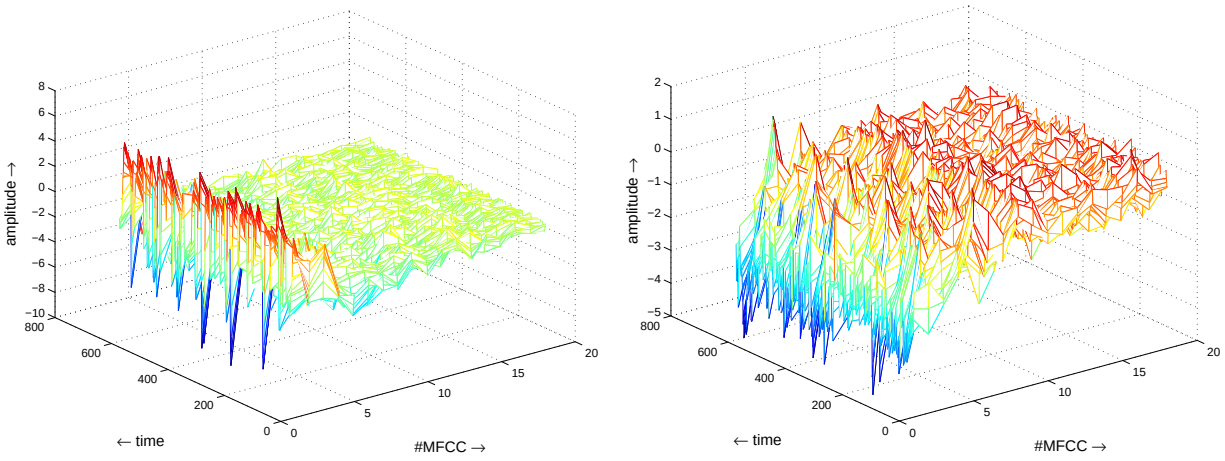
If f_L and f_H, in mel, are the lower and upper ends of the frequency range covered by the entire filter bank, the centre frequency f_{Cm} in mel of the m^{th} filter can be determined by

$$f_{\mathrm{Cm}} = f_L + \frac{m(f_H - f_L)}{M+1}, \quad 1 \leq m \leq M, \tag{3.13}$$

where M is the number of triangular filters in the range between f_L and f_H [16], [18].

3.3.2 VAD

To optimize the performance of the KWS algorithm while increasing the computational complexity just slightly, a VAD was implemented. In speech most of the energy is in low frequency regions while noise has also energy in higher frequency regions. For example white noise has a flat power spectral density. The introduced VAD uses this property to distinguish between speech and noise. Figure 3.13(a) represents MFCCs for speech and Figure 3.13(b) for noise. It can be seen that most of the energy in Figure 3.13(a) is located at low frequencies, i.e. low MFCCs, while Figure 3.13(b) has a roughly flat energy distribution.



(a) speech (b) noise

Figure 3.13: *VAD for KWS. (a) In a speech audio signal most energy is located in lower energy regions. (b) Noisy signals also have energy in higher frequency regions.*

We assume a time signal $x[n]$ which is represented in the cepstral domain. The result is a matrix $\mathbf{X}[m, c]$ where m relates to the time in samples and c relates to the number of MFCCs. The VAD algorithm calculates the magnitude ratio between high and low frequencies to distinguish between speech and noise. Therefore, the first L MFCCs are summed up over a time period T in samples and the following L MFCCs as well and the ratio is

$$ratio_{\text{VAD}} = \frac{\sum_{m=1}^{T} \sum_{c=L+1}^{2L} \mathbf{X}[m, c]}{\sum_{m=1}^{T} \sum_{c=1}^{L} \mathbf{X}[m, c]}. \tag{3.14}$$

If $ratio_{\text{VAD}}$ is smaller than a certain threshold, the VAD classifies the frame as speech, otherwise as noise.

3.3.3 DTW

DTW is a technique used to find an optimal alignment between two time-dependent signals. These signals are warped in a non-linear fashion to match each other. Figure 3.14 represents two time-dependet sequences $x[n]$ and $y[n]$. The two sequences are similar, although $y[n]$ has little changes in amplitude and time compared to $x[n]$. Intuitively, DTW tries to warp $y[n]$ in such a way that the alignment points, marked with an arrow, become congruent. The more similar $x[n]$ and $y[n]$ are, the less the DTW has to warp $y[n]$ and vice versa. DTW is a common technique in ASR, data mining and information retrieval.

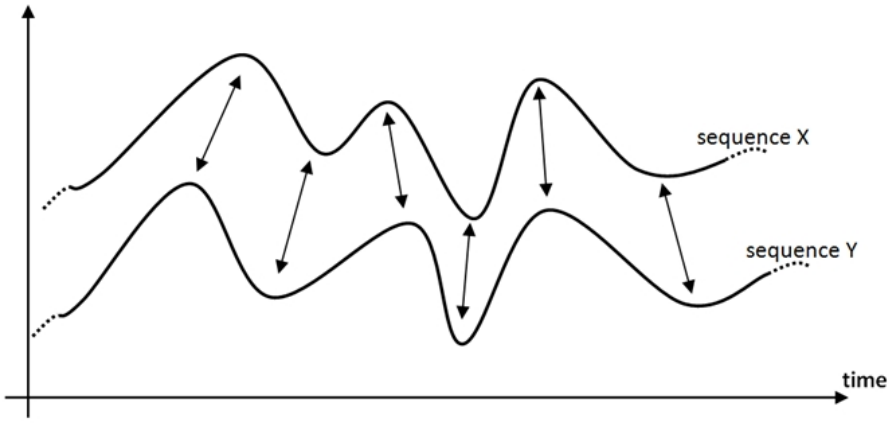


Figure 3.14: *Alignment of two signals via DTW.*

Cost Matrix

We assume two sequences $\mathbf{x} = [x_1, x_2, ..., x_N]$ and $\mathbf{y} = [y_1, y_2, ..., y_M]$ with length $N, M \in \mathbb{N}$. Furthermore, we assume a feature space $\mathcal{F}$, so that $x_n, y_m \in \mathcal{F}$ for $n \in [1 : N]$ and $m \in [1 : M]$. To compare two feature points $\mathbf{x}$ and $\mathbf{y}$ in the feature space $\mathcal{F}$ a local cost measure or local distance measure is needed. This cost measure is defined as

$$c : \mathcal{F} \times \mathcal{F} \to \mathbb{R}_{\geq 0}. \tag{3.15}$$

If x_n and y_m are similar to each other, i.e. the distance between them is small, the cost function $c(x_n, y_m)$ will usually decrease. Otherwise, the cost will increase. Now all distances between each pair of elements of $\mathbf{x}$ and $\mathbf{y}$ are calculated. The result is a cost matrix $\mathbf{C} \in \mathbb{R}^{N \times M}$ defined by $\mathbf{C}[n, m] = c(x_n, y_m)$ shown in Figure 3.15. Here the Manhatten distance, i.e. the absolute value of the distance was taken. Regions with low distance are marked dark and regions with high distance are marked in a bright color. The goal is to find an alignment between $\mathbf{x}$ and $\mathbf{y}$ with minimal cost, i.e. to find a warping path $\mathbf{p}$ through the cost matrix $\mathbf{C}$ with minimal overall cost, i.e. minimal distance.

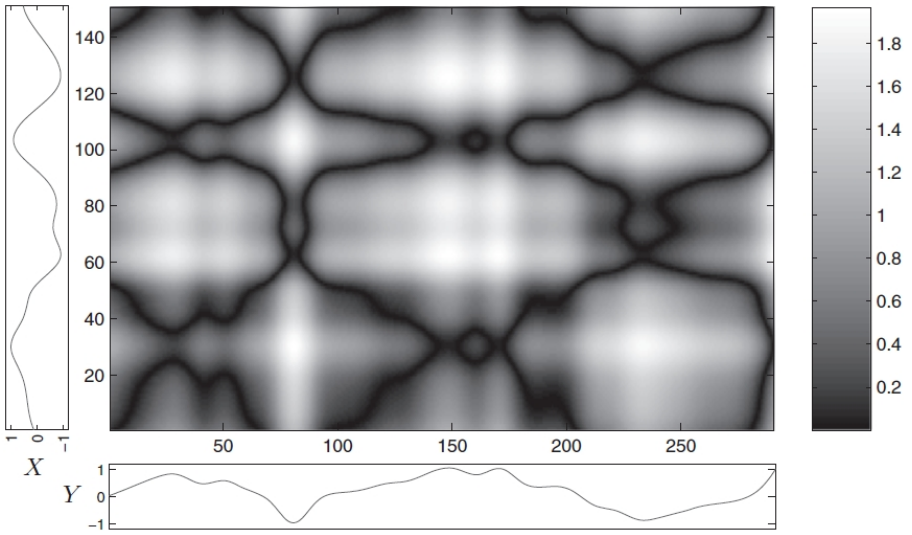


Figure 3.15: *Cost matrix between the sequences* $\mathbf{x}$ *and* $\mathbf{y}$ *using Manhatten Distance, taken from [19].*

Warping Path

After calculating the cost matrix, a warping path is calculated with minimal overall cost. This warping path is a sequence $\mathbf{p} = [p_1, p_2, ..., p_L]$ with $p_l = [n_l, m_l] \in [1 : N] \times [1 : M]$ for $l \in [1 : L]$ which aligns the sequences $\mathbf{x}$ and $\mathbf{y}$ by assigning each element x_{n_l} of $\mathbf{x}$ to an element y_{m_l} of $\mathbf{y}$. The warping path needs to fulfill three conditions:

boundary condition $p_1 = [1, 1]$ and $p_L = [N, M]$, i.e. the first element x_1 of $\mathbf{x}$ and y_1 of $\mathbf{y}$ and the last element x_N of $\mathbf{x}$ and y_M of $\mathbf{y}$ are aligned together. The alignment refers to the entire sequences $\mathbf{x}$ and $\mathbf{y}$.

monotonicity condition $n_1 \leq n_2 \leq ... \leq n_L$ and $m_1 \leq m_2 \leq ... \leq m_L$, i.e. requirement for faithfull timing: if an element of $\mathbf{x}$ precedes a second one, this should also hold for the corresponding elements in $\mathbf{y}$, and vice versa.

step-size condition $p_{l+1} - p_l \in \{[1, 0], [0, 1], [1, 1]\}$ for $l \in [1 : L-1]$, i.e. the path has to be continuous. No element x_n of $\mathbf{x}$ and y_m of $\mathbf{y}$ can be omitted and no replications in the alignment are allowed, i.e. all index pairs in the warping path are pairwise distinct.

Figure 3.16 shows four potential warping paths. 3.16(a) meets all three conditions. Thus it is a possible warping path. In 3.16(b) the *boundary condition* is violated since it is not starting in point $[1, 1]$ and ending in $[9, 7]$, i.e. not all elements are aligned together. In 3.16(c) the *monotonicity condition* is violated since the warping path goes from point $[5, 5]$ to $[6, 4]$, i.e. it is going back in time in y-direction. Finally, the warping path in

3.16(d) violates the *step-size condition* because it is jumping from $[4, 3]$ to $[7, 4]$, i.e. two elements in $\mathbf{x}$ are omitted.

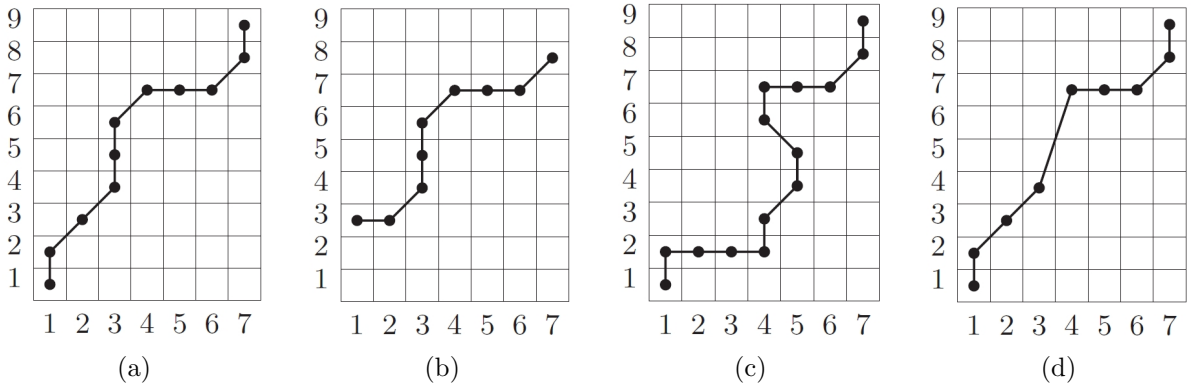


Figure 3.16: *Different warping paths for DTW. (a) Correct warping path. (b) Boundary condition is violated. (c) Monotonicity condition is violated. (d) Step-size condition is violated. Taken from [19].*

Optimal Warping Path

To find an optimal warping path $\hat{\mathbf{p}}$, the total cost function $c_{\mathbf{p}}(\mathbf{x}, \mathbf{y})$ defined as

$$c_{\mathbf{p}}(\mathbf{x}, \mathbf{y}) = \sum_{l=1}^{L} c(x_{n_l}, y_{m_l}) \tag{3.16}$$

needs to be minimized. The optimal warping path between $\mathbf{x}$ and $\mathbf{y}$ has minimal total cost among all possible warping paths, i.e. the *DTW distance* between $\mathbf{x}$ and $\mathbf{y}$ is defined as the total cost of the optimal warping path $\hat{\mathbf{p}}$

$$\begin{aligned} DTW(\mathbf{x}, \mathbf{y}) &= c_{\hat{\mathbf{p}}}(\mathbf{x}, \mathbf{y}) \\ &= min\{c_{\mathbf{p}}(\mathbf{x}, \mathbf{y}) \mid \mathbf{p} \text{ is an (N,M)-warping path}\}. \end{aligned} \tag{3.17}$$

To determine the optimal warping path, one can calculate all possible warping paths and take the one with the smallest total cost, but this will lead to a computational complexity which is exponential in the length of $\mathbf{x}$ and $\mathbf{y}$. Thus, another algorithm is taken which is $O(N \cdot M)$ and based on dynamic programming. Therefore the accumulated cost matrix $\mathbf{D}[N, M]$ is introduced

$$\mathbf{D}[n, m] = DTW(x[1:n], y[1:m]), \tag{3.18}$$

with $x[1:n] = [x_1, x_2, ..., x_n]$ for $n \in [1:N]$ and $y[1:m] = [y_1, y_2, ..., y_m]$ for $m \in [1:M]$.

$\mathbf{D}[N, M]$ fulfills following conditions:

$$\mathbf{D}[n, 1] = \sum_{k=1}^{n} c(x_k, y_1), \qquad n \in [1 : N], \tag{3.19}$$

$$\mathbf{D}[1, m] = \sum_{k=1}^{m} c(x_1, y_k), \qquad m \in [1 : M], and \tag{3.20}$$

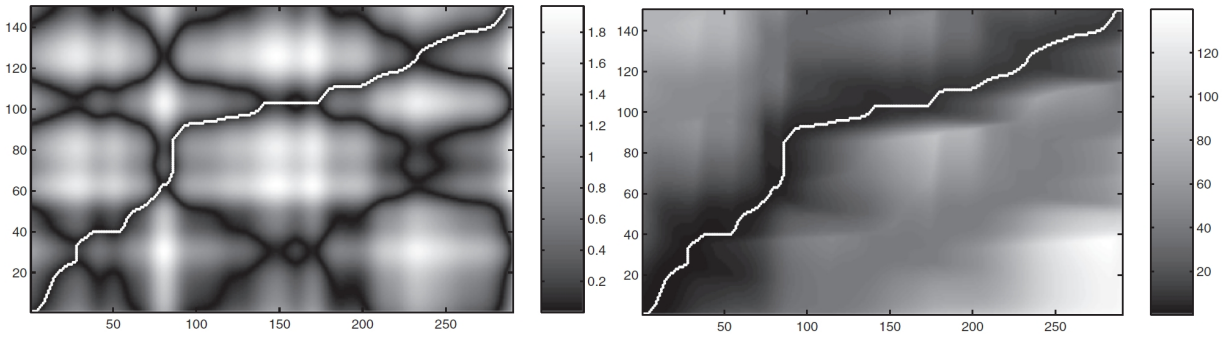
$$\mathbf{D}[n, m] = min \begin{cases} \mathbf{D}[n-1, m-1] + wc(x_n, y_m) \\ \mathbf{D}[n-1, m] + c(x_n, y_m) \\ \mathbf{D}[n, m-1] + c(x_n, y_m), \end{cases} \tag{3.21}$$

for $1 \leq n \leq N$ and $1 \leq m \leq M$. If the prior cell is $\mathbf{D}[n-1, m-1]$ sometimes the distance $c(x_n, y_m)$ is weighted with $w = 2$, because the diagonal step could be replaced by one step to the right side and one step upward [20]. Moreover $\mathbf{D}[N, M]$ can be calculated recursively. Thereby the matrix is extended by an additional row and column, i.e. $\mathbf{D}[n, 0] = \infty$ for $n \in [1 : N]$, $\mathbf{D}[0, m] = \infty$ for $m \in [1 : M]$ and $D[0, 0] = 0$, so that Equation (3.21) holds for $n \in [1 : N]$ and $m \in [1 : M]$. Now $\mathbf{D}[N, M]$ can be calculated in a column-wise fashion. The calculation of the m^{th} column only requires the storage of the $(m-1)^{th}$ column, which implies a storage requirement of $O(N)$. Similarly, $\mathbf{D}[N, M]$ can be calculated in a row-wise fashion, which implies a storage requirement of $O(M)$. In total there is a computation complexity of $O(N \cdot M)$. After the accumulated cost matrix $\mathbf{D}[N, M]$ has been calculated, the optimal path $\hat{\mathbf{p}}$ is determined in a reversed order of the indices starting with $p_L = [N, M]$. The calculation process is as follows

$$p_{l-1} = \begin{cases} [1, m-1] & \text{if } n = 1 \\ [n-1, 1] & \text{if } m = 1 \\ argmin\{\mathbf{D}[n-1, m-1], \mathbf{D}[n-1, m], \mathbf{D}[n, m-1]\} & \text{otherwise,} \end{cases} \tag{3.22}$$

where the lexicographically smallest pair is taken, if *argmin* is not unique. Figure 3.17 shows the optimal warping path $\hat{\mathbf{p}}$ in the cost matrix of Figure 3.15 and the accumulated cost matrix.

The DTW distance between the MFCCs x_i of the template consisting of $i = 1, ..., I$ frames and the MFCCs of the current audio signal is calculated for a block length of I frames using the absolute distance for cost function $c(x_n, y_m)$. It has been empirically observed that during reference recording hyper-articulation happens and the speaker talks slower than in general situations. Hence, we limit the block length to I frames. After processing time frame n, depicted in Figure 3.18 with red lines, the next time frame $n+1$ is processed, marked with green lines, with a step-size of $0.1I$ [21].



(a) Cost matrix **C** (b) Accumulated cost matrix **D**

Figure 3.17: *Optimal warping path determined from the accumulated cost matrix, taken from [19].*

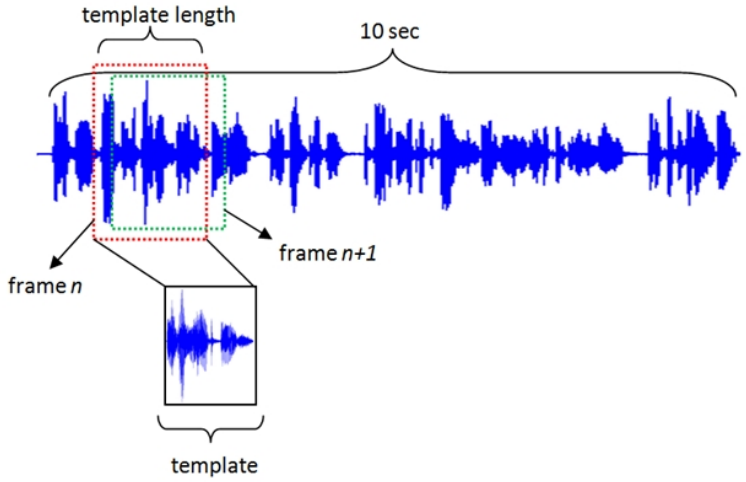


Figure 3.18: *Implementation of DTW for KWS.*

To norm the DTW distance, a MA is calculated for each vector

$$\mathrm{MA}_{\mathrm{DTW}}[n] = \begin{cases} \frac{1}{n}\sum_{i=1}^{n} DTW[i] & \text{if } n \leq N \\ \frac{1}{N}\sum_{i=n-N}^{n-1} DTW[i] & \text{if } N < n \end{cases}, \tag{3.23}$$

with $N = 20$. Afterwards the DTW vector is divided by its MA to receive the scaled DTW distance depicted in Figure 3.19. The smaller the DTW distance is, the more similar the current audio data is compared to the template. The position of the keyword is clearly visible because of the significant decrease of the DTW distance. Finally, the normed distance is compared to a threshold, i.e. if it is below a certain threshold, the

values of a vector DTW_{bit} are set to 1

$$DTW_{\text{bit}}[n] = \begin{cases} 1 & \text{if } DTW[n] \leq T_{\text{DTW}} \\ 0 & \text{otherwise} \end{cases} \tag{3.24}$$

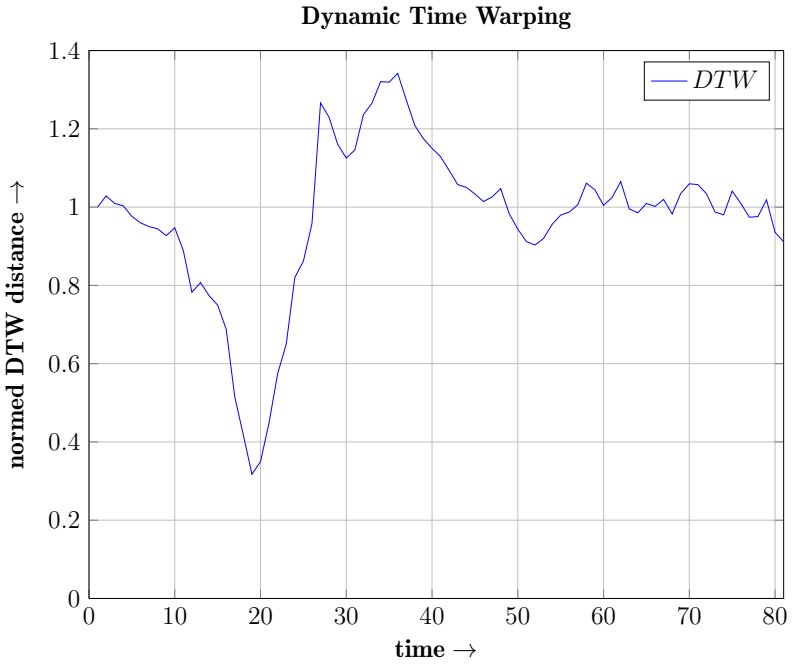


Figure 3.19: *DTW values for KWS.*

3.3.4 ED and CC

The ED between two vectors $\mathbf{x} = [x_1, x_2, ..., x_N]$ and $\mathbf{y} = [y_1, y_2, ..., y_N]$ is defined as

$$d(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{i=1}^{N} (x_i - y_i)^2}. \tag{3.25}$$

The CC is a measurement of the similarity between two signals. Equation (3.26) shows the CC for the discrete functions $x[n]$ and $y[n]$

$$r_{\text{xy}}[n] = (x \star y)[n] = \lim_{N \to \infty} \frac{1}{2N} \sum_{m=-N}^{N} x^*[m] y[n+m], \tag{3.26}$$

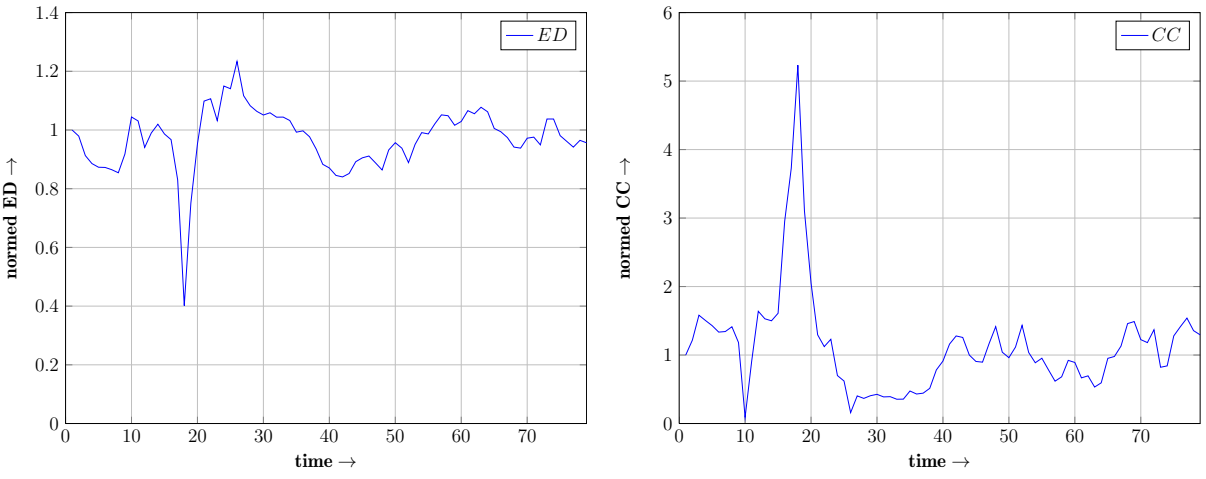
where $x^*[n]$ denotes the complex conjugate of $x[n]$. Since Equation (3.26) form a convolution between $x[n]$ and $y[n]$ without mirroring $y[n]$ there is a simple and easy way to calculate CC between two signals. If the spectra of the signals are calculated, the convo-

lution will turn into a multiplication [22], [23]. Therefore CC is realized by elementwise multiplication of the MFCC matrix $\mathbf{X}[m, c]$ of the template and $\mathbf{Y}[m, c]$ of the current audio data and accumulation afterwards.

For calculation of ED and CC each 10s audio frame is blocked into frames of length I with step-size $0.1I$ similar to the DTW calculation. Now ED and CC is calculated between the MFCC of each frame $\mathbf{Y}[m, c]$ and the MFCC $\mathbf{X}[m, c]$ of the template. This results in two vectors containing ED and CC values. To norm these vectors, a MA is calculated, similar to Equation (3.23) for DTW. The ED and CC is scaled by their MA. This resulting ED and CC are depicted in Figure 3.20. The existence of the keyword is noticeable because of the decrease of ED and the peak in CC respectively. Finally, these normed values are compared to a threshold. The smaller ED is or the bigger CC is, the more similar the current audio frame is compared to the template. If ED is below a certain threshold or if CC exceeds a certain threshold, the values of a vector ED_{bit} or CC_{bit} are set to 1

$$ED_{\text{bit}}[n] = \begin{cases} 1 & \text{if } ED[n] \leq T_{\text{ED}} \\ 0 & \text{otherwise,} \end{cases} \tag{3.27}$$

$$CC_{\text{bit}}[n] = \begin{cases} 1 & \text{if } CC[n] \geq T_{\text{CC}} \\ 0 & \text{otherwise.} \end{cases} \tag{3.28}$$



(a) (b)

Figure 3.20: *Scaled (a) ED and (b) CC values for KWS.*

3.3.5 Final combinations of ED, CC and DTW

In the end there are four sequences, i.e. $ED_{\text{bit}}[n]$, $CC_{\text{bit}}[n]$, $DTW1_{\text{bit}}[n]$ and $DTW2_{\text{bit}}[n]$, per 10s audio frame. These sequences containing 1, where a (possible) keyword is detected and 0 otherwise. Now different Boolean combinations of these criteria are possible:

logical *and* can be achieved by elementwise multiplication of the corresponding sequences, e.g. an alarm should only be raised if *ED and CC* detect the keyword, so the *final bit sequence* can be determined by elementwise multiplication of $ED_{\text{bit}}[n] \cdot CC_{\text{bit}}[n]$.

After calculating the *final bit sequence*, an additional criterion can be used to reduce false alarms, namely, VAD. If this is used, VAD classifies each block resulting in an alarm in the *final bit sequence*. If the VAD classifies this audio frame as speech, the alarm is triggered. Otherwise the next audio frame is processed. The nomenclature of the combination used is for example *ED-CC-DTW-VAD*, which means *ED and CC and (DTW1 or DTW2) and VAD*.

4
Rhythm Detection Algorithm

The implementation of the rhythm detection algorithm is explained in detail in this chapter. First, one needs to record a personalized rhythm. This rhythm should consist of at least three claps and at most six claps. If it has less than three claps, there will be many false alarms. On the other hand, if it has more than six claps, the reproduction of the rhythm is difficult and the recognition rate decreases. After recording this reference rhythm, i.e. the template, it is detected in the continuous audio signal.

4.1 AAAR

For detecting the rhythm the AAAR is introduced. Figure 4.1 shows the steps for computing the AAAR. In the first step, the absolute value of the audio signal is determined and summed over a frame-size of 25ms with a hop-size of 15ms. This results in the AAA, see Section 3.1.1. Afterwards the ratio between two consecutive energy values is calculated

$$AAAR[n] = \frac{AAA[n+1]}{AAA[n]}. \tag{4.1}$$

If the energy is changing slightly, the ratio is low. If there are strong variations between adjacent energy values, the ratio increases. Thus, calculating the ratio supresses signals with relative constant energy, i.e. speech, and emphasises transient signals such as clapping or knocking. The output of these calculations is a sequence of ratio values.

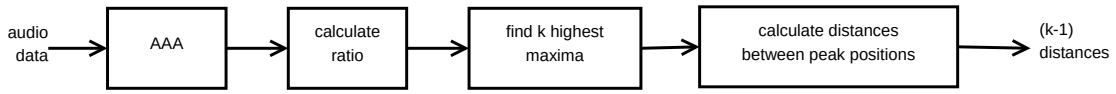


Figure 4.1: *Flow chart of the average absolute amplitude ratio.*

In a next step, the k highest maxima are determined, i.e. peak position and value. The output of this stage are k peak positions and k peak values. k depends on the number of claps in the reference recording.

Finally, all k peak values have to exceed a certain threshold T_{peak}. If not all k values pass this check, the next chunk of audio data is processed. If all k values exceed the threshold, the distances between their peak positions are calculated

$$AAAR_{\text{distance}}[m] = peak_{\text{position}}[m+1] - peak_{\text{position}}[m], \quad m = 1, ..., k-1. \tag{4.2}$$

The output are $k-1$ distances between the single peaks [24].

4.2 Implementation

The implementation of the rhythm detection algorithm is depicted in Figure 4.2. First, the audio data is again blocked in frames of 10s in length with 2s overlap. Afterwards the AAAR is calculated as described in Section 4.1. Figure 4.3 shows an audio signal in (a). From this, the AAA is calculated and depicted in (b). Finally, (c) shows the AAAR. There are significant peaks in the AAAR where the claps are situated.

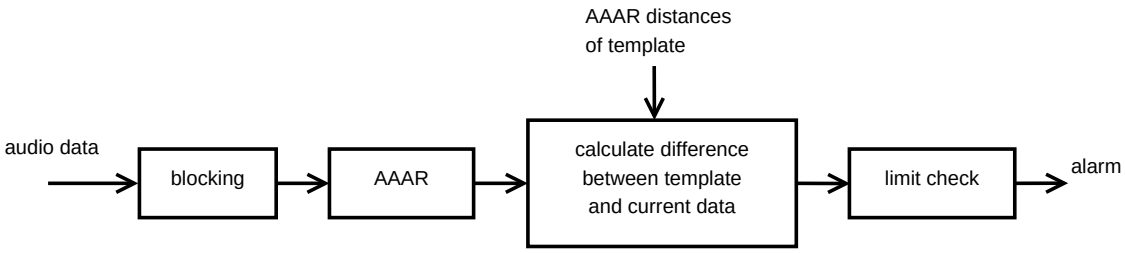


Figure 4.2: *Implementation of the rhythm detection algorithm.*

Subsequently, the signal is blocked into frames of the same length as the template with a step-size of $0.1 * LTemplate$ as depicted in Figure 4.4. In each of these frames, k maxima are determined. In this example, there are 4 maxima marked with a red circle. Where k is equal to the number of claps in the reference recording. If all maxima exceed a threshold $T_{\text{peak}} = 2$, the difference between the single peak positions is calculated resulting in $AAAR_{\text{distance,current}}[m]$, otherwise the next audio frame, marked in green colour, is processed.

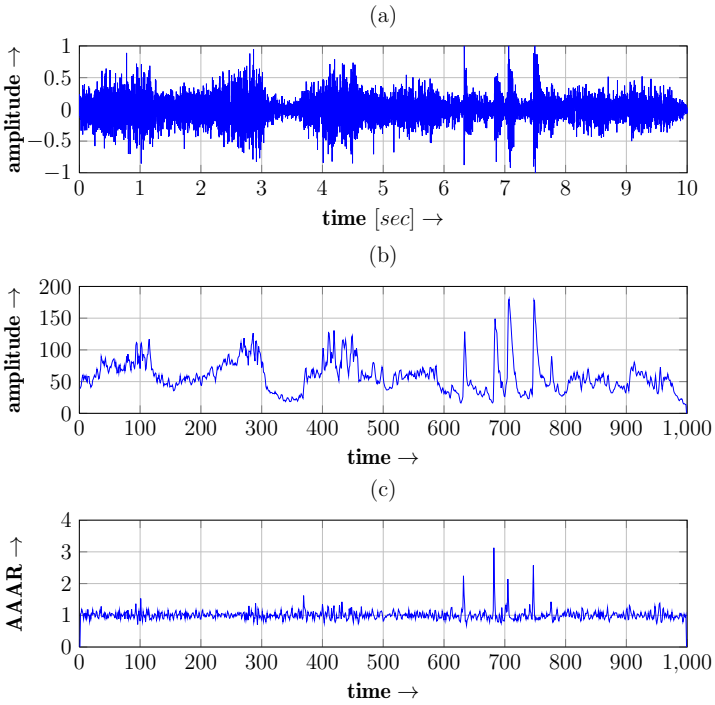


Figure 4.3: *AAAR of an audio signal. (a) PCM audio signal. (b) AAA of the audio signal. Frame-size is 25ms. Hop-size is 10ms. (c) AAAR, i.e. ratio between consecutive AAA values.*

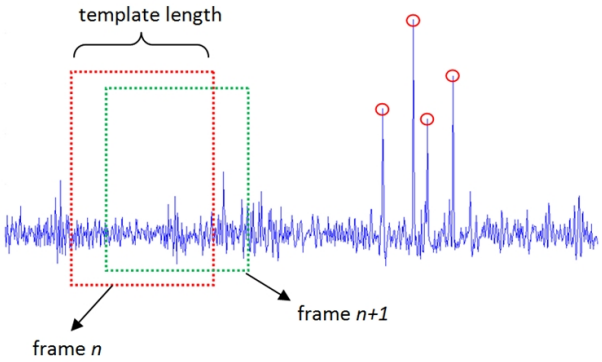


Figure 4.4: *Implementation of the AAAR for rhythm detection.*

Finally, the differences of the peak positions of the current audio data are compared to the differences of the peak positions of the template by calculating the absolute difference

$$AAAR_{\text{final}} = \sum_{m=1}^{k-1} |AAAR_{\text{distances,current}}[m] - AAAR_{\text{distances,template}}[m]|, \tag{4.3}$$

where k is the number of maxima, i.e. claps. If this final difference is below a certain threshold $T_{\text{AAAR}} = 8k$, i.e. the differences of the peak positions are similar between current audio data and template, an alarm is triggered.

5 Experimental Results

In this chapter performance measures are presented. Moreover the used databases for KWS and rhythm detection are introduced. Finally, some performance results are presented for the KWS algorithm and the rhythm detection algorithm.

5.1 Performance Measure

The performance is measured via *recall* (R) and *precision* (P)

$$R = \frac{TP}{TP + FN} \cdot 100 \qquad \text{in } [\%], and \tag{5.1}$$

$$P = \frac{TP}{TP + FP} \cdot 100 \qquad \text{in } [\%], \tag{5.2}$$

where TP are the true positives, i.e. the number of correct alarms. FN are the false negatives, i.e. the number of missing alarms and FP are the false positives, i.e. the number of false alarms. Therefore *recall* is a measurement of how many correct keywords are detected of all possible keywords, i.e. $R = 100\%$ means that all keywords that occurred are detected, $R = 50\%$ means that just half of all keywords are detected correctly. *Precision* measures how many correct alarms occurred compared to all alarms. $P = 100\%$ means all alarms are correct alarms, $P = 50\%$ means half of the alarms are correct, the other ones are false alarms.

Furthermore, we define the performance $\hat{\mathbf{P}}$ as

$$\hat{\mathbf{P}} = \frac{w_R R + w_P P}{w_R + w_P}, \tag{5.3}$$

where w_R and w_P is the weighting factor for *recall* and *precision* respectively. This is a requirement by the company, see Section 1.3. The combination of the distance measures is based on $\hat{\mathbf{P}}$.

5.2 Keyword Spotting Results

This section shows the performance of the KWS algorithm. First, different keywords are analyzed. Afterwards the best keyword is selected for a more detailed examination of the algorithm with background noise and different combinations of ED, CC, DTW and VAD.

5.2.1 Database for KWS

To evaluate the KWS algorithm, a database was recorded. This database consists of 10 different speakers in 3 different background noise scenarios and 4 different distances to the microphone. In total there are 12 recordings per speaker, i.e. 3 background noise scenarios with 4 different distances. In particular, the distance between speaker and microphone was 1m, 5m, and speaker was in an adjacent room to the recording device with either open or closed door. The background noise, i.e. television, was always in a distance of about 1m to the device. On each recording the speaker has to say the keyword ("Aktiviere Notruf!") 4 times. In total 480 keywords embedded in continuous audio data are recorded. Table 5.1 presents the mean a-posteriori signal to noise ratio SNR_{post} and the standard deviation σ of SNR_{post} for the different background noise scenarios and the different distances between microphone and speaker. SNR_{post} is determined as follows:

$$SNR_{\text{post}}[n] = 10 \log \left(\frac{E\{|x[n]|^2\}}{E\{|w[n]|^2\}} \right), \qquad \text{in } [dB], \tag{5.4}$$

where $E\{\cdot\}$ is the expectation operator. $x[n] = s[n] + w[n]$, where $s[n]$ is the speech signal and $w[n]$ is the noise signal. More details about the database can be found in the Appendix B.

	silence		**noise**		**high noise**	
	SNR_{post}	σ	SNR_{post}	σ	SNR_{post}	σ
1m	18.4	8.6	8.2	7.0	6.9	4.8
5m	21.5	6.7	6.3	6.6	4.3	2.7
ORO	15.7	8.0	5.2	7.2	3.5	3.6
ORC	4.3	3.5	1.5	4.5	-0.3	4.7

Table 5.1: SNR_{post} *in [dB] of different background noise scenarios for the KWS database. ORO is other room with open door, i.e. the user was in an other room than the smartphone. ORC is other room with closed door. The distance between smartphone and noise source, i.e. television, was always 1m.*

5.2.2 Performance depending on Keyword

Since the recognition rate of the KWS algorithm depends on the used keywords, different keywords are analyzed. In Figure 5.1 two keywords with different characteristics ("*Aktiviere Notruf*" and "*Hilfe Notruf*") are evaluated using the KWS algorithm. Therefore, speakers said the keywords two times in different situations. Afterwards this database is analyzed with the KWS algorithm. Each datapoint in Figure 5.1 represents the result of the KWS algorithm with different threshold settings. Table 5.2 displays the settings of the KWS algorithm and the database.

parameter name	**value**
used combination	*ED-CC-DTW*
probands	5 male, 4 female
	age 22 - 80 years
#audio recordings	27
#keywords	54
#MFCC bins	17
T_{DTW}	0.6 - 0.8
T_{ED}	0.6 - 0.8
T_{CC}	1.2 - 1.4

Table 5.2: *Settings for comparison of different keywords.*

The keywords have different performance characteristics. "*Aktiviere Notruf*" has by trend a higher precision but is more difficult to detect, i.e. it has a lower recall, whereas "*Hilfe Notruf*" has a higher recall but therefore causes more false alarms, i.e. lower precision. Since the KWS algorithm should have as little false alarms as possible, i.e. high precision, the keyword "*Aktiviere Notruf*" is taken to set up a database as described in Section 5.2.1 and in the Appendix B for further testing.

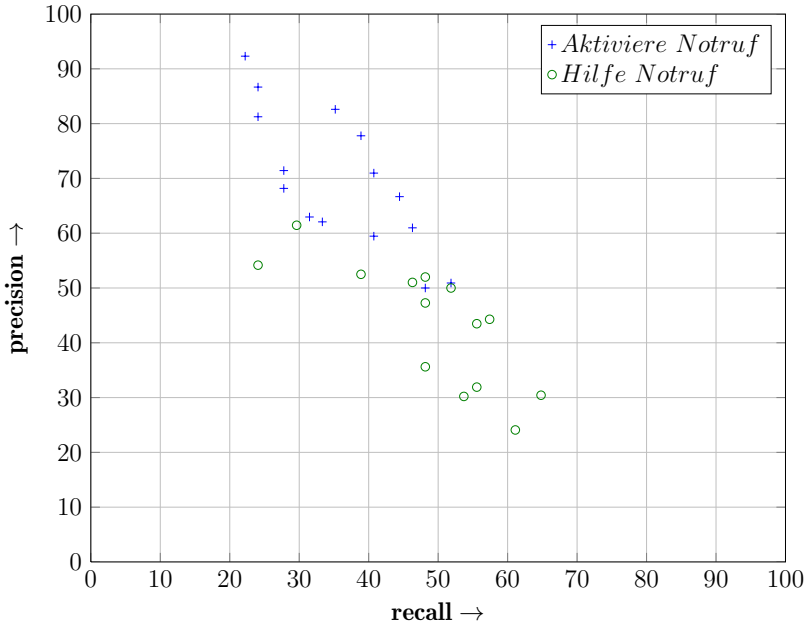


Figure 5.1: *Comparison of recall and precision of two different keywords.*

parameter name	value
used combination	*ED-CC-DTW*
#audio recordings	12 per proband
#keywords	48 per proband
#MFCC bins	17
T_{DTW}	0.75
T_{ED}	0.75
T_{CC}	1.3

Table 5.3: *Settings for comparison of different speakers.*

5.2.3 Performance depending on Speaker

The database consists of 10 different speakers. Figure 5.2 shows the result of the KWS algorithm according to the different speakers. The settings for simulation are presented in Table 5.3. There are no significant differences between male and female speakers. Table 5.4 presents the averaged results of male and female speakers.

	male	female
R [%]	55.4	53.8
P [%]	94.3	97.0

Table 5.4: *Experimental results depending on gender.*

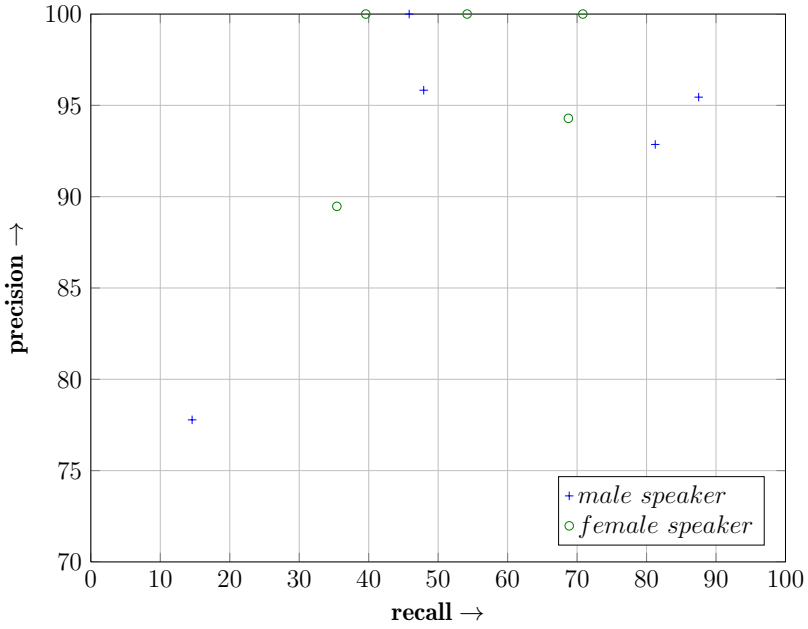


Figure 5.2: *Comparison of recall and precision of different speakers.*

parameter name	value
#audio recordings	40 per situation
#keywords	160 per situation
#MFCC bins	17
T_{DTW}	0.75
T_{ED}	0.75
T_{CC}	1.3
T_{VAD}	0.6

Table 5.5: *Settings for comparison of ED, CC, DTW and VAD.*

5.2.4 Performance depending on Background Noise and different Combinations of ED, CC, DTW and VAD

Different combinations of ED, CC, DTW and VAD are evaluated on the database. Therefore, the database was split into three parts *silence, noise* and *high noise* to find the best combination for each background noise situation. The settings of the simulation are shown in Table 5.5.

Table 5.6 presents the results of this experiment. The first three columns show the results of the different background situations, whereas in the last column the results of the full database, i.e. silence, noise and high noise together is presented. One can see that *recall* is decreasing while *precision* is increasing from top to bottom, because the more criteria are used, the less false alarms are produced. On the other hand, the correctly detected alarms also decrease. The results are also shown in Figure 5.3. One can clearly distinguish between the different background noise scenarios, since the performance is

decreasing with higher background noise.

Furthermore, the results of ED, CC, DTW and VAD are weighted with Equation (5.3) and $w_R = 1$, $w_P = 3$, i.e. precision is more important than recall, since false alarms should be avoided. Table 5.7 shows the performances of the different combinations. Now for each background noise situation the maximum performance is selected, marked with green color in the table. If no BNC takes place, only one combination can be used for all background noise situations. Therefore, the combination with the maximum performance on the database, *ED-CC-DTW*, is used, although this combination does not have optimal performance in each background situation. To remedy this drawback, BNC is implemented as described in Section 3.2. Now for each background noise scenario a different combination can be used with optimal performance, marked with green colour in Table 5.7. Depending on the classification of the background noise using HIST, an optimal combination is used. The final performance is presented in the next section.

Combination	**Silence**		**Noise**		**High Noise**		**Combined**	
	R	**P**	**R**	**P**	**R**	**P**	**R**	**P**
ED	80.0	84.2	65.0	54.5	49.4	44.9	64.8	59.9
DTW	87.5	99.3	66.3	76.8	40.0	68.8	64.6	83.3
CC	93.8	92.0	70.0	35.1	61.3	27.6	75.0	43.0
ED-CC	80.0	99.2	57.5	79.3	43.8	61.4	60.4	80.8
ED-DTW	75.6	100.0	58.1	92.1	32.5	83.9	55.4	93.7
CC-DTW	84.4	99.3	60.0	85.7	37.5	79.0	60.6	89.8
ED-CC-DTW	75.6	100.0	53.8	95.6	32.5	89.7	54.0	96.3
ED-VAD	67.5	84.4	51.9	55.0	31.9	48.6	50.4	63.0
DTW-VAD	73.1	99.2	56.3	79.7	31.9	75.0	53.8	86.3
CC-VAD	80.6	92.8	53.8	41.2	41.9	35.6	58.8	52.6
ED-CC-VAD	68.1	99.1	45.6	80.2	30.6	68.1	48.1	84.6
ED-DTW-VAD	64.4	100.0	47.5	92.7	24.4	95.1	45.4	96.5
CC-DTW-VAD	72.5	99.2	50.0	87.9	30.0	84.2	50.8	92.1
ED-CC-DTW-VAD	64.4	100.0	43.8	95.9	24.4	97.5	44.2	98.1

Table 5.6: *Experimental results of the KWS algorithm with different ED, CC, DTW and VAD combinations depending on the noise scenario. R and P in [%].*

5.2.5 Final Performance results on KWS Database

Finally, the KWS algorithm is tested on the database with and without BNC. If no classification takes place, only one combination can be used for all different background scenarios. If HIST is calculated, each 10s audio frame of an audio recording is classified into one of the following three categories *silence, noise* or *high noise* and depending on the output of the classifier a different ED, CC, DTW and VAD combination is used. Table 5.8 shows the settings for this experiment. Table 5.9 presents the results of the database

Combination	Silence $\hat{P}$	Noise $\hat{P}$	High Noise $\hat{P}$	Combined $\hat{P}$
ED	83.2	57.1	46.0	61.1
DTW	96.4	74.2	61.6	78.6
CC	92.5	43.8	36.0	51.0
ED-CC	94.4	73.9	57.0	75.7
ED-DTW	93.9	83.6	71.0	84.1
CC-DTW	95.6	79.3	68.6	82.5
ED-CC-DTW	93.9	85.1	75.4	85.7
ED-VAD	80.2	54.2	44.4	59.9
DTW-VAD	92.6	73.8	64.2	78.2
CC-VAD	89.8	44.3	37.2	54.1
ED-CC-VAD	91.4	71.6	58.7	75.5
ED-DTW-VAD	91.1	81.4	77.4	83.7
CC-DTW-VAD	92.5	78.4	70.7	81.8
ED-CC-DTW-VAD	91.1	82.9	79.2	84.7

Table 5.7: *Weighted Performance of the different combinations ($w_R = 1$ and $w_P = 3$). The best combination in each background noise scenario is marked with green color.*

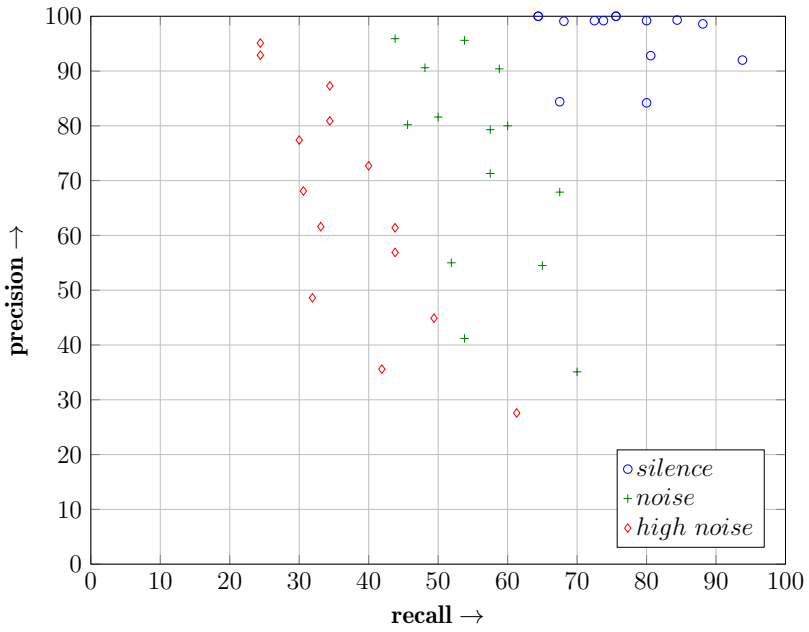


Figure 5.3: *Comparison of recall and precision in different situations and different ED, CC, DTW and VAD combinations.*

for different distances between speaker and smartphone and different background noise scenarios using no BNC and BNC, respectively. Table 5.10 presents the combined results of the different background noise scenarios for the final evaluation and compares it to the results without BNC.

parameter name	**value**
#audio recordings	40 per situation
#keywords	160 per situation
#MFCC bins	17
T_{DTW}	0.75
T_{ED}	0.75
T_{CC}	1.3
T_{VAD}	0.6
T_{HIST1}	0.45
T_{HIST2}	0.8

Table 5.8: *Settings for evaluation of the KWS.*

(a)

	Silence ED-CC-DTW				**Noise** ED-CC-DTW				**High Noise** ED-CC-DTW			
	1m	5m	ORO	ORC	1m	5m	ORO	ORC	1m	5m	ORO	ORC
R	90.0	87.5	80.0	45.0	75.0	72.5	55.0	12.5	55.0	35.0	35.0	5.0
P	100.0	100.0	100.0	100.0	93.8	96.7	95.7	100.0	95.7	87.5	100.0	40.0
$\hat{\mathbf{P}}$	97.5	96.9	95.0	86.3	89.1	90.6	85.5	78.1	85.5	74.4	83.8	31.3

(b)

	Silence DTW				**Noise** ED-CC-DTW				**High Noise** ED-CC-DTW-VAD			
	1m	5m	ORO	ORC	1m	5m	ORO	ORC	1m	5m	ORO	ORC
R	95.0	97.5	95.0	57.5	75.0	67.5	50.0	15.0	42.5	27.5	27.5	5.0
P	100.0	100.0	97.4	100.0	96.8	96.4	95.2	100.0	100.0	84.6	100.0	100.0
$\hat{\mathbf{P}}$	98.8	99.4	96.8	89.4	91.3	89.2	83.9	78.8	85.6	70.3	81.9	76.3

Table 5.9: *Final results of the KWS algorithm on different distances and different background noises. (a) Without BNC. (b) With BNC. R and P are in [%] and* $w_R = 1$ *and* $w_P = 3$*. ORO is other room with open door, i.e. the user was in an other room than the smartphone. ORC is other room with closed door. The distance between smartphone and noise source, i.e. television, was always 1m.*

It can be seen that the performance improves in all background noise scenarios if BNC is used. The differences in *recall* and *precision* between the results of the KWS algorithm without BNC in each background noise scenario, in Table 5.6, and the final results with BNC, in Table 5.9(b), are due to different classifications of single audio frames.

Moreover, in Figure 5.4 the performance curves of the KWS algorithm with and without BNC are presented. The settings for the evaluation to determine the performance curves are presented in Table 5.11. Each point on the curve corresponds to a different setting of the thresholds. It can be seen that BNC improves the precision. Whereas the BNC has an adverse effect on recall, i.e. the recall is decreasing compared to the recall without

(a)

	silence ED-CC-DTW	**noise** ED-CC-DTW	**high noise** ED-CC-DTW	**combined**
R	75.6	53.8	32.5	54.0
P	100.0	95.6	89.7	96.3
$\hat{\mathbf{P}}$	93.9	85.1	75.4	85.7

(b)

	silence DTW	**noise** ED-CC-DTW	**high noise** ED-CC-DTW-VAD	**combined**
R	85.6	51.9	25.6	54.4
P	99.3	96.5	95.3	97.8
$\hat{\mathbf{P}}$	95.9	85.4	77.9	86.9

Table 5.10: *Final results of the KWS algorithm. (a) Without BNC just one combination can be taken for all different background noise scenarios. (b) With BNC the best combination for each background noise scenario can be taken. R and P are in [%] and* $w_R = 1$ *and* $w_P = 3$.

BNC. But because more emphasis is put on precision, BNC is implemented. Moreover BNC reduces the computational costs, since not all criteria have to be calculated all the time.

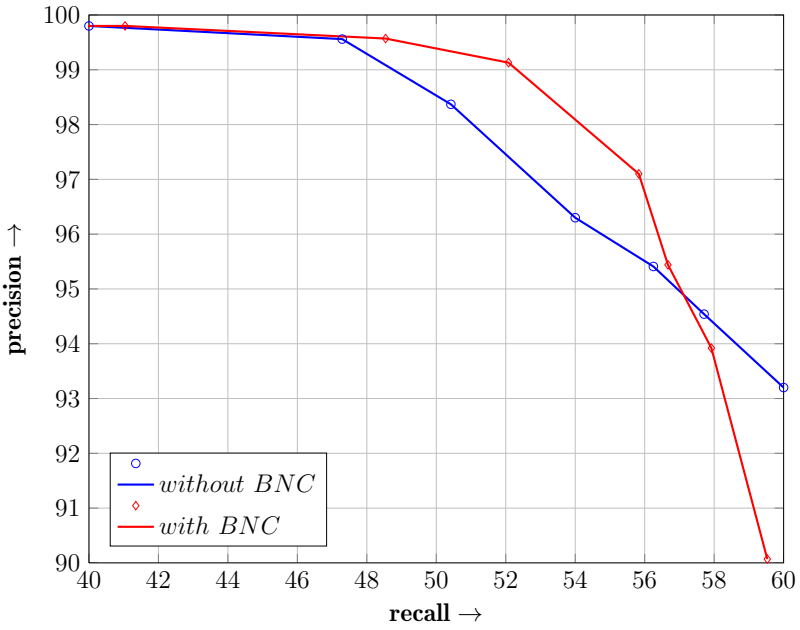


Figure 5.4: *Final performance curves of the KWS algorithm. R and P are in [%].*

parameter name	value
#audio recordings	40 per situation
#keywords	160 per situation
#MFCC bins	17
T_{DTW}	0.55 - 0.95
T_{ED}	0.55 - 0.95
T_{CC}	1.2 - 1.7
T_{VAD}	0.6
T_{HIST1}	0.45
T_{HIST2}	0.8

Table 5.11: *Settings to determine the precision-recall curve of the KWS.*

5.3 Rhythm Detection Results

The rhythm detection algorithm is also evaluated on a database. The results relating to different frame- and hop-sizes, different signals, i.e. clapping or knocking, and different background noise scenarios are presented in this section. But before the database is introduced.

5.3.1 Database for Rhythm Detection

The performance of the rhythm detection algorithm is evaluated using a database consisting of 128 audio samples, i.e. 64 audio samples with clapping and 64 audio samples with knocking. In each audio sample, the rhythm is clapped/knocked 2 times. In particular, the distance between speaker and microphone was 1m, 3m, 5m, and speaker was in an adjacent rrom to the recording device with closed door. The background noise, i.e. television, was always in a distance of 1m to the device, similar to the KWS database. More details about the database can be found in the Appendix C. The performance was measured using *recall* and *precision* as introduced in Section 5.1. Table 5.12 presents the mean a-posteriori SNR, SNR_{post} (see Equation 5.4) with $s[n]$ is the signal containing the rhythm and $w[n]$ is the noise signal, and its standard deviation σ for different background noise scenarios and the different signals clapping and knocking.

5.3.2 Performance depending on Frame-size and Hop-size

For calculating the AAA of the audio signal, different frame- and hop-sizes can be used. Table 5.13 shows the results of the rhythm detection algorithm using different frame- and hop-sizes. The settings of the simulation are presented in Table 5.14. It can be seen that *recall* is decreasing with higher frame-size while *precision* is increasing. On the other hand, *recall* is increasing with a higher hop-size while *precision* is falling. To find an

Distance	Clapping				Knocking			
	Silence		Noise		Silence		Noise	
	SNR_{post}	σ	SNR_{post}	σ	SNR_{post}	σ	SNR_{post}	σ
1m	27.7	4.2	16.6	2.3	35.6	7.0	16.1	2.8
3m	35.1	4.7	16.3	4.8	24.0	6.6	8.7	1.3
5m	32.1	2.1	15.0	1.5	24.4	1.8	9.9	2.0
ORC	24.5	2.7	8.5	2.0	27.4	8.3	10.7	1.3

Table 5.12: SNR_{post} *in [dB] of different background noise scenarios and distances for the rhythm detection database. ORC is other room with closed door, i.e. the person was in an other room than the smartphone. The distance between smartphone and noise source, i.e. television, was always 1m.*

optimal solution for the frame- and hop-size, the performance $\hat{\mathbf{P}}$ is determined using the Equation (5.3) with $w_R = 1$ and $w_P = 3$. Again the focus is a low false alarm rate. Now the combination with the highest performance, marked in green, is choosen for further processing.

	Hop-size [ms]								
	5			10			15		
Frame-size [ms]	**R**	**P**	$\hat{\mathbf{P}}$	**R**	**P**	$\hat{\mathbf{P}}$	**R**	**P**	$\hat{\mathbf{P}}$
10	73.4	100.0	93.4	-	-	-	-	-	-
15	72.7	100.0	93.2	96.9	93.2	94.1	-	-	-
20	69.5	100.0	92.4	93.8	96.0	95.4	97.7	70.6	77.4
25	64.8	100.0	91.2	93.0	99.2	97.6	97.7	84.5	87.8

Table 5.13: *Experimental results of different frame-sizes and hop-sizes for AAAR calculation. R and P are in [%] and* $w_R = 1$ *and* $w_P = 3$*. The best combination is marked with green color.*

5.3.3 Performance depending on Background Noise

To evaluate the performance depending on the background noise, the database is split into two parts, i.e. silence and noise. Table 5.15 presents the settings for the algorithm. In Table 5.16 the experimental results are presented. One can see that there are no significant changes in *precision* but that *recall* is dependent on the background noise. This is due

parameter name	value
database	clapping
reference	clapping
T_{AAAR}	32
T_{peak}	2

Table 5.14: *Settings for simulation with different frame- and hop-sizes.*

parameter name	value
database	clapping, knocking
reference	clapping, knocking
T_{AAAR}	32
T_{peak}	2
frame-size	25ms
hop-size	10ms

Table 5.15: *Settings for evaluation with different background noise scenarios.*

template	database	Silence		Noise		Combined	
		R	P	R	P	R	P
clapping	**clapping**	100.0	100.0	85.9	98.2	93.0	99.2
knocking	**knocking**	93.8	100.0	45.3	100.0	69.5	100.0
clapping	**knocking**	93.8	100.0	39.1	100.0	66.4	100.0
knocking	**clapping**	100.0	100.0	84.4	98.2	92.2	99.2

Table 5.16: *Experimental results in different background noise scenarios and different template and database signals. R and P are in [%].*

to the calculation process of AAAR. If there is background noise, the AAA increases and therefore the AAAR peaks that are needed for the detection of the rhythm are too small. As a result, they fall under the threshold and are disregarded for further calculations. Hence *recall* is decreasing. On the other hand, *precision* stays nearly the same because the background noise can be seen as stationary compared to clapping/knocking. Hence, the AAAR peaks do not exceed the threshold T_{AAAR} and the chunk of audio data is processed. Therefore, no false alarms occur.

Table 5.16 shows that in silence *recall* and *precision* are independent of the excitation signal, i.e. clapping or knocking. In contrast, in noisy environments *recall* of the knocking database is decreasing more strongly compared to the clapping database. This is due to the more quiet and low-frequency sound of knocking, which is masked more easily in background noise. On the other hand, the performance is independent of the used signal in the reference recording, since the detection process is only based on the distances between the peaks and hence independent of the acoustical characteristics of the signal.

5.3.4 Final Performance results on Rhythm Database

In Table 5.17 the results of the rhythm detection algorithm depending on the noise are presented. The settings of this experiment are presented in Table 5.18.

Moreover, the rhythm detection algorithm is evaluated on the combined database using the clapping reference, i.e. clapping and knocking database in silence and noise. In total there are 128 different audio samples consisting of 256 rhythms. The results are

(a)

	Silence				Noise			
	1m	3m	5m	ORC	1m	3m	5m	ORC
R	100.0	100.0	100.0	100.0	100.0	81.25	100.0	62.5
P	100.0	100.0	100.0	100.0	100.0	100.0	100.0	90.91
$\hat{\mathbf{P}}$	100.0	100.0	100.0	100.0	100.0	95.3	100.0	83.8

(b)

	Silence				Noise			
	1m	3m	5m	ORC	1m	3m	5m	ORC
R	100.0	93.8	100.0	81.3	81.3	31.3	25.0	18.8
P	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
$\hat{\mathbf{P}}$	100.0	98.4	100.0	95.3	95.3	82.8	81.3	79.7

Table 5.17: *Final results of the rhythm detection algorithm. Clapped reference is taken for both tables. (a) Clapping database. (b) Knocking database. R and P are in [%] and* $w_R = 1$ *and* $w_P = 3$.

parameter name	**value**
database	clapping and knocking
reference	clapping
T_{AAAR}	32
T_{peak}	2
frame-size	25ms
hop-size	10ms

Table 5.18: *Settings for the rhythm detection.*

presented in Table 5.19. The first column shows the results in silence. In the second column the results in a noisy environment are presented. There are 64 different audio samples consisting of 128 rhythms in both background noise scenarios. The last column shows the overall performance of the rhythm detection algorithm on the whole database, with 128 audio samples. Attention should be paid to the high overall precision of 99.5%, i.e. there are only 0.5% false alarms.

Figre 5.5 presents performance curves with different settings of the thresholds T_{AAAR} and T_{peak}. Each point on a performance curve corresponds to a different setting of the

	silence	**noise**	**combined**
R	96.9	62.5	79.7
P	100.0	98.8	99.5
$\hat{\mathbf{P}}$	99.2	89.7	94.6

Table 5.19: *Results of the rhythm detection algorithm. R and P are in [%] and* $w_R = 1$ *and* $w_P = 3$.

parameter name	value
database	clapping and knocking
reference	clapping
T_{AAAR}	20 - 200
T_{peak}	1, 1.5, 2, 3
frame-size	25ms
hop-size	10ms

Table 5.20: *Settings for the rhythm detection to determine the performance curves.*

threshold T_{AAAR}. The settings for the simulation of the curves are presented in Table 5.20. The blue curve is simulated with a threshold T_{peak} of 1, the green curve with a threshold T_{peak} of 1.5, the red curve with a threshold T_{peak} of 2 and the black curve with a threshold T_{peak} of 3. All maxima of AAAR less than this T_{peak} are not considered in the detection process, see Chapter 4. Therefore with an increasing T_{peak} the precision is getting better, while the recall decreases, because the maxima of AAAR do not exceed T_{AAAR} and are rejected in the rhythm detection process.

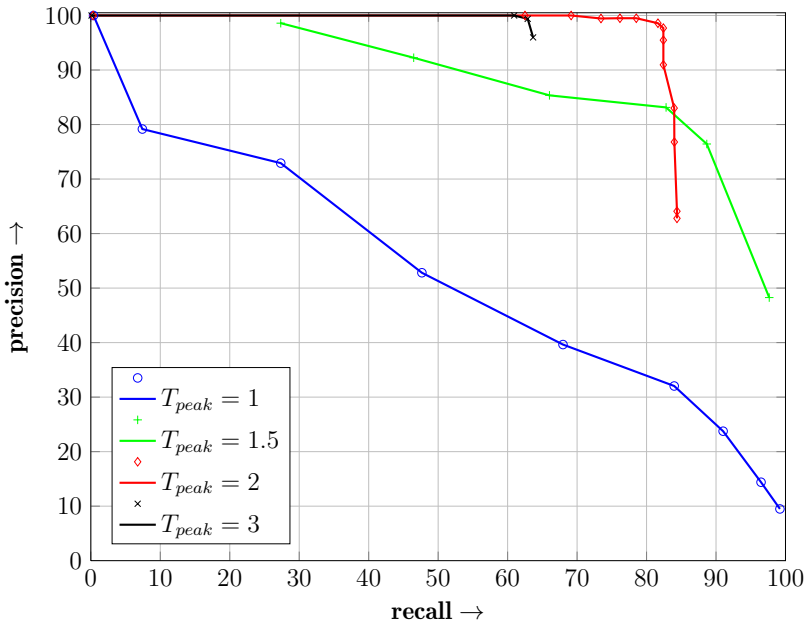


Figure 5.5: *Performance curves of the rhythm detection algorithm. R and P are in [%].*

6 Conclusion

This thesis describes a keyword spotting and a rhythm detection algorithm. Both algorithms were developed with particular emphasis on real-time processing and low energy consumption on a smartphone. The KWS algorithm is based on pattern matching, since it should work for personalized keywords, without any statistical model such as hidden Markov models. Hence it can be used for different languages without the need of adapting the speech model. Only the template pattern has to be individually recoreded. The algorithm detects the occurrence of a specified keyword that has been recorded beforehand. A single reference recording is enough to start the keyword spotting or the rhythm detection. First, the audio data is blocked into frames of 10s in length. Then it is transformed into frequency domain using mel frequency cepstral coefficients. Afterwards three distance measures, i.e. euclidian distance, cross-correlation and dynamic time warping are used to determine the similarities between the reference, i.e. the template, and the current audio data. Each block is classified into one of the following three classes: silence, noise, high noise using a histogram of the amplitudes. Depending on the background noise, a different combination of the distance measures and additional voice activity detection is used to detect the keyword by comparing the distance values to a threshold. Since a low false alarm rate is more important than a high recognition rate, the combination used for each background noise category is determined by calculating the performance with a weighting of recall to precision of 1 to 3. Then the combination with the highest performance in each background noise category is implemented for the final keyword spotting algorithm.

For rhythm detection the average absolute amplitude ratio is introduced. Thereby the ratio of adjacent average absolute amplitude values is calculated, which emphasises transient signals such as clapping or knocking and suppresses continuous signals such as speech. Afterwards the distances between the single peak positions in the average absolute amplitude ratio are calculated and compared to the template. An alarm is triggered if the distances do not exceed a certain deviation. Since the algorithm is working in time domain, it does not require any transformation. Consequently, it has very low computational and energy costs.

The keyword spotting algorithm is tested on different databases concerning the used keyword, different background noise scenarios and different speakers. The final keyword spotting database consists of 10 different speakers recorded in 12 different situations. In total there are 120 audio recordings consisting of 4 times the keyword, i.e. in total 480 times the keyword ("Aktiviere Notruf!"). To evaluate the rhythm detection algorithm, a rhythm database was recorded. There are 8 different situations each with 8 different recordings for clapping and knocking. In total there are 64 recordings consisting of 128 times the clapped rhythm and 64 recordings consisting of 128 times the knocked rhythm. Both algorithms were developed and tested in Matlab using the signal processing toolbox and the voicebox. After receiving satisfactory simulation results, they were translated into C code for processing on smartphones and tablets. As a result, both algorithms can be executed on smartphones in real time. The overall performance for keyword spotting is a recall of 54.4% and a precision of 97.8%. The rhythm detection algorithm achieves a recall of 79.7% and a precision of 99.5%. In particular, the rhythm detection is more robust in noisy and for distance, i.e. other room closed door) scenarios.

At the same time, the conclusion of this thesis can also be regarded as an outlook for further development. Instead of MFCCs, other features such as linear predictive coding (LPC), perceptual linear predictive (PLP) or delta and delta-delta features of the MFCCs can be used. Investigations can also be conducted in the classification of background noise. Statistical models of the background noise could lead to a more sophisticated system.

A
File Directory

Name	Description
AEDcalc	.m-function to calculate AAAD for reference recording quality check
AERcalc	.m-function to calculate AAAR for rhythm detection algorithm
AZCcalc	.m-function to calculate AZCR for reference recording quality check
Corrcalc	.m-function to calculate CC for KWS algorithm
DTWcalc	.m-function to calculate DTW for KWS algorithm
EDcalc	.m-function to calculate ED for KWS algorithm
HISTcalc	.m-function to calculate HIST for background noise classification
MAcalc	.m-funciton to calculate MA for KWS algorithm
maxcalc	.m-funtion to calculate k highest maxima for rhythm detection algorithm
VADcalc	.m-function to calculate VAD for KWS algorithm
KWS_DA	.m-script to simulate the finished KWS algorithm with HIST
KWS_DA_crit	.m-script to simulate the KWS algorithm without HIST to determine the results of the single criterias
Rhythm_DA	.m-script to simulate rhythm detection algorithm

Table A.1: Matlab files

Name	Description
KWS_results	Presents *recall* and *precision* of the KWS algorithm on the used database with different combinations of *ED, CC, DTW* and *VAD*. Via *recall weight* and *precision weight* the emphasis of *recall* and *precision* can be changed to find the best combination.
KWS_results_single	Presents *recall* and *precision* of the final KWS algorithm on the used database split up into the different background noise scenarios and the different distances between speaker and microphone. Moreover the SNR_{post} of the database is presented.
Rhythm_results	Presents *recall* and *precision* of the rhythm detection algorithm on the rhythm database with different settings of frame- and hop-size for AAAR. Via *recall weight* and *precision weight* the emphasis of *recall* and *precision* can be changed to find the best setting for hop- and framesize.
Rhythm_results_single	Presents *recall* and *precision* of the final rhythm detection algorithm on the database. Split up into the different signals, background noise scenarios and distances between user and microphone. Moreover the SNR_{post} of the database is presented.

Table A.2: Excel files

B

Experimental Set-Up for KWS database recording

To evaluate the KWS algorithm a database consisting of 120 audio samples (including 480 times the keyword) was recorded. The recording took place in a closed room (living room, eating room) and the recording device is a smartphone. Further details of the set-up are described in the following.

speakers:

- sex: 5 male, 5 female
- age: 1x 10-20 years, 1x 20-30 years, 2x 30-60 years, 1x 60+ years

distances: distance between speaker and phone

- 1m
- 5m
- other room with open door (i.e. speaker was in an other room than the smartphone)
- other room with closed door

background noise:

- silence: some low noise due to open doors.
- low noise: smartphone was 1m next to a television running at low volume.
- high noise: smartphone was1m next to a television running at high volume.

nomenclature of recordings: sex - age - situation - background noise

- male → M, female → W
- 1m → 01m, 5m → 05m
- other room open door → 0AR
- other room closed door → ARG
- silence → SIL, low noise → 0HG, high noise → HG+
- example: male, 25 years, other room with open door, low noise → M25_0AR_0HG

procedure of recording:

- each audio recording is about 2 min
- each person had to do all 4 different distances with the 3 different background noises
- there are 12 audio recordings per person (4x silence in all situations, 4x low noise in all situations, 4x high noise in all situations)
- in each of these 12 audio recordings, the proband had to say the keyword 4 times (after about 30s, 60s, 90s and 120s)

Checklist for audio recording:

proband: Max Muster, 107 years	**Check**
1m, Silence	
5m, Silence	
other room open door, silence	
other room closed door , silence	
1m, low noise	
5m, low noise	
other room open door, low noise	
other room closed door, low noise	
1m, high noise	
5m, high noise	
other room open door, high noise	
other room closed door, high noise	

C

Experimental Set-Up for Rhythm Detection database recording

To evaluate the rhythm detection algorithm a database consisting of 128 audio samples (including 256 times the rhythm) was recorded. The recording took place in a closed room (living room, eating room) and was done with a smartphone. Further details are described below.

signal:

- clapping with hands
- knocking on the table

distances: distance between proband and phone

- 1m
- 3m
- 5m
- other room with closed door (i.e. the speaker was in an other room than the smartphone)

background noise:

- silence: some low noise due to open doors.
- low noise: smartphone was 1m next to a television running at low volume

nomenclature of recordings: situation - signal - background noise - number of record

- 1m → 1m, 3m → 3m, 5m → 5m
- other room closed door → ARG
- clapping → kla, knocking → klo
- silence → stille, low noise → tv
- example: clapping, other room with closed door, with noise 3rd record → ARG-kla-tv3

procedure of recording:

- each audio recording is about 2min
- 4 different distances and 2 different background noises → 8 different situations
- in each of these 8 situations there are 8 recordings
- there are 64 audio recordings for clapping and 64 audio recordings for knocking (8x 1m, 8x 3m, 8x 5m, 8x other room with closed door, each with and without background noise)
- in each of these 64 audio recordings, the rhythm is clapped/knocked 2 times (after about 50s and 100s)

List of Figures

List of Tables

List of References

[1] Huan, X. ; Acero, A. ; Hon, H.-W.: *Spoken Language Processing: A Guide to Theory, Algorithm and System Development.* Prentice Hall PTR, NJ, 2001.

[2] Kepuska, V. ; Klein, T.B.: A novel Wake-Up-Word speech recognition system, Wake-Up-Word recognition task, technology and evaluation. In: *Nonlinear Analysis 71, 2009*, 2009, S. 2772–2789.

[3] Cole, R. ; Mariani, J. ; Uszkoreit, H. ; Varile, G.B. ; Zaenen, A. ; Zampolli, A. ; Zue, V.: *Survery of the State of the Art in Human Language Technology.* Cambridge University Press and Giardini, 1997.

[4] Benayed, Y. ; Fohr, D. ; Haton, J-P ; Chollet, G.: Confidence measures for keyword spotting using support vector machines. In: *IEEE International Conference on Acoustics, Speech, and Signal Processing.*, 2003, vol. 1, S. 588–591.

[5] Ketabdar, H. ; Vepa, J. ; Bengio, S. ; Bourlard, H.: Posterior based keyword spotting with a priori thresholds. In: *Proceedings of Interspeech*, 2006.

[6] Rose, R.C. ; Paul, D.B.: A hidden Markov model based keyword recognition system. In: *International Conference on Acoustics, Speech, and Signal Processing.*, 1990, vol. 1, S. 129–132.

[7] Lleida, E. ; Marino, J.B. ; Salaverda, J. ; Bonafonte, A. ; Martinez, A.: Out of vocabulary word spotting modeling and rejection for keyword spotting. In: *EUROSPEECH*, 1993, S. 1265–1268.

[8] Bahi, H. ; Benati, N.: A new keyword spotting approach. In: *International Conference on Multimedia Computing and Systems*, 2009, S. 77–80.

[9] Keshet, J. ; Grangier, D. ; Bengio, S.: Discriminative keyword spotting. In: *Proceedings of Workshop on Non-Linear Speech Processsing*, 2007.

[10] Yu, P. ; Seide, F.: Fast Two-Stage Vocabulary-Independent Search In Spontaneous Speech. In: *IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2005, vol. 1, S. 481–484.

[11] Seide, F. ; Yu, P. ; Ma, C. ; Chang, E.: Vocabulary-independent search in spontaneous speech. In: *IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2004, vol. 1, S. 253.

[12] Ordelman, R. ; Jong, F. de ; Larson, M.: Enhanced Multimedia Content Access and Exploitation Using Semantic Speech Retrieval. In: *IEEE International Conference on Semantic Computing*, 2009, S. 521–528.

[13] Bridle, J.: An efficient elastic-template method for detecting given words in running speech. In: *Proceedings British Acoustic Society Meeting*, 1973, S. 1–4.

[14] Guo, H. ; Huang, D. ; Zhao, X.: An algorithm for spoken keyword spotting via subsequence DTW. In: *IEEE International Conference on Network Infrastructure and Digital Content*, 2012, S. 573–576.

[15] Zhao, H. ; Zhao, L. ; Zhao, K. ; Wang, G.: Voice Activity Detection Based on Distance Entropy in Noisy Environment. In: *International Joint Conference on INC, IMS and IDC*, 2009, S. 1364–1367.

[16] Bin Amin, T. ; Mahmood, I.: Speech recognition using dynamic time warping. In: *International Conference on Advances in Space Technologies*, 2008, S. 74–79.

[17] Logan, B.: Mel frequency cepstral coefficients for music modeling. In: *International Symposium on Music Information Retrieval*, 2000.

[18] Fastl, H. ; Zwicker, E.: *Psycho-Acoustics. Facts and Models*. Springer, Heidelberg, 2007.

[19] Müller, M.: *Information Retrieval for Music and Motion*. Springer, Berlin Heidelberg, 2007.

[20] Kroschel, K. ; Rigoll, G. ; Schuller, B.: *Statistische Informationstechnik. Signal- und Mustererkennung, Parameter- und Signalabschätzung*. Springer, Berlin Heidelberg, 2011.

[21] Barakat, M. S. ; Ritz, C.H. ; Stirling, D.A.: Keyword spotting based on the analysis of template matching distances. In: *International Conference on Signal Processing and Communication Systems*, 2011, S. 1–6.

[22] Kammyer, K.-D. ; Kroschel, K.: *Digitale Signalverarbeitung. Filterung und Spektralanalyse mit Matlab-Übungen.* Vieweg+Teubner, Wiesbaden, 2009.

[23] Kiencke, U. ; Jäkel, H.: *Signale und Systeme.* Oldenbourg Wissenschaftsverlag, München, 2008.

[24] Lesser, N. ; Ellis, D.P.W.: Clap detection and discrimination for rhythm therapy. In: *IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2005, vol. 3, S. 37–40.