Noise Robust Keyword Spotting for Low-power Speech Communication Systems

- with Application to Hearing Aids -

A thesis submitted to the department of Electronic Systems at Aalborg University, by

Mohammad El-Sayed

In Partial Fulfillment of the Requirements for the Degree of Master of Science (MSc) in Engineering.
June 2016.

Aalborg University
Department of Electronic Systems
Fredrik Bajers Vej 7B
DK-9220 Aalborg

Department of Electronic Systems
Fredrik Bajers Vej 7
DK-9220 Aalborg Ø
http://es.aau.dk

AALBORG UNIVERSITY
STUDENT REPORT

Title:
Noise Robust Keyword Spotting for Low-power Speech Communication Systems - with Application to Hearing Aids

Theme:
Signal Processing and Computing

Project Period:
Sep. 2015 - Jun. 2016

Project Group:
15gr970

Participant(s):
Mohammad El-Sayed

Supervisor(s):
Zheng-Hua Tan
Jesper Jensen

Copies: 1

Page Numbers: 109

Date of Completion:
June 2, 2016

Abstract:

This project investigates the use of keyword spotting (KWS) to the application of low power mobile communication devices, and more specifically for hearing aids. Implementation of KWS into a mobile device allows communication to the device via a hands-free voice interface. A baseline KWS framework is selected from the literature, which accommodates the platform constraints of a mobile device. The selected system is subject to an evaluation of its keyword detection and rejection capabilities, based on real-life hearing aid recordings conducted by the author. Experiments have revealed that the selected KWS system is competitive to contemporary KWS algorithms and even outperforms them in some aspects. Finally, noise robustness is incorporated into the KWS system; two back-end noise adaption strategies are proposed, one of which has demonstrated superior performance over the baseline KWS system, and classical speech enhancement algorithms.

Contents

List of Figures

List of Tables

Preface

This master thesis presents the work prepared by Mohammad El-Sayed at the Institute of Electronic Systems at Aalborg University between September 2015 and June 2016. The work is as a part of the fulfillment of his Master of Science (MSc) degree in Engineering, in the field of Signal Processing and Computing.

The work has been proposed and supervised by Associate Professor Zheng-Hua Tan from Aalborg University, and Professor Jesper Jensen affiliated with both Oticon A/S and Aalborg University.

Aalborg University, June 2, 2016

Mohammad El-Sayed
<melsay11@student.aau.dk>

List of Abbrevations

ASR	Automatic Speech Recognition
AUC	Area Under the ROC Curve
DCT	Discrete Cosine Transform
DFT	Discrete Fourier Transform
DP	Dynamic Programming
DTW	Dynamic Time Warping
EER	Equal Error Rate
EM	Expectation-Maximization
FFT	Fast Fourier Transform
FN	False Negative
FNR	False Negative Rate
FOM	Figure Of Merit
FP	False Positive
FPR	False Positive Rate
GMM	Gaussian Mixture Model
HA	Hearing Aid
HATS	Head And Torso Simulator
HMM	Hidden Markov Model
KL	Karhunen-Loéve
KWS	Keyword Spotting

LPC	Linear Prediction Coefficients
LPCC	Linear Prediction Cepstral Coefficient
LVCSR	Large Vocabulary Continuous Speech Recognition
MAP	Mean Average Precision
MFCC	Mel frequency cepstral coefficient
ML	Maximum Likelihood
MSE	Mean Squared Error
NN	Neural Network
PDF	Probability Density Function
PSD	Power Spectral Density
ROC	Receiver Operating Characteristic
SDTW	Segmental Dynamic Time Warping
SSN	Speech Shaped Noise
SNR	Signal to Noise Ratio
STFT	Short-Time Fourier Transform
TP	True Negative
TN	True Positive
TPR	True Positive Rate
OOV	Out Of Vocabulary
VAD	Voice Activity Detection
WSS	Wide Sense Stationary

Chapter 1

Introduction

Over the past decades, the rapid development of ubiquitous mobile devices such as smart phones, wearables (i.e. body-attached devices), hearing aids etc, has nourished a search into more natural human-computer interfaces. In the traditional sense, humans and machines have heavily relied on intermediate physical devices to establish a communication link between the two, e.g. keyboards are commonly used for machine input and a computer display or screen as output. In terms of efficiency of communication speed, interactions that involve visual and/or auditory senses can obtain higher bandwidths; i.e. humans speak much faster than they type and can perceive information contained in images even faster than speech. When sensing modalities of different kind are combined (e.g. audition and sight) it further aids the perception of information at either the human and/or machine-end [1].

One such natural human-machine interface is speech, which have been an active research field for several decades, and thus have matured to such an extend that it is found in many practical implementations. Prime examples of such systems are constituted by Apple's Siri [2], Microsoft Cortana [3], Google Now [4], Amazon Echo/Alexa [5], and many more. Interaction via speech is a two-way communication between human and machine; automatic speech recognition (ASR) is employed at the machine-end to transform speech into text, the reverse action is speech synthesis which generates a feedback to the speaker from the machine. However, this does not imply that ASR algorithms can run in reverse to accomplish speech synthesis. These are considered two distinct areas within this field, and are therefore handled separately [1]. The work presented in this paper is bounded to address matters concerning ASR, despite the overlapping technical issues that are encountered in both fields of study.

The applications of ASR are countless, hence ASR systems adopt various complexities and constraints. Ranging from the simplest task of recognizing isolated words

of a single user, to speaker-independent dictation where all words in a continuous utterance stream are detected from a large vocabulary. The above-mentioned examples adopt the later approach that besides speaker variability also need to account for environmental noise conditions. Somewhere between the two extremes, keyword spotting (KWS) systems are accommodated. KWS aims at detecting the occurrence of predefined keywords (or a phrase of keywords) in an audio stream among other out-of-vocabulary (OOV) words, also denoted (non-keyword) filler words. Hence KWS can be considered a subproblem to ASR where only partial information within the speech signal needs to be decoded. The keyword may begin and end anywhere in the spoken utterance, rendering KWS a non-trivial task. In order for a keyword spotter to gain a high accuracy it should detect all keywords in an utterance whilst minimizing the number of false alarms (i.e. false-positives) [6].

The work of this thesis addresses the task of KWS particularly designed for mobile devices, such a system is imposed by the constraints of the mobile environment in which it operates. However, before drafting any such system, a mathematical definition of the KWS problem setting is formulated.

1.1 The Keyword Spotting Problem

In the following a mathematical description is presented in Definition 1.1 [7] in order to formalize the KWS task. The process can roughly be split into two stages; a *spotting stage* that calculates the confidence score of a keyword being present in a given utterance, and a *decision stage* that collects the scores exhibiting highest confidence and determines whether or not a given keyword is present in the utterance.

In conventional ASR systems, it is commonplace to resort to a transform domain, i.e. a feature domain in which a compressed representation of the speech signal can be found and which accounts for the natural variations of speech signals. Both temporal and spectral speech variabilities result from a combination of physiological and ethnic differences; e.g. age, dialect, gender, emotional state, etc. [8]. Hence features are designed to counteract the effects caused by these factors.

Definition 1.1 (KWS Problem Setting)

1. *Spotting stage*

   An input utterance composed of a sequence of acoustic features;

   $$\bar{\mathbf{x}} = [x_1 \; x_2 \; \ldots \; x_T] \,, \; x_i \in \boldsymbol{\mathcal{X}} \subset \mathbb{R}^d \tag{1.1}$$

   for all $1 \leq i \leq T$

where $\mathcal{X}$ is the space of all feature vectors, and x_i is the acoustic feature vector extracted from the i^{th} time frame and has dimension d. For now it is assumed that $\bar{\mathbf{x}}$ contains at most one keyword. Moreover, there is a natural variation in the time duration of a speech signal, even for the same uttered word, hence T is not fixed.

The predefined keywords are indexed by $k \in \mathcal{K}$, where $\mathcal{K}$ is the dictionary of keywords. The goal of the spotting stage can be described in terms of a mapping of $\bar{\mathbf{x}}$ to a confidence measure for the occurrence of k in $\bar{\mathbf{x}}$. Depending on the applied pattern matching, the score express a probability in case a statistical model is used, such as a neural network (NN) or hidden Markov model (HMM). Alternatively, it might be a distance measure, e.g. in the case where dynamic time warping (DTW) is applied. Likewise the different pattern matching methods are parametrized by a distinct set of parameters θ. For an HMM, θ may express the initial probability distribution, the state transition distribution etc.
In summary, the mapping function takes as inputs $\bar{\mathbf{x}}$ and k along with the parameters θ, and returns the confidence score $S_k \in \mathbb{R}$:

$$S_k = f_1(\bar{\mathbf{x}},\ k\ ;\ \theta) \tag{1.2}$$

The same procedure is repeated for all keywords in $\mathcal{K}$, to generate a score for each keyword.

2. *Decision stage*

Based on the score acquired in the *Spotting stage*, i.e. S_k, a decision on whether keyword k occurs in the spoken utterance $\bar{\mathbf{x}}$ is required. Therefore the *Decision stage* can be described in terms of a function $f_2(\cdot)$ that maps the confidence information contained in S_k to a binary decision output,

$$O_k = f_2\left(S_{k,o}\right) \tag{1.3}$$

let S_k be an expression of distortion, then the output value assumes either two values

$$O_k = \begin{cases} 1 & \text{if } S_{k,o} \leq b \\ 0 & \text{if } S_{k,o} > b \end{cases} \tag{1.4}$$

where b is a threshold value. Hence, decisions are commonly generated by comparing S_k against a threshold value $b \in \mathbb{R}$. When the distortion score is equal to or below the threshold value, the hypothesis that k is contained in $\bar{\mathbf{x}}$ is accepted, otherwise the hypothesis is rejected. Note that S_k is not limited to be an expression of distortion, in principle it can be an representation of any similarity metric which expresses the confidence that

$\boldsymbol{k}$ has occurred in $\bar{\mathbf{x}}$, and therefore may require the inequalities in Equation (1.4) to be reversed.

As a final measure, the two stages may be combined into a single stage. Whereby the KWS objective is formulated as to learn a composite function $f = f_1 \circ f_2$, parameterized by θ such that the decision output,

$$\boldsymbol{O}_{\boldsymbol{k}} = f\left(\bar{\mathbf{x}},\ \boldsymbol{k}\ ;\ \theta\right) \tag{1.5}$$

generates as a good performance as possible, according to some predefined figure of merit.

Numerous solutions have been developed to solve the KWS problem, among those the most common solution categories from the literature are reviewed and presented in the following.

1.2 State-of-the-Art Keyword Spotting Methods

Solutions for the KWS problem have been extensively explored throughout recent years, and has produced a vast number of different solutions. These often rely on ASR techniques that have found its applicability in KWS with some modifications, and hence typically fall into one of three major categories; template-based (or query-by-example) methods, keyword-filler methods, and large vocabulary continuous speech recognition (LVCSR) methods.

- *Query-by-Example Methods (Template Matching)*: Among the earliest methods used in the field of ASR is the template matching paradigm [11][12], in which the quired keyword template is matched against an input test utterance across all possible time segments. In the literature it is sometimes referred to as the sliding window approach [13], because the keyword template is slided across the input utterance in overlapping time windows. At each time window a similarity metric is computed using dynamic time warping (DTW) that also accounts for potential duration mismatch between input and test utterance. Thereby the fit between the template and the test utterance is evaluated at different time instances, i.e. a similarity score is calculated for each time window, and the keyword is regarded as detected whenever the score exceeds some predefined threshold value. The choice of the threshold value is of great importance as it settles the precision-recall trade-off.
  The whole procedure is carried out in two main steps; firstly, front-end processing is applied that extracts acoustic features at the frame-level, next the feature vectors of the template and test inputs are matched using DTW, from

which the similarity scores are computed [14]. A decision rule then uses the similarity scores to determine whether the putative keyword is deemed present or not. The procedure is summarized in Figure 1.1.

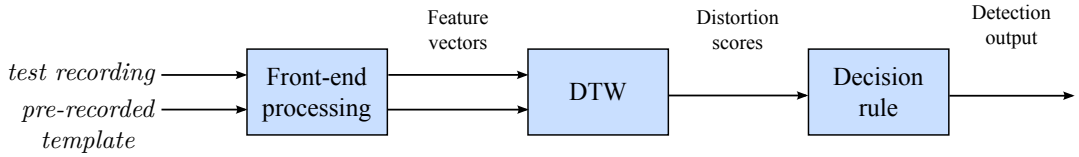


Figure 1.1: System-block overview of a template-based keyword spotter.

Template-based methods are generally computationally cheap since they do not attempt to transcribe the whole word sequence of the test signal. Template matching is in its essence simply a two-class discriminator that seeks to classify time regions of the test signal into either keyword or non-keyword instances. However, basic template-based systems also suffer from a number of drawbacks, in particular, they are prone to changes in the recording conditions between the template and input utterance [9]. Moreover template-based approaches become infeasible both memory-wise and computationally for a growing number of templates, e.g. due to the use of multiple keywords and/or multiple users.

- *Keyword-Filler Methods*: In the task of keyword spotting, the input recording is assumed to be a concatenation of keywords and non-keyword instances. Thus, one approach is to model them individually using a keyword model in parallel with a "filler" or "garbage" model for the non-keyword. The front-end processing remains unchanged in comparison to the template-based approach, i.e. keyword-filler models rely on per-frame speech features. However, the decoding process changes in that an acoustic model is being employed to decode the test utterance as outlined in Figure 1.2. Acoustic models are used to establish statistical representations of the acoustic feature vectors, thereby it can estimate the probability that a certain subword unit is uttered in the speech input. Similarly a pronunciation dictionary containing the keywords is applied to define the partitioning of a keyword into its subunits.

  A common choice for the acoustic model is the subword unit based HMM. Decoding is carried out by building a word-loop containing a network of keyword and filler models as depicted in Figure 1.3. A distinct model is allocated for each target keyword, and a filler-model is allocated to handle non-keywords. Each keyword HMM is trained using its corresponding keyword, and the filler HMM is trained using non-keyword segments [15].

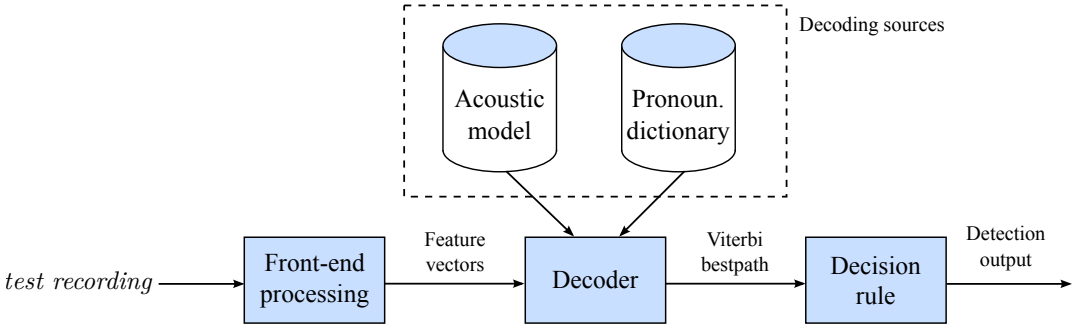


Figure 1.2: System-block overview of a keyword spotter using keyword-filler approach.

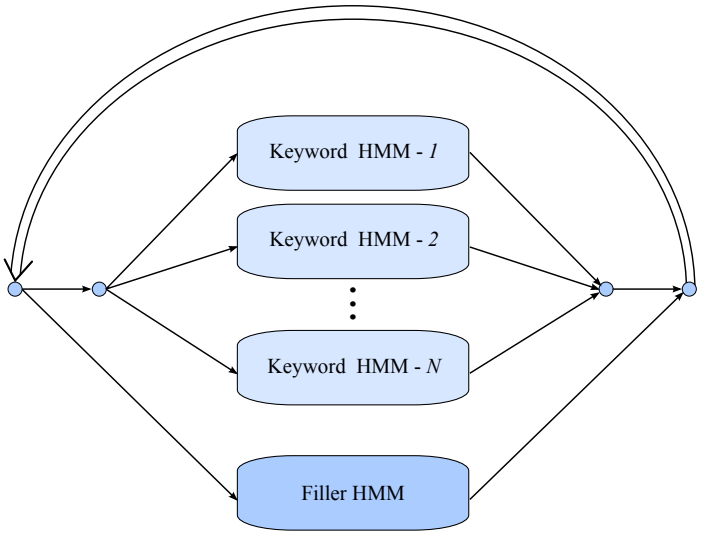


Figure 1.3: HMM-based modeling using a keyword-filler approach. Each observation in the HMM corresponds to an acoustic feature of the input utterance, whereas the hidden state corresponds to the subword unit accountable for generating the acoustic feature.

Using Viterbi decoding, the purpose is to determine the underlying subword sequence which is most likely to have generated the observed series of feature vectors. Keyword detection is carried out by verifying whether the Viterbi best-path includes the keyword HMM or not. In this regard, the choice of state-transition probabilities in the keyword HMM settles the precision-recall trade-off [9].

The keyword-filler methods generally outperform the classical template-based methods, as they explicitly model both keyword and non-keywords. Including a non-keyword provides higher confidence when accepting or rejecting putative instances of keywords in comparison to template-based methods that attempt discrimination solely based on knowledge of the target class [13]. But unfortunately, keyword-filler models require large amounts of tran-

scribed speech data for training the acoustic models, and in particular the filler model. Moreover, the Viterbi algorithm generates the best-path from a sequence of local decisions, making it prone to any local model mismatch, since a mismatch of a single subword can cause a keyword occurrence to be completely ignored. To gain robustness against the later limitation, likelihood ratio strategies have been suggested in [16] and [17].

- *Large Vocabulary Continuous Speech Recognition Methods*: The LVCSR-based methods differ from the other two methods in mainly two aspects; they attempt full transcription of the speech input and next apply a search routine for the designated keywords. Moreover they carry out recognition at the word level and therefore require a large word vocabulary to match the input signal against.
  As outlined in Figure 1.4, an LVCSR engine is employed to transcribe the entire input speech at word level. For this task conventional ASR system resources are needed, i.e. a language model to learn how to compose sequences of words, a large vocabulary of words, and not least acoustic models. The LVCSR engine finds the most probable word sequence based on Viterbi decoding. Next, a search algorithm is applied that examines the recognized text for the presence of of query terms. Prior to the search algorithm, the transcribed text is sometimes indexed in order to accelerate the search response time. One classical approach is to use word lattices for conducting the indexing [18]. Some later approaches adopted confusion networks [19][20] that makes use of posterior probabilities as opposed to prior probabilities that are used in word lattices.

  Although having proven very effective in terms of spotting accuracy [21], the LVCSR methods require a considerable amount of speech resources, and additionally has high computational demands as compared to template matching and keyword-filler methods [22]. This is mainly due to the fact that a conventional ASR system is employed prior to keyword spotting. Another limitation of LVCSR-based KWS methods is the lack of ability to define and include new keywords. Since the keyword list is usually predefined, any additional term will be considered an OOV word. To overcome this problem, other techniques such as subword modeling has been proposed in [23] and [24].

1.3 Project Delimitations

Provided with an overview of the cutting-edge technologies within the field of KWS, the next step is to delimit the range of potential solutions to the KWS problem at hand. In this context, it is of particular importance to select a solution

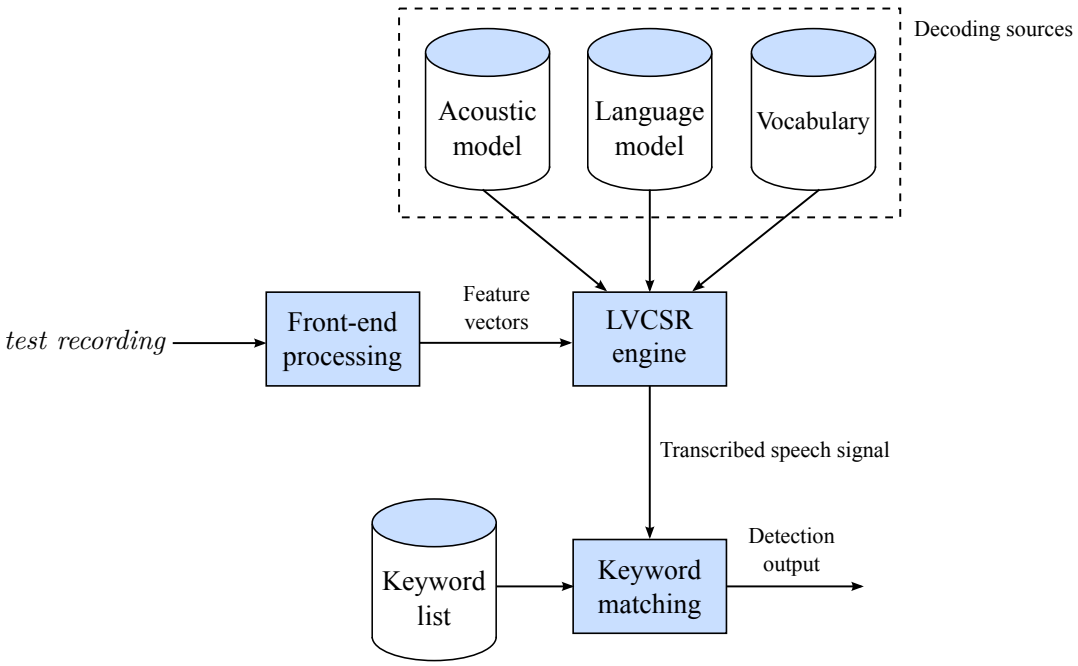


Figure 1.4: System-block overview of a keyword spotter based on a LVCSR typology.

that ensures feasibility for the computational environment in which it operates. In other words, the KWS solutions must be selected according to the capabilities or resources of the targeted platform, which comprises the major constraint. As outlined in the introduction, the application is aimed to run on a typical battery-driven modern mobile device, e.g. a hearing aid. Such a device holds considerable restrictions with respect to the available computational power, memory-size and not least the battery lifetime, rendering power consumption a crucial design metric. Furthermore, it is desired to have a KWS algorithm that executes in soft real-time, with a latency that resembles human-human communication delay (also known as mouth-to-ear delay) i.e. in the vicinity of 150-200 [ms] [27].

The above-mentioned constraints disqualify the LVSCR-based systems, as they attempt a full speech transcription that is computationally burdensome, not to mention the computational cost of indexing and the search routine. In addition, a large memory footprint follows from the requirement of a comprehensive vocabulary that contains up to hundreds of thousands of words, in that the words of interest need to be in the predefined vocabulary to be detectable. The keyword-filler methods have smaller memory footprint, since they resort to subword-based modeling using e.g. phonemes, which are the smallest units of sound capable of distinguishing one word from another. Most languages typically have a few tens of unique phonemes [25] that can be used to form millions of words. In particular the En-

glish language holds 44 phonemes [26], although this number may vary slightly depending on the type of English being described. Thus phonetic-based keyword-filler relaxes the memory requirement. But, the keyword-filler methods still attain higher computational complexity in comparison to the template-based KWS, mainly because they explicitly model both keyword and non-keyword terms using two or more distinct models. In addition to low computational complexity and small memory footprint, template-based methods have either little or no requirements of annotated speech data or prior knowledge of the underlying language and hence hold considerable promise for low resource languages. In the context of low power speech communication systems, template matching approaches are therefore an appealing choice of solution to the KWS problem, and they will be the main focus throughout this thesis.

As for the framework of the KWS system, we will make the following assumptions:

- The targeted platform is a personal mobile device that exclusively can be triggered by the native user of the system, i.e. we limit our scope to speaker-dependent recognition. As a practical example of a personal mobile communication device, the hearing aid will be highlighted.
- The size of the vocabulary is comprised of one or few personalized keyword, but at maximum 10 keywords. The personalized keywords are recorded by the user during a training phase.
- The keywords are carefully selected (in advance) by the system provider, implying that these are distinct words that rarely occur in everyday conversations. This also has the desired effect of averting false alarm detections.

1.4 Objectives

The core objective of this thesis is to enable users to communicate to a mobile device via a speech interface by developing an "always-on" (i.e a continuously listening) system that listens for one or more keywords to initiate voice input. The system is characterized by being highly specialized and accurate in detecting a few spoken keywords, while it refuses all other words, phrases, noise and other sound inputs. Thus keyword spotting is used as a mean of explicitly requesting the attention of some mobile device. Upon successful detection of the keyword/wake-up word, the system can initiate different applications or take certain actions. Triggering the keyword detector can e.g. be used to resign from a sleep mode, or initiate a "command-mode", where any subsequent voice input is interpreted a command for controlling the mobile device.
Generally, we will consider the KWS system to be language independent provided

that the system is trained for the respective language for which it will be tested. The system is therefore not aimed to handle multiple languages simultaneously.

A number of subgoals are formulated that pinpoint the steps taken to reach the overall objective. The answers of the following questions will be presented in Chapter 6.

- Review the fundamental principles behind speech production and speech representation used in the context of keyword spotting.
- Propose a KWS system that takes advantage of state-of-the-art techniques and accommodate the low-power constraints of a personal mobile device.
- Evaluate the selected keyword spotter for a practical mobile communication device by using real-life hearing aid recordings, and compare its performance relative to contemporary KWS systems.
- Investigate the degree of speaker dependency of the selected keyword spotter.
- Suggest various methods to improve noise robustness of the KWS system, and evaluate the effect on the spotting performance accordingly.
- Compare the proposed noise compensation strategy against state-of-the-art speech enhancement algorithms.

Chapter 2

Speech Fundamentals

Knowing the fundamental concepts and terminology behind speech theory is imperative to the understanding of existing as well as development of novel KWS algorithms. The speech signal characteristics are vital to the understanding of how speech signals must be processed in KWS systems and which assumptions can be made. Therefore the human anatomy that generates such signals will be subject to a brief description, along with an explanation of the human auditory system that senses such signals. The human auditory perception mechanism is e.g. utilized in acoustic feature extraction, as MFCC coefficients are derived by mimicking aspects of the human auditory system.

2.1 The Human Communication Paradigm

The primary human-human communication interface is speech; the communication link is initiated when a speech signal in form of a sound pressure wave is generated from the mouth of a speaker. The signal travels through the air and is subsequently perceived in the ear of the listener.

The communication pathway from the speaker to the listener is split into several steps at different abstraction levels as formalized by Denes and Pinson [28] trough their concept named the *speaker chain*. They conceive human communication as a series of levels, where the output of one level serve as an input to the next level [29], such as outlined in the speaker-chain diagram shown in Figure 2.1. The communication process begins at the linguistic level, where an idea (or message) is formulated in the mind of the speaker. Next, the idea is translated into language code, and words/sentences are formulated in accordance with the grammar of the spoken language. At the physiological level, the brain creates electrical signals that stimulates the motor nerves controlling the speech organs, which ensure that the requisite muscles are activated at the necessary time. The motor activity eventually results in a sound wave that propagates through space and into the ear of the

human counterpart. In turn, the sound wave brings about a pressure change in the ear canal of the listener and thereby causes vibrations in the ear drum. The ear drum is a membrane that convert acoustic pressure variations to mechanical vibrations. Subsequently, these vibrations are transmitted through a group of small bones denoted ossicles located in the middle ear, and thence to the cochlea of the inner ear. In the cochlea mechanical vibrations are converted to electrical signals that stimulates the sensory nerves and travels to the brain of the listener [30]. At the final stage speech recognition and understanding is carried out; the brain interprets the nerve impulses as messages in a linguistic form [29]. At this point the listener has perceived the message of the speaker, which happens at the same abstraction level as the outset level of abstraction.

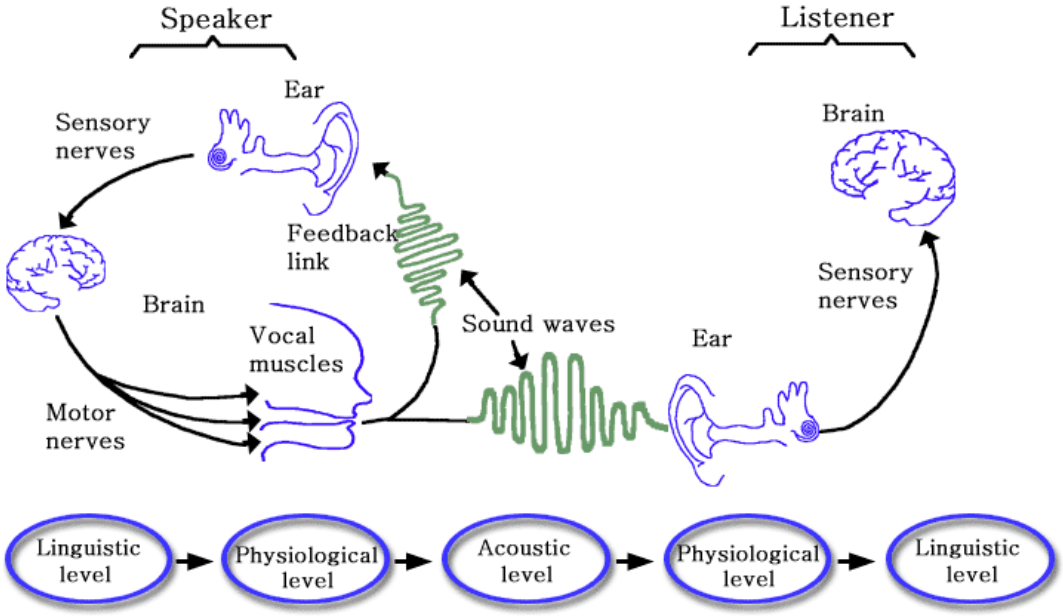


Figure 2.1: The Speaker Chain [28].

In relation to classic communication systems, the speaker and listener can be regarded as the transmitter and receiver, respectively. However, the transmitter and receiver serve other purposes than simply communicating. The transmitter has a feedback loop through the ear that enables monitoring and regulation of its own voice. The receiver part performs speech recognition that is robust to different types of noise sources and interferences, it can e.g. handle recognition of a single speaker among multiple (and possibly louder) interfering speakers [30].

2.2 Speech Production

In order to formulate a mathematical model for speech production, it is imperative to first consider and understand the physical speech generation process. Roughly speaking, a speech signal is a pressure wave that is generated by movements of the anatomic structures that make up the human speech production system. The speech organs and muscles involved in the production process are shown in Figure 2.2, which provides a cross-sectional views of these components. The three main groups of speech organs are the lungs, the larynx (commonly called the voice box), and the vocal tract.

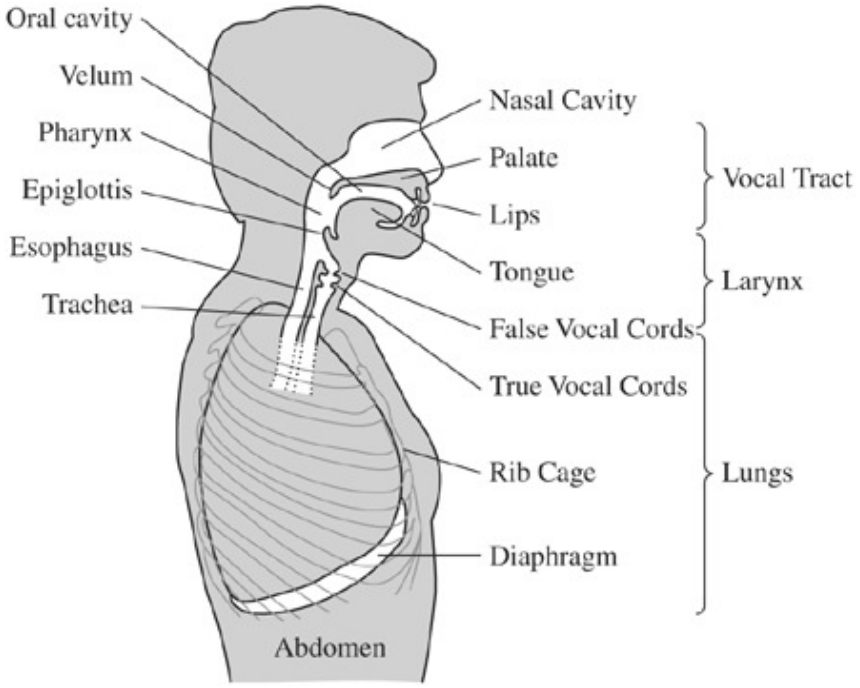


Figure 2.2: Cross-sectional view of the anatomy of speech production [30].

The main functionality of the lungs is to deliver air pressure to the speech production process. Inhaling air causes the air pressure in the lungs to decrease, whereas exhaling air causes the air pressure in the lungs to increase as the chest cavity is reduced. The increased air pressure forces air to flow into the larynx via the trachea [31].

In the larynx stage, voice is generated by vibrations in the vocal tract or simply by the absence of vibrations. The larynx is built from masses of muscles and ligaments that control the vocal folds (or vocal cords). In between the two folds there is a opening called the glottis. The vocal folds can assume a number of states; when the folds are vibrating in a periodic manner, the folds are said to be in

a voiced state. The resulting air waveform has a time period T_0 known as the *pitch period*, and the reciprocal value $\frac{1}{T_0}$ is known as the *fundamental frequency*. When the vocal folds are not vibrating, they are in a unvoiced state and the resulting speech signal has a non-periodic time representation. Vowels also known as voiced sounds are produced in the voicing state, whereas unvoiced sounds that include the majority of consonants are produced in the unvoicing state [31].

The air flow finally reaches the vocal tract after passing the vocal folds. At this stage the oral and nasal cavity are located. The vocal tract can be modeled as a time-varying linear filter that shapes the input waveform to generate different types of sound output. The characteristics of the filter changes with the placement of the vocal tract organs, i.e. the lips, tongue, jaw, etc [31]. The produced sound is a composition of different sound units, of which the smallest units (that are capable of conveying a distinct meaning) are called phonemes.

2.3 Speech Signal Representation and Speech Modeling

In general terms, speech signals can be attributed to number of different properties and characteristics. It is well-known that speech signals do not change abruptly, rather they change continuously over time. Besides being time variant, speech signals are high non-stationary, meaning that their statistical properties vary with time [31]. This becomes apparent by examining the spectrogram of a speech signal, in that it reveals how the power at different frequencies changes over time, and thus the variation in its second order statistics. An example waveform is shown in Figure 2.3 along with its computed spectrogram.

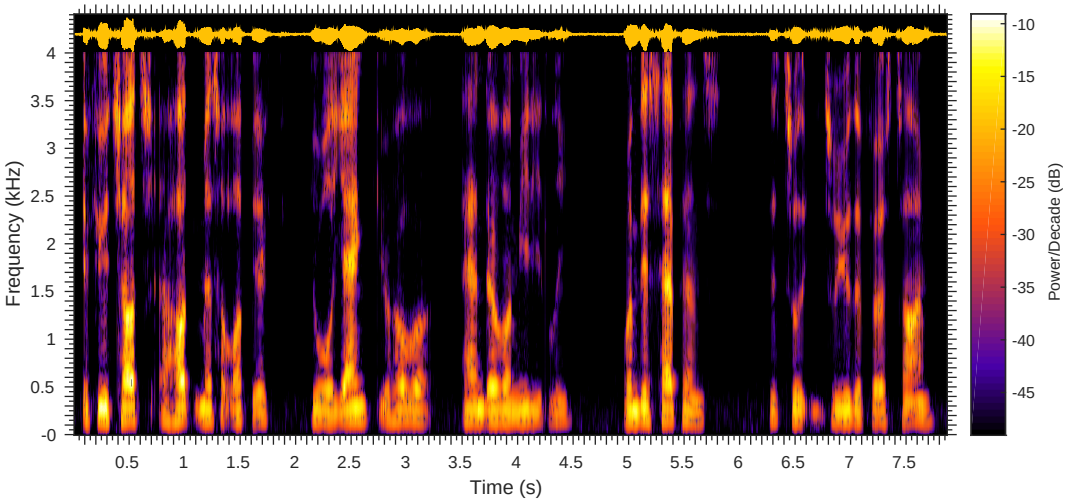


Figure 2.3: A spectrogram generated from the speech waveform (plotted in yellow), illustrates the change of power over time, and thereby the non-stationary property of speech signals.

When analyzing non-stationary signals, a common practice is to divide speech signals into small time frames of 10-30 ms duration, whereby each time frame obtain spectral properties that are approximately stationary. Therefore speech is generally assumed to be quasi-stationary over short time periods, which renders Fourier analysis applicable [31]. Especially the short-time Fourier transform (STFT) is motivated by analyzing the spectral content of non-stationary signals. The basic idea is to split the time domain signal into successive overlapping frames of some sample length L, subsequently the STFT applies an analysis windows for which the DFT is computed. While the windows moves along the time axis the discrete Fourier transform (DFT) is computed at each instance, thereby a time-frequency representation is obtained that exposes the variance in the frequency content over time. The core principle behind the STFT is illustrated in Figure 2.4. Formally, the discrete case of the STFT is defined as a function of a time index n and a frequency index ω:

$$X(n,\omega) = \sum_{n=-\infty}^{\infty} x(m)w(n-m)e^{-j\frac{2\pi}{N}\omega m} \tag{2.1}$$

where $x(m)$ is the discrete time signal to be analyzed, $w(n)$ is some analysis window, and N can be interpreted as the frequency resolution. Given the STFT, it is straight forward to generate the spectrogram from the following definition $|X(n,\omega)|^2$ [31]. For frequency analysis of stationary stochastic signals, it is sufficient to compute the power spectral density (PSD) that corresponds to a single "time-slice" of the spectrogram or similarly a single plot of the ones shown in the bottom of Figure 2.4.

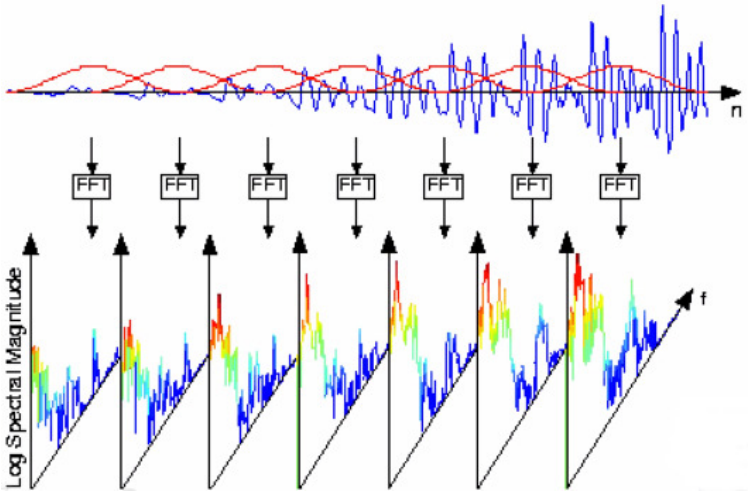


Figure 2.4: The figure illustrates the basics behind the STFT; a window function (red) extracts a small sequence of the time signal (blue) that is sufficiently small to be viewed as stationary. The analysis window moves along the time axis, and at each instance the DFT (or fast Fourier transform (FFT)) is computed.

2.3.1 Speech Production Model

There exist considerable differences in the signal characteristics of different spoken sounds or words. Depending on the state of the vocal folds during speech production, the excitation is periodic or non-periodic. In particular, sound segments produced in the voicing state (i.e. the vocal folds vibrate) are quasi-periodic by nature, whereas unvoiced sounds produced in the unvoicing state (i.e. vocal folds do not vibrate) are more aperiodic and noise-like [31]. These properties are widely reflected in the speech production model presented in Figure 2.5 that is an engineering model of the human speech production process. The acoustic theory behind the model, assumes the production process to be a linear system consisting of a source and filter. As mentioned earlier, the vocal tract is modeled as a time-invariant linear filter over a time frame and its parameters (e.g. the filter coefficient) are determined by a set of resonances frequencies. Acoustic tubes of any kind (and here in particular the vocal tract) have natural resonances, which are a function of its physical shape [32]. The resonance frequencies in the context of speech production are also referred to as formants, as they form the speech spectrum [1].

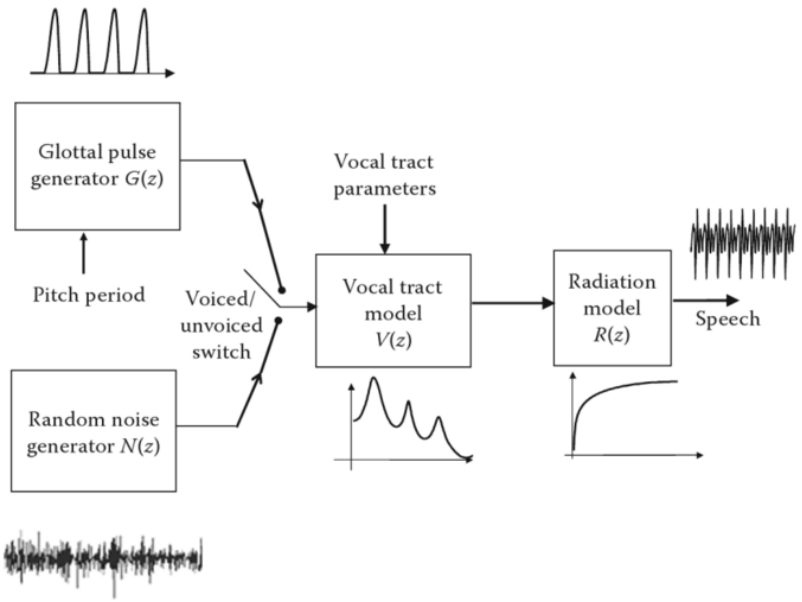


Figure 2.5: An engineering model of the human speech production system [31].

The linear filter is fed by a source signal that is governed by a switch and thus given either from a pulse generator or a random noise generator depending on the voiced state. In the simplest case, the vocal tract filter is approximated by an

[1] "Formant" originates from the Latin word *formāre* (v.) which means to shape [33].

all-pole filter, with the following z-domain representation

$$V(z) = \frac{g}{1 - \sum_{k=0}^{p-1} a_k z^{-k}} \tag{2.2}$$

where g is the system gain, p is the number of coefficients, and $\{a_k\}, k = 0, \ldots, p - 1$ are the all-pole coefficients. The output generated from the vocal tract filter is eventually passed on to another filter that simulates the radiation effect from the lips. Such a filter usually takes the form

$$R(z) = 1 - z^{-1}, \tag{2.3}$$

and is designed to add a 6 dB/octave tilt to the original source spectra, i.e. it introduces about a high-pass boost [31].

In the voiced case, the z-transform of the production system is defined as a product of the transfer functions that describes the glottal pulse input $G(z)$, the vocal tract filter $V(z)$, and the lip radiation filter $R(z)$:

$$X(z) = G(z)V(z)R(z) \tag{2.4}$$

For the unvoiced case, the source signal is instead modeled as a random noise sequence with a flat spectrum (i.e. white noise), thereby the new output from the lips becomes:

$$X(z) = N(z)V(z)R(z) \tag{2.5}$$

where $N(z)$ is the z-transform of the noise sequence.

Note that the presented source-filter model is a simplistic linear model of the speech production system, which assumes the excitation signal to be independent from the vocal tract system, i.e. the filter that shapes the sound. But in fact the coupling between source and filter is non-linear and far more complex than such. While only an approximation, the source-filter model underlies nearly all speech recognition system, mainly due to its relative simplicity in modeling speech signals [31].

2.4 Acoustic Feature Representation

The auditory information contained in a speech signal can be represented in multiple ways, one classical approach is the time domain waveform, which says very little about the phonological content of the signal. Alternative representations (such as the ones introduced in Section 2.3) include the amplitude spectrum, power spectrum and so forth. However, conventional ASR systems resort to acoustic features for representing speech signals in a concise and compact manner by preserving

only its key aspects, while discarding other less important information. The process of transforming the original speech sequence into a set of dimension-reduced features is known as *feature extraction*, but sometimes also referred to as *front-end processing* or *acoustic preprocessing* [38]. In the feature extraction process, the short-term stationarity property is taken into account, as a feature vector is computed for each time frame. Hence a full speech signal is represented by a sequence of feature vectors, rather than just a single feature vector.

It is desired to have acoustic features that preserve the information needed to determine the phonetic class, while being general of nature, i.e. they are invariant to e.g. the speaking person, speaker accent, speaking rate, background noise, etc. The features need to represent the phonetic content in each segment in such a way that segments of similar characteristics can be grouped together by simply matching their features [39].

2.4.1 Mel Frequency Cepstral Coefficients

The Mel frequency cepstral coefficients (MFCCs) were introduced by S.B. Davis and P. Mermelstein in the 1980's [34], and have been state-of-the-art in the field of ASR since then. The type of acoustic features that preceded the MFCCs are mainly linear prediction coefficients (LPCs) and linear prediction cepstral coefficients (LPCCs).

The MFCCs as described in the following are inspired by the human auditory system. This includes a mapping from normal frequencies to *Mel frequencies*. The Mel frequency scale models the non-linear pitch perception characteristics of the human ear. The relationship between the Mel frequencies F_{mel} and the actual frequencies F_{Hz} is approximated by the following expression [40]:

$$F_{\text{mel}} = \frac{1000}{\log(2)} \left(1 + \frac{F_{\text{Hz}}}{1000}\right) \tag{2.6}$$

As it an be seen from the plot of the above function in Figure 2.6, the curve is approximately linear below 1000 Hz and logarithmic above. The Mel scale is generated using an experimental approach, where a number of test objects were asked to adjust a tone at some fixed reference frequency until they perceived the tone to have reached half its frequency [40]. The experiment has among other things revealed that humans are better at distinguishing frequencies in the low-end of the frequency scale as compared to the high-end.

Recall from Section 2.3.1 that speech signals can be modeled as an excitation signal $e(t)$ filtered by a time-varying filter with impulse response $v(t)$ that represents the shape of the vocal tract. Hence the speech signal over a single frame denoted $s(t)$, can be defined as follows (while neglecting the radiation effect) [43]:

$$s(t) = e(t) * v(t) \tag{2.7}$$

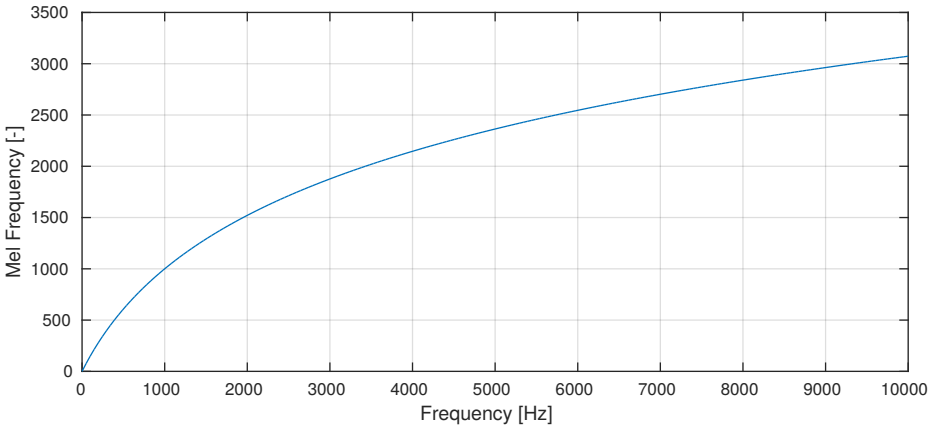


Figure 2.6: The Mel scale plotted as a function of the actual frequency scale.

Since the characteristics of the vocal filter shapes the produced sound, it is desired to accurately determine these characteristics, to thereby identify the phonetic content in the speech signal. Now, the challenging part lies in the fact that only the speech signal is observable in most cases, the parameters of the speech model and the vocal filter are not directly obtainable. One approach to overcome this issue used in MFCC extraction settings, is to apply cepstral analysis to deconvolve the excitation signal from the filter response. This involves taking the logarithm of the frequency domain representation of Equation (2.7), whereby two additive parts are obtained [43]:

$$\log S(f) = \log\left(E(f) \cdot V(f)\right) = \log E(f) + \log V(f) \tag{2.8}$$

Now, the final separation into the excitation model and the impulse response of the vocal filter can be carried out by applying the inverse DFT to the log spectrum. This particular operation transforms the signal to the *quefrency* domain or cepstral domain, in which the vocal tract components are located near the low quefrency regions, while the excitation model components are located at the higher quefrency region. A general mathematical formulation of the Cepstrum $c(n)$ is outlined in Equation (2.9) [40], and is defined as the inverse Fourier transform of the log spectra of the speech signal.

$$c(n) = \mathcal{F}^{-1}\left\{\log\left|\mathcal{F}\left\{s(n)\right\}\right|\right\} \tag{2.9}$$

where $\mathcal{F}\{\cdot\}$ denotes the DFT. The terminology used for the word *Cepstrum*, originates from reorganizing the first part in the word *spectrum*.

MFCC features combine the Mel frequency scale with cepstral analysis in a multi-step procedure outlined in Figure 2.7 [41]. Each step in the feature extraction

process mimics aspects of the human auditory system, due to its effectiveness in perceiving and recognizing sounds.

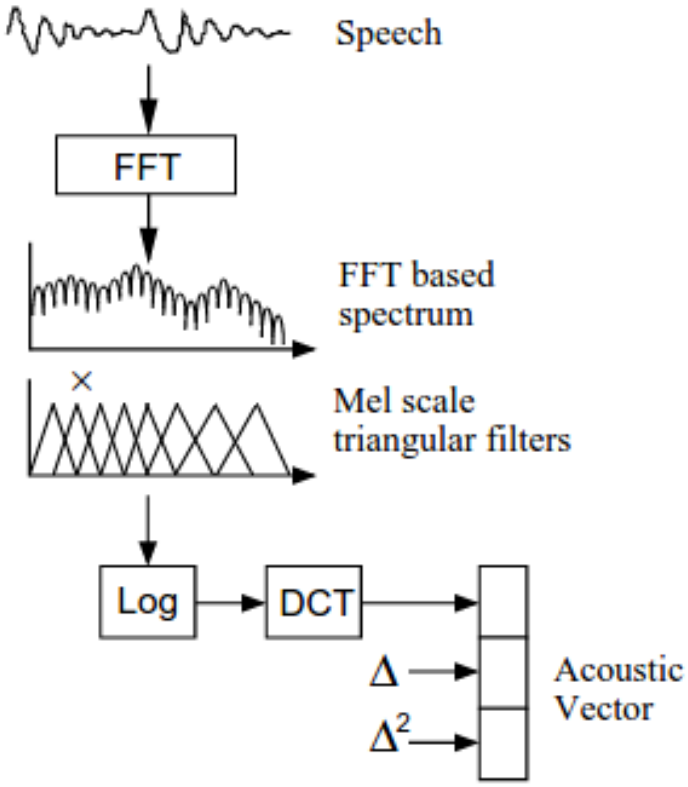


Figure 2.7: MFCC calculation procedure [41].

The procedure for computing the MFCC features can be categorized into four major steps, assuming that pre-emphasis framing of the speech signal has been carried out in advance [41]:

1. *Short Time Fourier Transform*: STFT is applied to obtain a power spectrum of each frame.

2. *Mel frequency filter bank*: In order to emphasize the perceptually meaningful frequencies and smoothen the spectrum, a Mel filter bank is applied, comprising M triangular filters distributed on the Mel frequency scale.

3. *Logarithm operator*: Once the filter bank energies have been extracted, the logarithm is applied to the results. This is due to the fact that loudness is perceived by humans on a non-linear scale.

4. *Discrete Cosine Transform*: As a final step the DCT is applied on the log filter bank energies. It serves the purpose of decorrelating the filter bank energies,

as some sort of correlation between the energies are expected due to the overlapping filters. The decorrelation effect of the DCT is beneficial in the sense that it makes the feature vector more compact, by discarding the high-order DCT coefficients.

In the last step, the inverse Fourier transform has been replaced with the DCT (as compared to the definition in Equation (2.9)). However, the optimal decorrelator is theoretically the Karhunen-Loéve (KL) transform, as it exhibits the best energy compactness properties, meaning that it has the ability to concentrate the energy in as few coefficients as possible. The main obstacle with the KL transform is its data dependency, therefore the KL transform is approximated by the DCT in speech applications [42].

While MFCCs contain static information about a speech frame, other types of features can be used in conjunction with the MFCC features (as outlined in Figure 2.7) to add dynamic information. This pertains particularly to delta and delta-delta coefficients, which explore the rate of change between successive frames. The addition of delta features to the MFCC coefficients have proven to strongly improve detection accuracy, and thus they are found in nearly all ASR systems [35].

2.4.2 Delta and Delta-Delta Features

Unlike the MFCC features, the delta and delta-delta coefficients also referred to as differential and acceleration coefficients contain information pertaining to the dynamics of the speech signal, i.e. they infer about the trajectories of the MFCC coefficients over time. Generally, delta coefficients are defined as the difference across m successive frames represented by their corresponding static features [43]:

$$\Delta [n,k] = c [n+m,k] - c [n-m,k] \tag{2.10}$$

where $\Delta [n,k]$ and $c [n,k]$ are the delta coefficient and the static feature coefficient of the n^{th} frame and the k^{th} frequency bin, respectively. Delta-delta features $\Delta^2 [n,k]$ are calculated by substituting the static feature in Equation (2.10) with the delta-feature [43]. The delta and delta-delta coefficients are interpreted as first and second order time derivatives, respectively, as they characterize the temporal variation in the signal [36].

An alternative and perhaps more applied definition of the delta coefficient, uses regression analysis to fit the slope of a straight line to the acoustic features for a number of successive frames. This definition found in [37] can be expressed mathematically as:

$$\Delta [n,k] = \frac{\sum_{t=-m}^{m} tc [n+m,k]}{\sum_{t=-p}^{p} t^2} \tag{2.11}$$

In comparison to the definition in Equation (2.10), the later definition includes more features in the computation of the delta coefficient, and therefore is the preferred choice for estimating the frame rate. The delta-delta coefficients are computed in a similar manner as in Equation (2.10), i.e. by replacing the static features with the delta features.

Appending the MFCC trajectories to the original feature vector (of MFCC coefficients) have proven to increase recognition performance, as compared to the case where MFCC features are used in isolation. However, it should be noted that the performance improvement provided by appending the delta coefficients drops at lower signal to noise ratio (SNR) values, and therefore do not always achieve good robustness against noise and reverberation [35]. Anyhow, this will no affect the choice of applying these features for keyword detection.

Chapter 3

Keyword Spotting using Template Matching

This chapter presents a keyword spotting algorithm that complies with the platform constraints and the delimitations presented in Section 1.3 to ensure implementational feasibility. The work presented in [45] will serve as the baseline system, around which modifications and simplifications will be introduced on a smaller scale. Unlike the conventional statistical-based spotting methods, the work in [45] proposes a template-based and unsupervised approach, i.e. it follows the first KWS paradigm presented in Section 1.2, and further that it does not require annotated speech data for training. This chapter covers the design of both the front-end and back-end aspects of the system that includes feature extraction, pattern matching, scoring and decision making. Finally, an evaluation of the spotting performance will be carried out by testing the keyword spotter under different acoustic scenarios.

In Section 3.1, a system overview is provided and followed by an explanation of the different stages in the keyword spotter. The system is divided into a number of training and test subsystems. The subsequent sections will provide a more detailed description of each subsystem, and cover the applied methods and techniques along with their underlying theory. In Section 3.2, the applied pre-processing techniques on the raw speech recordings are explained, comprising signal normalization, speech framing, and voice activity detection. Section 3.3 covers the feature extraction phase, with particular emphasis on the Gaussian mixture model and its application to KWS. Section 3.4 deals with pattern matching and classification of the test signal using dynamic programming methods, which includes DTW and its variant segmental DTW (SDTW). Finally scoring and decision making is covered in Section 3.5.

3.1 System Overview

The overall system as depicted in Figure 3.1 is partitioned into two training subsystem and a test subsystem. The basic principle behind the presented keyword spotter highly resembles the first approach put forth in Section 1.2 and Figure 1.1, and has two major steps. In the first step templates of keyword(s) are generated and stored during an offline training phase.This is followed by an online test procedure, where features from a keyword template and a target utterance are extracted and compared against each other, in order to determine possible occurrences of the keyword term in the target utterance. The described approach assumes that a number of audio examples of the keyword(s) are attainable from the native system user. That is a fundamental prerequisite for the system, since it searches the test string for the presence of template instances based on the known audio examples.

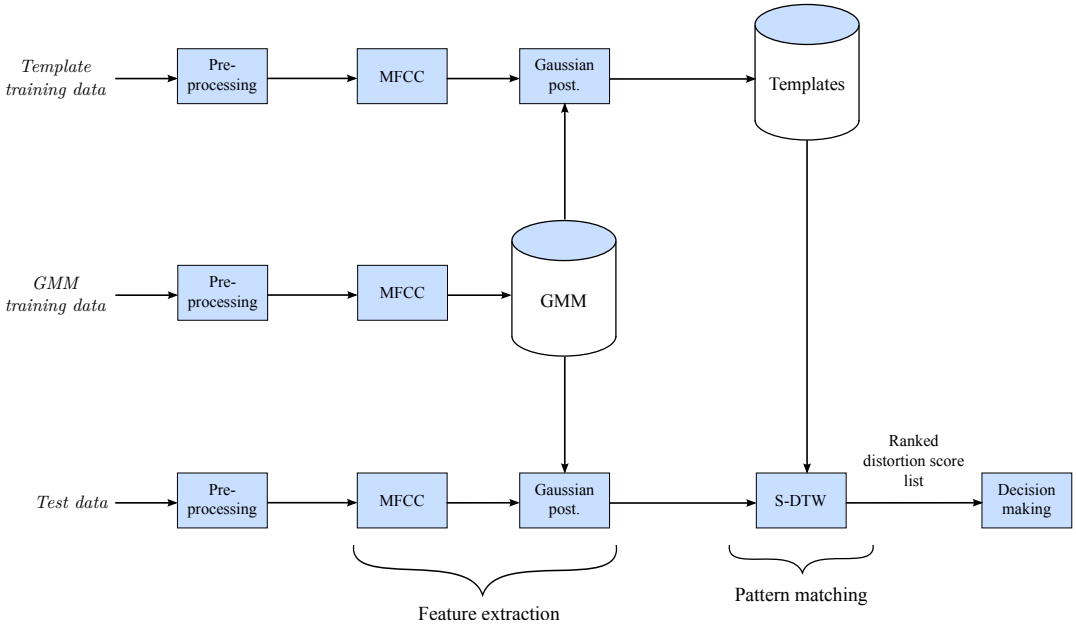


Figure 3.1: This figure provides an overview of the different system blocks which constitute the keyword spotter presented in [45], and includes both test and training procedures. The keyword spotter combines Gaussian posteriorgrams with segmental DTW.

In Figure 3.1 the two training phases are outlined. The first training phase is used for training the Gaussian mixture model (GMM) from a set of observed acoustic features. An often used approach for GMM training is the expectation-maximization (EM) algorithm, and will similarly be the approach used for our system. The GMM is a probabilistic model used for posterior feature extraction, it transforms the traditional cepstral-based features to a type of posterior features.

In this context, the GMM can roughly be regarded as a variant of a phone recognizer, which is trained to estimate the phone posterior probabilities based on acoustic features. As opposed to the conventional phone recognizer, each cluster in the GMM typically represents an unspecified independent acoustic unit rather than a specific phonetic class. Hence the mixture of Gaussians aim at identifying the underlying probability distribution of these acoustic subword units, instead of explicitly modeling the phonetic units. The second training subsystem is used to generate keyword templates. As with the first training phase, the training data is preprocessed before the acoustic features are extracted. An MFCC feature vector is extracted from each frame, hence an entire speech signal is necessarily represented by a sequence of feature vectors stored in a feature matrix. These features are subsequently fed into the GMM. The concatenation of posterior probability vectors obtained from feeding the cepstral-based feature matrix into the GMM is denoted a *Gaussian posteriorgram*, and contains the posterior probability vector for each frame. The computed posteriorgrams from each keyword are then stored in a template database.
Once training of the keyword spotter has been fulfilled, an online test routine can be applied. It works by first preprocessing the test input before being passed to any subsequent detection routines. Preprocessing implies performing energy normalization and next partitioning the speech signals into overlapping frames before voice activity detection (VAD) finally is applied. The VAD aims at detecting and discarding the frames in which speech is absent, therefore it is occasionally referred to as *silence suppression*. Subsequently, the acoustic feature matrix is generated from a series of feature vectors. In the posterior feature handling, the feature matrix is converted into a Gaussian posteriorgram. Given one or more keyword templates (stored in the template database), SDTW is used to compare the template posteriorgram against the test posteriorgram to detect any common patterns. The SDTW is a simplified variant of the conventional DTW algorithm which implements a sliding window approach. The test template is slided across the entire test utterance in overlapping windows. At each window instance a similarity/distortion score is computed between the keyword template and the windowed signal. The resulting distortion scores are then collected and ranked accordingly to find the best alignment path. As a final step, a decision threshold is applied to indicate the extent of match and infer about the presence of a keyword.

3.2 Preprocessing

The major component in the preprocessing stage is undoubtedly the VAD, which seeks to eliminate all silence regions in the applied speech data. A VAD algorithm typically generates a binary decision output based on each time frame, to indicate whether speech is present in that particular frame. The motivation behind intro-

ducing a VAD into the system pertains to a general clustering problem experienced in the presence of non-signal and noise components in the training data. This issue will be manifested in the GMM output, where the vast majority of the probability mass is concentrated in only a few Gaussian component, thereby creating unbalanced posteriorgrams [45]. More details on this particular issue is provided in Section 3.3.3.

The used VAD is found and described in [46], hence only a brief introduction to the main system functionality is given in the following. The applied VAD is based on an unsupervised two-pass segment-based method. A system overview of the VAD is shown in Figure 3.2.

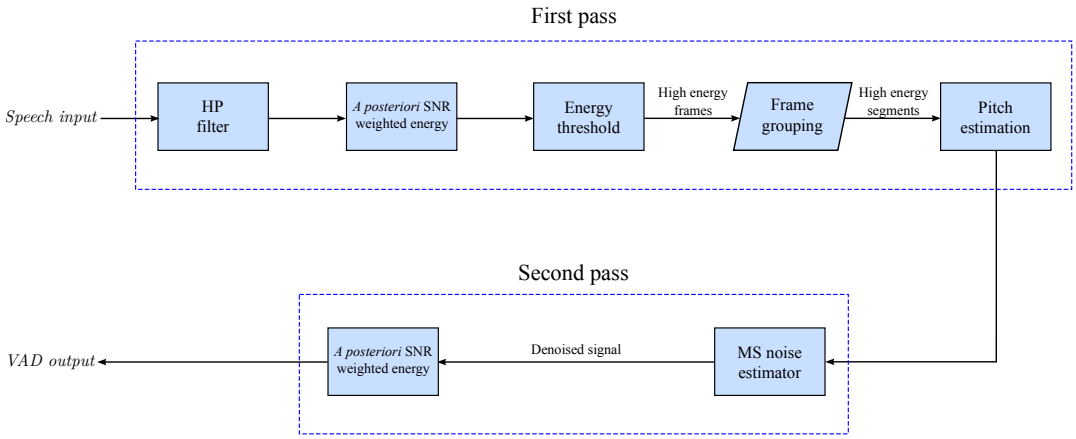


Figure 3.2: VAD system block overview.

In the first pass, the input signal is filtered through a high-pass filter with a cut-off frequency of 60 Hz. To detect the high-energy segments in the filtered signal, a posteriori signal-to-noise-ratio (SNR) weighted energy difference is applied. If the difference measure between two consecutive frames exceeds a predefined threshold, the frames are considered to be of high energy. Subsequently, high-energy frames that occur in consecutive order are grouped together to form high-energy segments. Next, pitch detection is applied to each segment; if no pitch is detected within the segment, it is considered to be a high-energy noise segment. In the second pass, noise reduction is applied using a modified minimum statistics (MS) noise estimator to eliminate stationary noise from the speech signal, while the high-energy noise segments are set to zero. As a final measure, the denoised signal generated from the MS noise estimator is passed onto a subsequent block, where the a posteriori SNR weighted energy difference is calculated to make voice activity detection.

3.3 Feature Extraction and Posterior Feature Handling

This section describes the basic theory behind GMMs and its application to KWS, moreover we motivate its use for posterior feature handling. The speech analysis for extracting static acoustic features (MFCCs) and dynamic features (deltas and double deltas) is already described in Section 2.4. Hence, this section will focus mainly on posterior feature processing.

An essential part of every ASR system is to model the individual phonetic sound units within the test signals prior to any detection routines. Segmentation of the speech signals into its phonetic classes is done either explicitly or implicitly. The baseline system as presented in Figure 3.1 [45] relies heavily upon the research by Hazen et al. in [47] in the aspect of posterior feature handling. In the later they apply an independently trained phonetic recognizer to model the speech, from which a *phonetic posteriorgram* is extracted. The phonetic posteriorgram can be regarded a phonetic class time versus matrix as exemplified in Figure 3.3, whose columns represent a probability vector containing the posterior probabilities for a set of predefined phonetic classes at a specific time frame. These values are derived directly from the acoustic likelihood scores as given by the phonetic recognizer.
However, the approach in [45] resorts to another modeling strategy which uses implicit phonetic modeling, meaning that the posterior features are rendered a symbolic phonetic representation rather than a direct representation of the phonetic classes. Implicit modeling typically relies on unsupervised clustering to classify acoustic features, during both training and testing. For the particular system presented in Figure 3.1, subword unit clustering of the speech data is handled by the GMM. As a consequence of substituting the phonetic recognizer with a GMM, the phonetic posteriorgram effectively turns into an Gaussian posteriorgram. While the phonetic posteriorgram represents the likelihood scores for each phonetic class for a certain time frame, the Gaussian posteriorgram models the posterior probabilities for each Gaussian component. Using a completely unsupervised approach, i.e. without any transcription information of the speech data, the GMM is trained to label both test and template instances with a Gaussian posteriorgram. Using an unsupervised training approach adds the advantage that the system can be adapted to any language, rendering the system an appealing choice for e.g. low-resource languages.

Besides the ability to cluster speech signals into their subword units, the posterior features have other desirable properties that render them an appealing choice for pattern matching. In particular, the posterior features exhibit an added stability and robustness to noise as compared to the traditional MFCC features [48]. Motivated by their desired characteristics, posterior feature handling using GMMs will

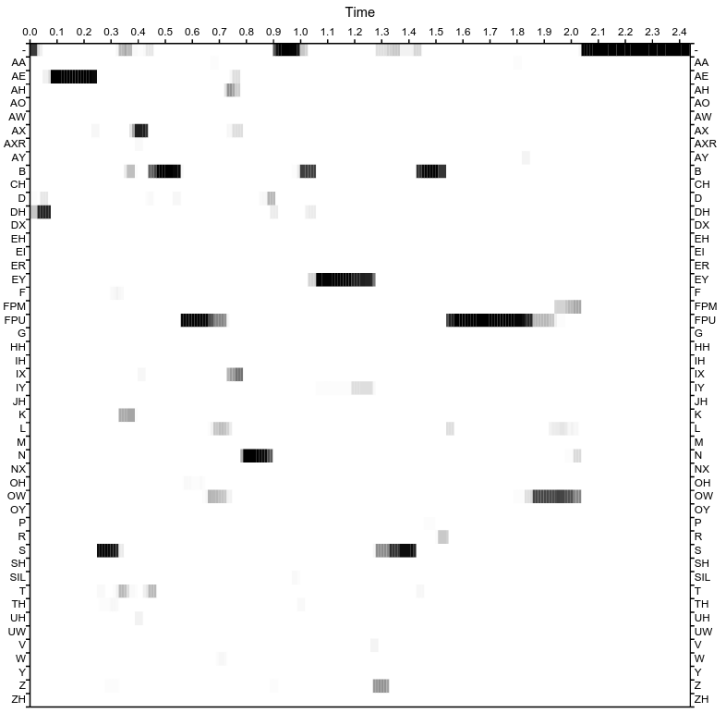


Figure 3.3: A phonetic posteriorgram of the spoken phrase "*Basketball and baseball*". The horizontal axis shows the time in seconds, while the vertical axis represents the individual phonetic sound units. The proportion of darkness in the figure expresses the posterior probability values, where black pixels corresponds to one and white to zero.

be reviewed to the application of template matching.

3.3.1 Gaussian Mixture Models

Many natural occurring phenomena including speech have successfully been modeled using the Gaussian distribution. GMMs are generally considered a powerful tool for generating smooth approximations to any arbitrarily-shaped distribution [51]. In particular, mixture of Gaussians has proven effective in modeling the probability distribution of the underlying acoustic units in a speech signal when combined with e.g. MFCCs [49]. This is mainly due to the fact that feature vectors appear in clusters in their respective feature space, rather than e.g. being centered around a single point. Therefore a mixture of Gaussians has suitable modeling capabilities for this particular application, since it has the ability to model each cluster by a distinct Gaussian component. This is e.g. used in [45], where the number of assigned Gaussian components roughly corresponds to the number of underlying phonetic units.

In general, a GMM can be described as linear combination of K normal distribu-

tions, with a probability density function (PDF) that is defined as follows:

$$p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}\left(\mathbf{x} \mid \boldsymbol{\mu}_k,\ \boldsymbol{\Sigma}_k\right) \tag{3.1}$$

where the model parameters of the k^{th} component are given as

$$\lambda_k = \left\{\pi_k,\ \boldsymbol{\mu}_k,\ \boldsymbol{\Sigma}_k\right\}. \tag{3.2}$$

The parameter π_k is known as the mixing coefficient and expresses the prior probability that an observation originates from the k^{th} Gaussian component. The remaining model parameters of the k^{th} Gaussian component are given by the mean vector $\boldsymbol{\mu}_k$ and the covariance matrix $\boldsymbol{\Sigma}_k$. The mean vector represents the expected feature vector within the k^{th} component, while the covariance matrix represents the variability and correlations of the feature vectors in the k^{th} component. The mixture components are unimodal multivariate Gaussians, and hence can be described by the following expression for some D-dimensional input vector $\boldsymbol{x}$:

$$\mathcal{N}\left(\mathbf{x} \mid \boldsymbol{\mu}_k,\ \boldsymbol{\Sigma}_k\right) = \frac{1}{(2\pi)^{\frac{D}{2}}\left|\boldsymbol{\Sigma}_k\right|^{\frac{1}{2}}} \exp\left[-\frac{1}{2}\left(\mathbf{x}-\boldsymbol{\mu}_k\right)^{\mathsf{T}} \boldsymbol{\Sigma}_k^{-1}\left(\mathbf{x}-\boldsymbol{\mu}_k\right)\right]. \tag{3.3}$$

Given a set of feature vectors (e.g. from the training set), the GMM component parameters can be estimated using a maximum likelihood (ML) approach which maximizes the probability of observing the feature set. One such approach is the EM algorithm which estimates the model parameters in a unsupervised and iterative manner. Provided feature vectors derived from a training set of speech data, the EM algorithm iteratively refines the Gaussian model parameters to maximize the likelihood that the model fits the training data distribution. This type of training holds no requirements of annotated speech data, and converges to the final solution in only few iterations [50].

3.3.2 Maximum Likelihood Parameter Estimation

Given a training set of feature vectors, the aim of the training procedure is to estimate the GMM model parameters $\lambda_k = \{\pi_k,\ \boldsymbol{\mu}_k,\ \boldsymbol{\Sigma}_k\}$, $k = 1,\ \ldots, K$ such that they best match the distribution of the training set in some sense. There exist several methods for carrying out this training task, but the most applied and well-established technique for GMM training is ML estimation [51]. The goal of ML training is to estimate the model parameters which maximize the GMM likelihood of observing the sequence of training vectors. The ML estimate of the parameter set can be expressed by the following optimization problem

$$\hat{\lambda} = \arg\max_{\lambda} \quad \mathcal{L}\left(\lambda \mid \mathbf{X}\right) \tag{3.4}$$

where $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ is the training data given by a set of feature vectors, and $\mathcal{L}(\mathbf{X} \mid \lambda)$ is the log-likelihood function of the GMM which has the definition

$$\mathcal{L}(\lambda \mid \mathbf{X}) = \log p(\mathbf{X} \mid \lambda) = \sum_{n=1}^{N} \log p(\mathbf{x}_n \mid \lambda). \tag{3.5}$$

Thus ML estimation seeks to maximize the probability of the observed feature vectors by adjusting the model parameters λ. Unfortunately, there exist no analytical solution to the problem in Equation (3.15). Instead ML parameter estimation can be conducted iteratively by using a variant of the EM algorithm, which is outlined in Algorithm 1. The basic idea behind the algorithm, starting from some model $\lambda^l = \{\pi^l, \boldsymbol{\mu}^l, \boldsymbol{\Sigma}^l\}$, is to estimate a new model λ^{l+1} such that $p(\mathbf{X} \mid \lambda^{l+1}) \geq p(\mathbf{X} \mid \lambda^l)$ [51]. Thereby the new model becomes the current model of the next iteration. Using this approach the model parameters of each Gaussian component is recalculated iteratively and runs until the convergence threshold is reached, i.e. until the estimates of λ have stabilized.

Algorithm 1: EM algorithm used for GMM training.

Initialization: $\boldsymbol{\mu}_k^0$, $\boldsymbol{\Sigma}_k^0$, π_k^0
for $l = 0, 1, \ldots$ **do**
  for $k = 1, \ldots, K$ **do**
    E-step;

$$\gamma_{n,k} = \frac{p(\mathbf{x}_n \mid \boldsymbol{\mu}_k^l, \boldsymbol{\Sigma}_k^l)\pi_k^l}{\sum_{j=1}^{K} p(\mathbf{x}_n \mid \boldsymbol{\mu}_j^l, \boldsymbol{\Sigma}_j^l)\pi_j^l}$$

    M-step;

$$\boldsymbol{\mu}_k^{l+1} = \frac{\sum_{n=1}^{N} \gamma_{n,k}\mathbf{x}_n}{\sum_{n=1}^{N} \gamma_{n,k}}$$

$$\boldsymbol{\Sigma}_k^{l+1} = \frac{\sum_{n=1}^{N} \gamma_{n,k}\left(\mathbf{x}_n - \boldsymbol{\mu}_k^{l+1}\right)\left(\mathbf{x}_n - \boldsymbol{\mu}_k^{l+1}\right)^{\mathsf{T}}}{\sum_{n=1}^{N} \gamma_{n,k}}$$

$$\pi_k^{l+1} = \frac{\sum_{n=1}^{N} \gamma_{n,k}}{N}$$

  end
end

After initialization, the EM algorithm alternates between two steps in each iteration. The dual-step procedure ensures that the GMM likelihood function of the observed feature set increases monotonically for each iteration. In short terms, the functionality of each step may be summarized as follows:

- **E-Step**: To compensate for the hidden nature of the data labels, i.e. the affiliation of a feature vector $\mathbf{x}_n$ to the K Gaussian components is unknown, the algorithm resorts to (soft) label estimation using posterior probabilities.

To find which Gaussian is most likely to have generated a feature $\mathbf{x}_n$, the posterior probability that each Gaussian generated $\mathbf{x}_n$ is found, i.e. $\gamma_{n,k}$ for $k = 1, \ldots, K$ are computed.

- **M-Step**: Assuming that the data points was generated according to the probability distribution from the E-step, the parameters of each Gaussian $\lambda_k = \{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}$ are updated to maximize the probability that it would generate the data it is currently responsible for.

The use of the EM algorithm is illustrated in Figure 3.4, where it is being applied on the Old Faithful data set [54] and modeled by a mixture of two Gaussians. Figure 3.4(a) depicts the Gaussians in their initial state, whereas the first E and M step are shown in Figure 3.4(b) and 3.4(c), respectively. The plots in Figure 3.4(b), 3.4(c), and 3.4(c) shows the results after 2, 5, and 20 EM iterations.

The model parameters are often initialized at random, and followed by clustering using the K-means algorithm. Combined with K-means clustering, ten iterations of the EM algorithm will typically suffice for parameter convergence [52].
For cepstral-based features such as MFCCs, the computational requirements are often relaxed by assuming the covariance matrices to be diagonal. This assumption is valid as the DCT applied in the extraction process of the MFCC features acts as a decorrelator [64].

3.3.3 Gaussian Posteriorgrams

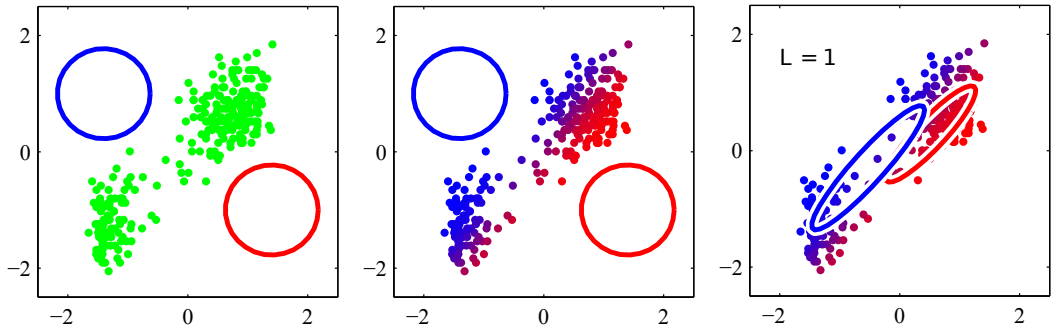
Gaussian posteriorgrams are a novel type of posterior feature representation, which have been adopted by recent template-based KWS systems such as the one in [45], while [48] and [55] apply similar posterior feature representations. Whereas the phonetic posteriorgram is defined as a phone versus time matrix, the Gaussian posteriorgram is a time versus class matrix, where the classes refer to the Gaussian components. In other words, a Gaussian posteriorgram is a probability vector containing the posterior probabilities of each Gaussian component for a number of speech frames. To state it formally, for a speech signal of N feature vectors $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$, the corresponding Gaussian posteriorgram is defined as [45]

$$GP(\mathbf{X}) = [\boldsymbol{q}_1, \boldsymbol{q}_2, \ldots, \boldsymbol{q}_N]^\mathsf{T} \tag{3.6}$$

where $\boldsymbol{q}_n$ is the posterior probability vector of the n^{th} feature vector for all Gaussians and is calculated by

$$\boldsymbol{q}_n = [p(C_1 \mid \mathbf{x}_n), p(C_2 \mid \mathbf{x}_n), \ldots, p(C_K \mid \mathbf{x}_n)]^\mathsf{T} \tag{3.7}$$

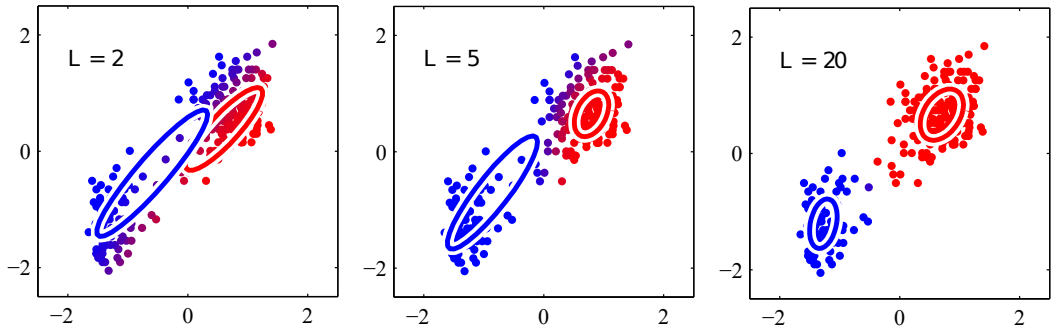
where C_k denotes the k^{th} Gaussian component, and K represents the total number of Gaussian components. Using the posteriorgram representation, a speech signal



(a) Unlabelled data points shown in green, along the initial state of the two mixtures shown as the red and blue contours.

(b) The result of the initial E-step, where each point is depicted as a mixture of red and blue to signify the posterior probability values of each Gaussian.

(c) The result of the initial M-step, where the component parameters are reestimated with respect to the probability distribution of each point.



(d) Results after 2 EM iterations.

(e) Results after 5 EM iterations.

(f) Results after 20 EM iterations, where the model parameters are close to convergence.

Figure 3.4: Selected steps of the EM-algorithm applied on the Old Faithful data set modeled with two Gaussian mixtures. [53].

of N frames is transformed into a matrix of dimensions $K \times N$, where each column represents a single frame.

An example of a Gaussian posteriorgram is visualized in Figure 3.5 which plots the speech frames along the horizontal axis, while the Gaussian components are shown on the vertical axis.
Generating Gaussian posteriorgrams is a two-fold process;

1. The GMM is trained on a training set of feature vectors, and once the model parameters are estimated, the likelihood function of each Gaussian is used

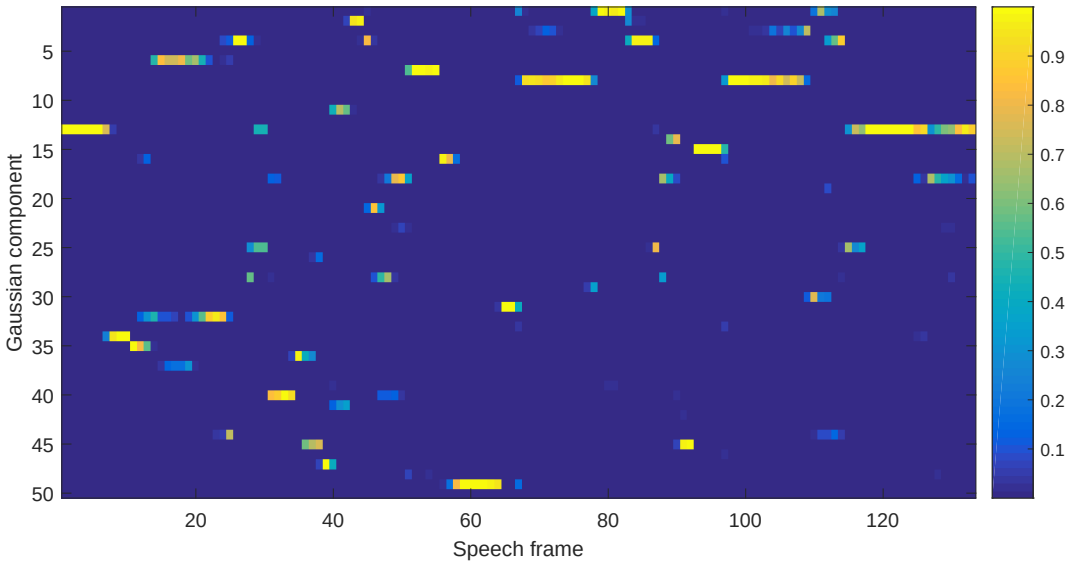


Figure 3.5: A Gaussian posterior gram shown for the uttered phrase "*Oticon alta*", where the posterior probabilities values are illustrated with different color codes. .

to generate the posterior probability vectors (according to the definition in Equation (3.7)). These probability vectors are concatenated to form the raw posteriorgram.

2. A discounting-based smoothening technique is applied on the raw Gaussian posteriorgrams to avoid approximation errors. Meaning that a small amount of the probability mass is shifted from high density to low density dimensions.

GMM training should be handled carefully since it is of critical importance with respect to creating a model that discriminates well between the underlying phonetic classes. Otherwise, the Gaussian posteriorgrams become unbalanced, in the sense that a few Gaussian components will dominate the probability space, while the remainder of Gaussians will represent only a small portion of the training data. This is usually the case when training examples are noise contaminated and other non-speech sources are present. In some of the worst-case scenarios observed in [45], a few dimensions can occupy up to 95 % of the probability mass, while only 5 % of the probability mass is distributed among the majority of the dimensions. One approach to avoid unbalanced posteriorgrams is to apply a VAD, and subsequently only train the GMM on the extracted speech segments.
After GMM training, the raw posteriorgrams can be generated in accordance to the definition in Equation (3.6). Next, smoothening is applied to the posteriorgrams to avoid any approximation errors. This implies setting a probability threshold floor P_{min}, below which all raw posterior values are zeroed out. The posterior probabil-

ity vector is then re-normalized to set the sum across the dimensions to one. As a final step in the discounting-based smoothening strategy, a small amount of the probability mass is moved from the non-zero to the zero dimensions to remove the effects of thresholding. For a posterior probability vector q_i, each zero dimension $z_{i,j}$ is rescaled to

$$z_{i,j} = \frac{\beta}{\text{count}\,\{z_i\}} \quad \text{for} \quad j = 1, 2, \ldots, \text{count}\,\{z_i\} \tag{3.8}$$

where $z_{i,j}$ is the j^{th} zero dimension in q_i, β is a smoothing factor, and $\text{count}\,\{z_i\}$ denotes the total number of zero dimensions in q_i. The non-zero dimensions $v_{i,j}$ are redefined according to

$$v_{i,j} = (1 - \beta) v_{i,j} \quad \text{for} \quad j = 1, 2, \ldots, \text{count}\,\{v_i\} \tag{3.9}$$

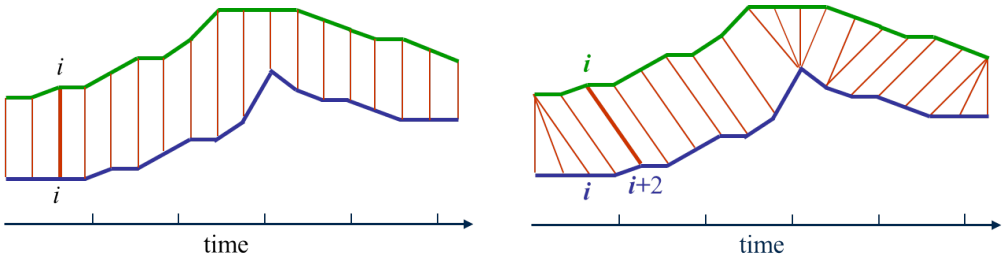
where where $v_{i,j}$ is the j^{th} non-zero dimension in q_i, and $\text{count}\,\{v_i\}$ denotes the total number of non-zero dimensions in q_i. These are the last steps in generating the final posteriorgram. Using the above procedure template posteriorgrams can be extracted from the training keywords (and stored in the template database), similarly test posteriorgrams can be extracted from the test inputs. Thus the next stage in the system process is to apply a matching routine, which searches for common patterns between a test input and a template instance, each represented by a Gaussian posteriorgram.

3.4 Pattern matching

In the field of ASR, dynamic time warping (DTW) was initially applied to isolated word recognition for aligning isolated test instances with examples of keyword terms. Later the technique was adapted to continuous word recognition, wherein a reference template is matched against a continuous test utterance by applying a sliding window approach on the speech segments. Although being initially developed for speech applications, DTW has found its applicability in other fields of study, e.g. in handwriting recognition, bioinformatics, finance etc [59]. DTW is a non-parametric method for time series alignment and comparison. It seeks to align two feature vector sequences by warping the time axis iteratively until an optimal match between them is found according to some suitable distance measure.

The main motivation of using DTW, is to account for the natural time duration variance in spoken words. The same word is almost never uttered exactly the same way twice, e.g. due to varying speaker rate, causing fluctuations in the time duration of the speech signal. To cope with time deformations, non-linear warping of the time axis is applied to ensure that the signal is matched against

its reference signal in some optimal alignment. The difference between regular linear alignment and non-linear type of alignment is outlined in Figure 3.6, where alignment is attempted on two time sequences (i.e. the green and blue curves) that are out of phase. However, since the optimal alignment will vary from case to case, and no simple calculations for finding it exist, the alignment task must be dynamic as indicated by the name of the method.



(a) A distance measure of any type which aligns the i^{th} point in one sequence to the i^{th} point in the reference signal is regarded as linear alignment, and will produce a poor comparison for temporally mismatched signals.

(b) Non-linear alignment is an elastic type of matching which allows points to be aligned across the time axis. It is often a more intuitive similarity metric, as it allows similar shapes to match even when they are out of phase.

Figure 3.6: Comparison of linear and non-linear alignment of two curves with time deformations [56].

3.4.1 Dynamic Time Warping

DTW has extensively been applied in solving temporal mismatch problems between two data sequences. In the context of KWS, the objective of the DTW algorithm is to optimally align the Gaussian posteriorgrams of the test input and keyword template, while evaluating the similarity between them. The optimal alignment path between the posteriorgrams is computed as the minimized residual distance between them, after having eliminated the time difference. Afterwards the accumulated cost along the optimal path is used as a basis for comparison. DTW (and its variants as introduced later) lie the foundation for the pattern matching methodology used throughout this thesis.

Consider two speech patterns expressed as a series of feature vectors (or a series of posterior feature vectors for that matter):

$$\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}, \tag{3.10}$$

$$\mathbf{Y} = \{\mathbf{y}_1, \ldots, \mathbf{y}_M\}. \tag{3.11}$$

To find the alignment path with respect to some minimal cost, a local distance (or cost) metric between two features, i.e. $c(\mathbf{x}_n, \mathbf{y}_m)$, needs to be defined. The

choice of cost metric highly depends on the properties of the feature space [48]. Whereas traditional features apply the Euclidean or Mahalanobi distances for similarity measure, the detection system in [45] uses a dot-product based measure, while yet others use a novel type of distance measure such as the Kullback-Leibler (KL) divergence in [48]. Typically, a low cost measure signifies high similarity between $\mathbf{x}_n$ and $\mathbf{y}_m$, while a high cost signifies a low similarity measure. Evaluating the cost measure of aligning each pair of vectors $\{\mathbf{x}_n,\ \mathbf{y}_m\}$, one attains the *cost matrix* $C \in \mathbb{R}^{N \times M}$ [57]:

$$C(n, m) = c(\mathbf{x}_n, \mathbf{y}_m), \quad \forall \{n \wedge m\} \tag{3.12}$$

In Figure 3.7(a) [57], the cost matrix of the two time sequences X and Y is depicted, where each point $(n,\ m)$ represents the cost value of aligning the n^{th} element in X to the m^{th} element in Y, thereby indicating regions of high and low similarity measures.
Given the cost matrix, the objective of DTW is to find the optimal alignment between $\mathbf{X}$ and $\mathbf{Y}$. To do so, we define a warping path as a sequence of ordered pairs [58]

$$\varphi = (n_k,\ m_k), \quad k = 1,\ 2,\ \ldots, T \tag{3.13}$$

where $T = \max\{N,\ M\}$ is the number of steps in the warping path. The warping path φ represents a mapping function for each point in one sequence to a point in the reference sequence:

$$\begin{aligned} \mathbf{x}_{n_1} &\leftrightarrow \mathbf{y}_{m_1} \\ \mathbf{x}_{n_2} &\leftrightarrow \mathbf{y}_{m_2} \\ &\vdots \\ \mathbf{x}_{n_T} &\leftrightarrow \mathbf{y}_{m_T} \end{aligned}$$

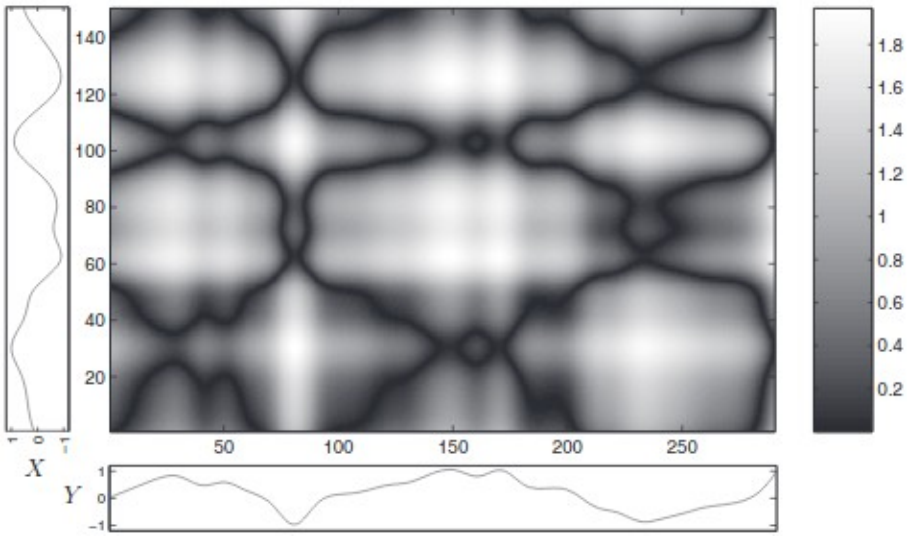
Hence two sequences can be compared based on the accumulated cost $D_\varphi(\mathbf{X},\ \mathbf{Y})$ along the alignment path of φ:

$$D_\varphi(\mathbf{X}, \mathbf{Y}) = \sum_{k=1}^{T} c(\mathbf{x}_{n_k},\ \mathbf{y}_{m_k}) \tag{3.14}$$

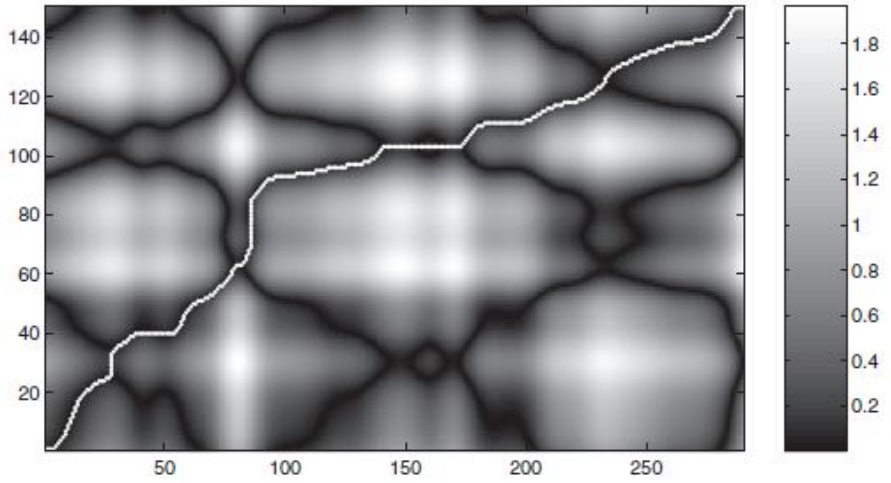
The optimal warp path $\hat{\varphi}$ is defined as the path minimizing the accumulated cost measure [18]:

$$\hat{\varphi} = \arg\min_{\varphi} \quad D_\varphi(\mathbf{X},\ \mathbf{Y}) \tag{3.15}$$

An ideal match between the segments in the feature vectors will be manifested as a sequence of similar regions along the lower-left to the upper-right diagonal within the cost matrix C. The same tendency can be observed in Figure 3.7(b), where the optimal alignment path of the cost matrix in Figure 3.7(a) is emphasized. The



(a) Cost matrix C of matching X and Y computed on the Manhattan distance, i.e. the absolute residual distance. Regions expressing high similarity are marked in black while low similarity regions are marked in white.



(b) Cost matrix introduced in Figure 3.7(a) shown along with the optimal warp path $\hat{\varphi}$ (white curve).

Figure 3.7: Cost matrix computed on the two time sequences X and Y using the Manhattan distance as a cost metric [57].

more the optimal path $\hat{\varphi}$ deviates from the the diagonal, the more time warping is required between the two sequences **X** and **Y** to get a suitable match.

Testing each and every combination of the components in $\mathbf{X}$ and $\mathbf{Y}$ to find the optimal path is a tedious and computational expensive process. Such a brute force procedure will result in a computational complexity that is exponential in the lengths of N and M. The optimal alignment can instead be found using dynamic programming (DP) techniques by applying a greedy algorithm, while imposing a number of constraints. Using DP-based algorithms reduces the computational complexity to $\mathcal{O}(MN)$ [57].

For a warping path φ described as a series of alignment points $\varphi = (\varphi_1, \ldots, \varphi_T)$ with $\varphi_k = (n_k, m_k)$, three types of conditions need to be met, i.e. [57]:

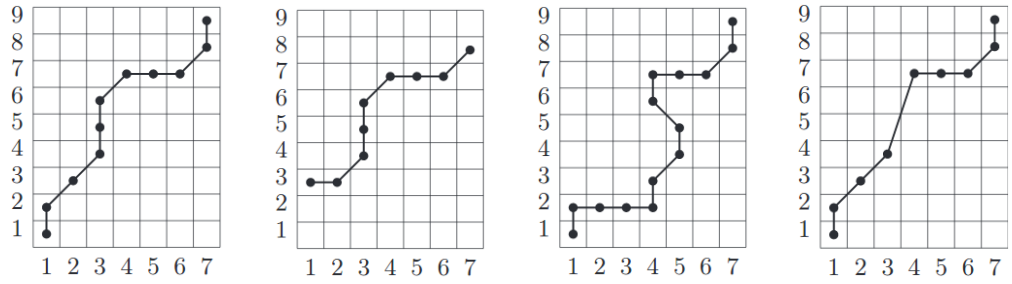
[1.] Boundary condition:
$\varphi_1 = (1, 1)$ and $\varphi_T = (N, M)$

[2.] Monotonicity condition:
$n_1 \leq n_2 \leq \ldots \leq n_T$ and $m_1 \leq m_2 \leq \ldots \leq m_T$

[3.] Step size condition:
$\varphi_{k+1} - \varphi_k \in \{(1, 0), (0, 1), (1, 1)\}$ for $k = 1, \ldots, T$

Condition [1.] enforces the alignment path to tie the start points and the end points of $\mathbf{X}$ and $\mathbf{Y}$. In other words, the path starts at the bottom left index within C and ends at the upper right index. The boundary condition ensures that both sequences are considered in their entirety. The second condition pertains to the principle of faithful timing, i.e. the alignment path does not turn back in time, either both indexes (n_k, m_k) increase or remains the same, but they never decrease. Finally, [3.] represents a continuity condition, ensuring none of the elements in either $\mathbf{X}$ or $\mathbf{Y}$ are omitted, nor do any replicates occur (i.e. all ordered pairs (n_k, m_k) are unique). Note that [2.] can be derived directly from [3.], but has been included anyway for the sake of clarity. All the conditions are depicted in Figure 3.8 given a simple toy-example, with Figure 3.8(a) illustrating a scenario where condition [1.], [2.] and [3.] are satisfied. While violations of the three conditions are illustrated in Figure 3.8(b)-3.8(d).

In addition to the constraints in [1.]-[3.], supplementary global constraints are sometimes imposed to limit the scope of the warping path. This helps to reduce the complexity of the search routine, by restricting the types of admissible moves. Prime examples of such constraints are e.g. the Sakoe-Chiba bound [60] that enforces the warping paths to lie within a fixed distance to the diagonal of the cost matrix, and the Itakura parallelogram [61] which constraints the path to lie within a parallelogram located around the diagonal. The Sakoe-Chiba bound and the Itakura parallelogram are illustrated in Figure 3.9(a) and 3.9(b), respectively.



(a) An example of an admissible alignment path φ which satisfies the conditions in [1.], [2.], and [3.].

(b) Violation of the alignment path w.r.t. condition [1.].

(c) Violation of the alignment path w.r.t. condition [2.].

(d) Violation of the alignment path w.r.t. condition [3.].

Figure 3.8: Illustrating admissible and invalid alignment paths through the cost matrix of the data sequences X of length $N = 9$ and Y of length $M = 7$ [57].

These restrictions of the warping path seek to lower the computational complexity by delimiting the number of optimal path candidates. The intuition behind the path bounds is that a decent warping path is unlikely to wander too far away from the diagonal. Yet other constrains restricts the shape of the path slope, to ensure the gradient is not too step nor too gentle, otherwise unrealistic warping of the time-axis might occur. Having a too step or gentle slope causes an undesired effect of mapping relatively long patterns to very short ones [63].

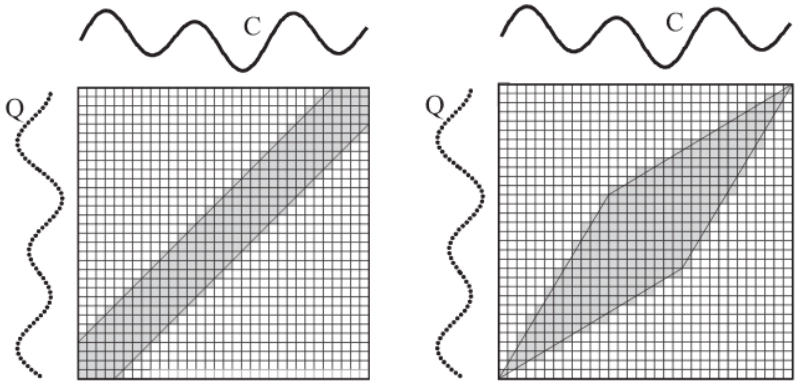
3.4.2 Dynamic Programming Algorithm

Revisiting the minimization problem in Equation (3.15), a DP algorithm will be defined which aims at finding the optimal path alignment in a recursive manner. DP is often deployed in applications where consecutive decisions are dependent, and where the resulting sequence must ensures a optimal cost. By doing so the number of possible alignment paths are reduced from $\mathcal{O}\left(N^M\right)$ to $\mathcal{O}\left(MN\right)$.

In a DP setting, the optimal path is found using a cumulative cost matrix instead of the traditional cost matrix defined in Equation (3.12). Given a set of partial pattern sequences $\mathbf{X}\left(n\right) = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ and $\mathbf{Y}\left(m\right) = \{\mathbf{y}_1, \ldots, \mathbf{y}_m\}$, the entry of the cumulative cost matrix $G(n,\ m)$ is expressed as the shortest path up to the index pair $(n,\ m)$:

$$G(n,\ m) = c(\mathbf{x}_n, \mathbf{y}_m) + \min\left\{G\left(n-1,\ m\right), G\left(n,\ m-1\right), G\left(n-1,\ m-1\right)\right\} \quad (3.16)$$

where the three terms inside the minimization reflects the step size condition on the warping path φ as defined in item [3.]. Instead of treating a single cell $(n,\ m)$



(a) The Sakoe-Chiba bound. (b) The Itakura parallelogram.

Figure 3.9: Two types of global constraints are illustrated on a cost matrix. The constraints are imposed to delimit the number of optimal warping path candidates [62].

as the alignment of the n^{th} element in X to the m^{th} element in Y as in $C(n,\ m)$, each cell $G(n,\ m)$ expresses the cost of the shortest path up to that particular cell defined recursively based on the preliminary cells. The cumulative cost matrix can be computed in a column-wise fashion by processing one column at the time starting from the first column. In that case, all elements in the m^{th} column are derived from the previous column. Likewise, the matrix can be formed in a row-wise fashion. However, the running time is the same in either case, i.e. $\mathcal{O}\,(MN)$.

The algorithm for computing the optimal path $\hat{\varphi}$ using a DP approach is outlined in Algorithm 2, and takes as input the cumulative cost matrix evaluated at all row and column indices, i.e.

$$\mathbf{G} = G(n,\ m) \quad \text{for} \quad \{n = 1,\ \ldots,\ N\ \wedge\ m = 1,\ \ldots,\ M\} \tag{3.17}$$

The optimal path is derived in reverse order of the start indices starting from the upper right corner cell $\hat{\varphi}_T = (N,\ M)$, and tracing the optimal path backwards until it reaches the start indices $\hat{\varphi}_1 = (1,\ 1)$, where the algorithm eventually terminates. At each iteration step l, the prior point of alignment $\hat{\varphi}_{l-1}$ is defined as the set of adjacent cell indices which provides minimal accumulated cost. Except in the specific cases where φ_l is located in the border regions, then $\hat{\varphi}_{l-1}$ can assume only a single value given by either decreasing the row index or the column index by one (depending on whether the current alignment point is situated at the horizontal or vertical border region). After termination of the back-tracing routine, the accumulated cost of the optimal path is scaled by the total number of steps to achieve the time normalized cost $\overline{G}$.

Algorithm 2: DTW algorithm for finding the optimal path $\hat{\varphi}$.

Input: Cumulative cost matrix **G**

Initialization: $\hat{\varphi}_T = (N,\ M)$

Back-tracing:

for $l = T,\ \ldots,\ 2$ **do**

$$\hat{\varphi}_{l-1} = \begin{cases} (1,\ m-1) & \text{if} \quad n = 1 \\ (n-1,\ 1) & \text{if} \quad m = 1 \\ \arg\min \begin{cases} G\,(n-1,\ m) \\ G\,(n,\ m-1) \\ 2 \cdot G\,(n-1,\ m-1) \end{cases} & \text{otherwise} \end{cases}$$

end

Termination:

$$\overline{G} = \frac{1}{N+M} G(N,\ M)$$

In relation to the plots in Figure 3.7, the optimal alignment path derived using the DTW algorithm is shown in Figure 3.10 [57] together with the cumulative cost matrix. As compared to the former results presented in Figure 3.7(b), the trace of the optimal path is more obvious from the cumulative matrix presented in Figure 3.10.

In the following a DTW variant denoted segmental DTW (SDTW) will be introduced, which deviates from the classical DTW algorithm in respect to the constraints imposed on the warping path. However, in the other aspects the two algorithms are much alike, e.g. do both incorporate DP into the algorithms.

3.4.3 Subsequence and Segmental Dynamic Time Warping

Matching sequences with a significant difference in the lengths is a common issue encountered in many applications, including KWS where a relatively long test posteriorgram representing a multitude of word is compared against a typically shorter template posteriorgram representing a single word. In this regard, it is inappropriate to enforce a global alignment path between the two sequences, as such a well-defined path does not exist. A particular branch within DTW named *subsequence DTW* [57] is tailored to handle these type of issues, which is the category of which the SDTW algorithm also belongs to. Instead of attempting a global alignment of the sequences, the goal of subsequence DTW is to identify a subsequence

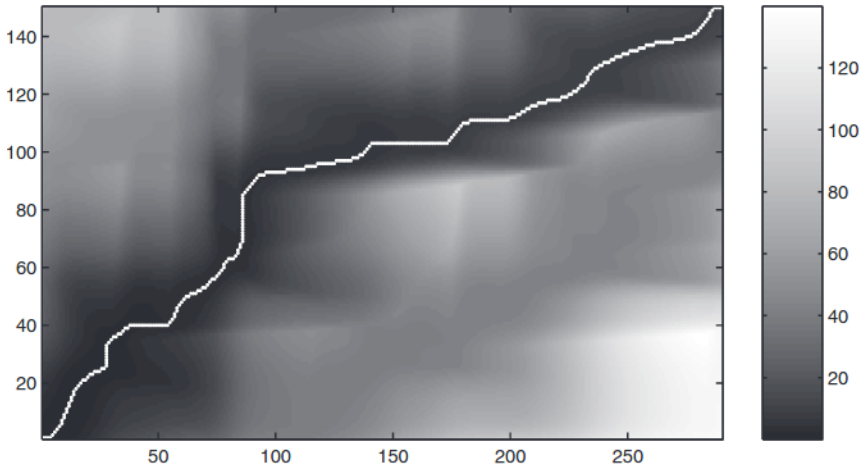


Figure 3.10: Optimal alignment path $\hat{\varphi}$ (white curve) depicted along the cumulative cost matrix derived from the data values in Figure 3.7 [57].

within the longer sequence which matches the shorter one in an optimal way. The principle behind subsequence DTW in a KWS context is illustrated in Figure 3.11, where a template signal X is aligned to a test sequence Y, to find the fragment within the test sequence that optimally fits the keyword. In practice, subsequence alignment can be conducted by sliding the short reference sequence across the longer target sequence in overlapping windows. Each observation window is then scored with a similarity value, corresponding to the extent of the match.

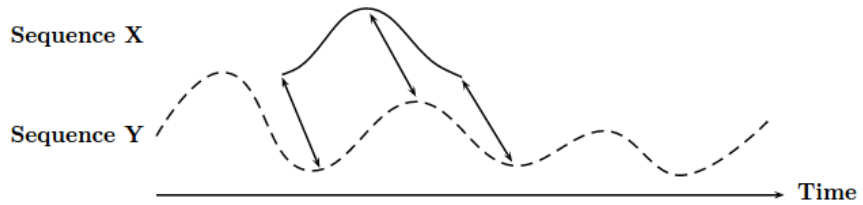


Figure 3.11: Optimal alignment procedure using subsequence DTW where a short template sequence X and is matched against a test sequence Y. The goal is to detect the fragment within the sequence Y providing the best fit to the keyword sequence X [57].

In [45], SDTW is explored as a framework for solving the subsequence alignment problem, its effectiveness in unsupervised word detection has formerly been demonstrated in [58]. The problem of comparing a test string to a keyword template, is translated into detecting common patterns between two posterior probability distributions in the form of Gaussian posteriorgrams. A cost measure is selected based on the criteria that two posterior probability vectors $\mathbf{x}_n$ and $\mathbf{y}_m$ originating from the same phonetic event must exhibit high similarity. A common cost metric $c\left(\mathbf{x}_n,\ \mathbf{y}_m\right)$ used in e.g. [45] and [47] is the dot-product evaluated on $\mathbf{x}_n$ and

$\mathbf{y}_m$ in the log probability space as:

$$c\left(\mathbf{x}_n,\ \mathbf{y}_m\right) = -\log\left(\mathbf{x}_n \cdot \mathbf{y}_m\right). \tag{3.18}$$

The resulting dot-product between the two probability vectors $\mathbf{x}_n$ and $\mathbf{y}_m$ will always lie in the range of 0 to 1, and is inclined towards the higher end when the vectors exhibit strong correlation. The log operation maps the similarity score into an expression of distortion. As shown from the function plot in Figure 3.12, a strong similarity score is translated to a low distortion score, and vice versa.

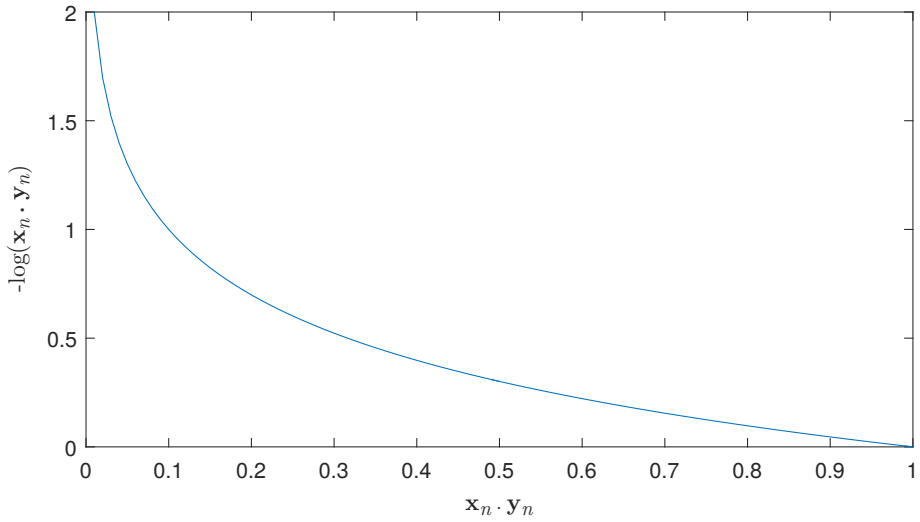


Figure 3.12: Mapping from similarity to distortion score via the log-function.

According to Equation (3.12), the new cost matrix can be extracted by evaluating all combinations of the instances in $\mathbf{X}$ and $\mathbf{Y}$ on the above inner-product operation. From the essence of SDTW, two further conditions are made in addition to constraints [1.]-[3.] as introduced in Section 3.4.1. They are as follows:

[4.] A global constraint is imposed to restrict the shape of the warping path, by applying an adjustment windows of some size R.

[5.] Search only specific and predefined warping paths, defined at different start and ending points.

The first additional constraint is imposed to prevent an excessive amount of temporal skew in the warping path. This is done by applying an adjustment window which restricts any coordinate pair in the warping path to deviate no more than a fixed distance R from each other. To state it formally, the k^{th} coordinate pair of the warping path, i.e. $\varphi_k = (n_k,\ m_k)$, must satisfy the inequality,

$$|n_k - m_k| \leq R. \tag{3.19}$$

Imposing such a constraint ensures the warping path is not too far ahead nor lacks too far behind with respect to either of the posteriorgrams $GP(\mathbf{X})$ and $GP(\mathbf{Y})$. As seen from Figure 3.13, the constraint is manifested as a diagonal with width $2R+1$ that surrounds the warping path.
Introducing constraint [5.] fixes not only the start coordinates of the warping paths but also the step size between them. The adjustment windows size is set to R, meaning the distance between two consecutive segmentation paths is R steps. Thereby, a number of fixed path bounds are generated based on the start coordinates. As a consequence of fixing the starting points, the adjustment window will restrict not only the shape but also the range of possible ending points. Thus combining both constraints result in a number of fixed path structures, i.e. the cost matrix is divided into multiple contiguous diagonal regions of width $2R+1$. For a given case where $R = 2$, Figure 3.13 outlines the first two of such paths.

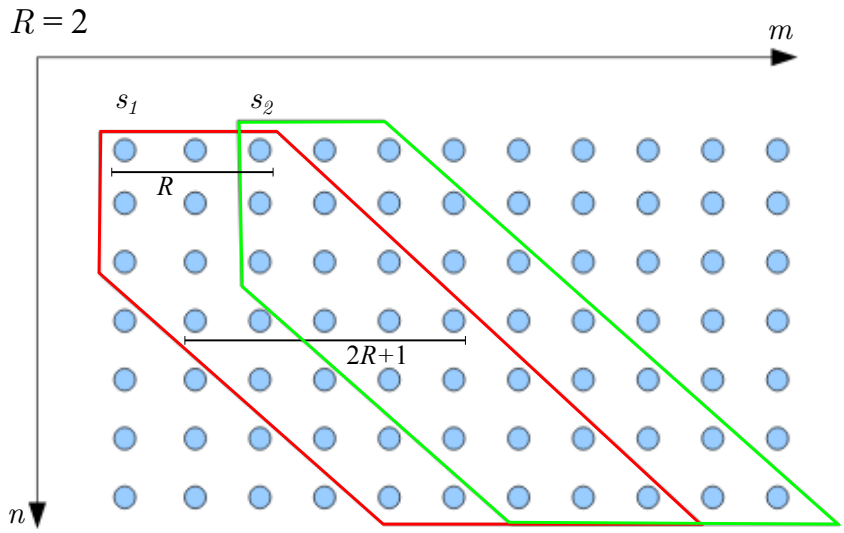


Figure 3.13: SDTW works by splitting the cost matrix (the blue dots illustrate indices in the matrix) into regions of overlapping pentagons with width $2R+1$. The points s_1 and s_2 are the start coordinates for the two first warping path bounds, with $R = 2$ [45].

The amount of overlap between the alignment regions is a balance between avoiding redundant computations, and giving thought to potential keyword occurrences across the segmentation boundaries. Specifically, the overlapping rate is regulated by R, as the adjustment window size determines the number of steps before a new DTW search is conducted. Since the width of the segmentation regions is selected to $2R+1$ a overlapping rate of roughly 50 % is achieved.

For the particular application of KWS, only the test utterance is segmented while

the template instance is fixed. Let $GP_{\mathcal{K}}$ and $GP_{\mathcal{T}}$ represent the posteriorgrams of the keyword template and test input, respectively. From the definition in Equation (3.12), the cost matrix then becomes

$$C(n,\ m) = c\left(GP_{\mathcal{K}}\left(\mathbf{x}_n\right),\ GP_{\mathcal{T}}\left(\mathbf{y}_m\right)\right) = -\log\left(\boldsymbol{q}_{\mathcal{K},\,n},\ \boldsymbol{q}_{\mathcal{T},\,m}\right) \tag{3.20}$$

where $\boldsymbol{q}_{\mathcal{K},\,n}$ is the n^{th} posterior probability vector belonging to the keyword posteriorgram, while $\boldsymbol{q}_{\mathcal{T},\,m}$ is the m^{th} posterior probability vector in the test posteriorgram. From the above definition it becomes clear that segmentation and thus windowing only is required along the m-dimension. For a given value of R and a length N of the test utterance, the start coordinates of the sliding window are [45]

$$\left(1,\ (k-1)\,R+1\right), \quad 1 \leq\ k\ \left\lfloor \frac{n-1}{R} \right\rfloor \tag{3.21}$$

Each time k is incremented by one, the window slides to the next position as specified by the starting point, and so forth until k reaches its upper limit. For a single keyword template, the sliding window procedure results in a total of $\left\lfloor \frac{n-1}{R} \right\rfloor$ warping paths, each representing an alignment between a template example and test subsequence found by applying the DP algorithm in Algorithm 2.

In summary, the SDTW algorithm can be regarded a DTW variant which implements an identical warping routine, but differs in the aspects of the imposed path constraints. Or interpreted in another way, SDTW is a DTW algorithm with an additional pre-stage that extracts multiple bounded segments (in the shape of pentagons) from the cost matrix and subsequently passes these to the warping routine. Afterwards, a distortion score (i.e. the accumulated cost along the optimal warping path) is computed for each alignment of the template and test subsequence. Once all warping paths have been scored, the resulting values are sorted in ascending order, and the one providing minimum distortion is then selected to be the candidate region within the test signal where the keyword is most likely to have occurred.

3.5 Score Ranking and Evaluation Metrics

In Section 3.4, SDTW was reviewed as a method to pattern matching between the test and template posteriorgrams. The matching routine returns a set of distortion scores, one for each region segment, indicating the fit of a template instance across a test utterance. All distortion scores are then collected and ranked accordingly, such that appointing a candidate region for the keyword occurrence is simply a matter of selecting the time region affiliated with the score giving minimum distortion (i.e. the first score on the ranked score list). In case multiple templates are provided for the same keyword, a merging strategy is needed to calculate the

final score considering the contribution form each keyword template, such an approach is e.g. discussed in [45]. Ideally, a system inspecting a test input with N keyword occurrences, will return a ranked score list where the scores affiliated with the keyword regions appear on the top N. However, such an ideal behavior is rarely seen in practice, often detection systems make wrong classification and therefore decisions. Hence suitable performance metrics are of great importance when evaluating detection performance, as well as when comparing different keyword detectors.

3.5.1 Metrics for Detection Performance

Prior to presenting the different evaluation metrics, it is essential to understand the type of decision errors a keyword spotter can make. Two types of detection errors may arise from the keyword spotter, one is triggering the keyword spotter when no keyword is present, i.e. false-positive (FP), and the second error option is to overlook a keyword occurrence, i.e. false-negative (FN). Conversely, correct classification results either from identifying a target keyword, i.e. true-positive (TP), or simply rejecting a non-keyword, i.e. true-negative (TN). These four possible classification outcomes are represented in a confusion matrix as shown in Table 3.1.

		Actual state	
		keyword	no keyword
Hypothesis	keyword	TP	FP
	no keyword	FN	TN

Table 3.1: A confusion matrix representing all possible classification outcomes in KWS.

The metrics displayed in from Table 3.1, are rarely sufficient when viewed in isolation for evaluating the keyword spotter performance. This is due to the unbalanced nature of the KWS problem; the target keywords occur infrequently in a continuous listening system as compared to non-keywords, hence the system is considerably more prone to false alarms than overlooking keyword terms [9]. A useless system that does not detect any keywords will hence obtain a low error-rate, because it will not raise any false alarms. Thus, a more befitting and informative figure-of-merit (FOM) is required to ensure a fair system evaluation. Three such metrics are described in the following.

Receiver Operating Characteristic curve

One such classic FOM in the context of KWS is the receiver operating characteristic (ROC) curve, which measures the true positive rate (TPR) also known as the recall, as a function of the false positive rate (FPR). Each of which are defined as [10]:

$$\text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{3.22}$$

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}} \tag{3.23}$$

The TPR measures the fraction of positive examples (i.e. keyword occurrences) that are correctly classified, whereas the FPR measures the fraction of negative examples (i.e. non-keyword occurrences) that are misclassified as keywords [10]. Hence, the ROC relates both to the ability of the system to spot the occurrence of keywords and reject non-keywords occurrences.

Each point on the ROC curve corresponds to one decision outcome at one specific threshold value b (in accordance with the description in the *Decision stage* from Definition 1.1), for which the metrics in Equation (3.22) - (3.23) are calculated. Therefore, generating the ROC curve necessitates a threshold sweep across all possible output scores [9]. In Figure 3.14 two example curves are shown, one which exhibits a excellent performance (solid curve), and secondly a dashed curve that corresponds to random guessing, and therefore represents a worthless detector.
Each point on the ROC curve represents a trade-off between the two types of errors; i.e. either reducing the number of false alarms while more keyword occurrences are overlooked, or vice versa. This trade-off is sometimes referred to as the precision-recall trade-off. Often the preferred trade-off is not known on beforehand, in this case the average performance over all the points is used as an alternative evaluation metric. In particular, preference will be given to systems that attains the highest area under the ROC curve (AUC) [9].

Equal Error Rate

The equal error rate (EER), is a commonly accepted evaluation metric of overall system performance. It corresponds to the threshold at which the false positive rate is equal to the false negative rate (FNR), indicating the proportion of false rejections is equal to the proportion of false acceptance at this very point. Since the FNR is defined as

$$\text{FNR} = 1 - \text{TPR} = \frac{\text{FN}}{\text{TP} + \text{FN}}, \tag{3.24}$$

it can be obtained directly from the intersection point between the ROC curve and the off-diagonal of the unit-square. A corresponding visual explanation is given in Figure 3.15. The lower the EER value is, the better the system performance is.

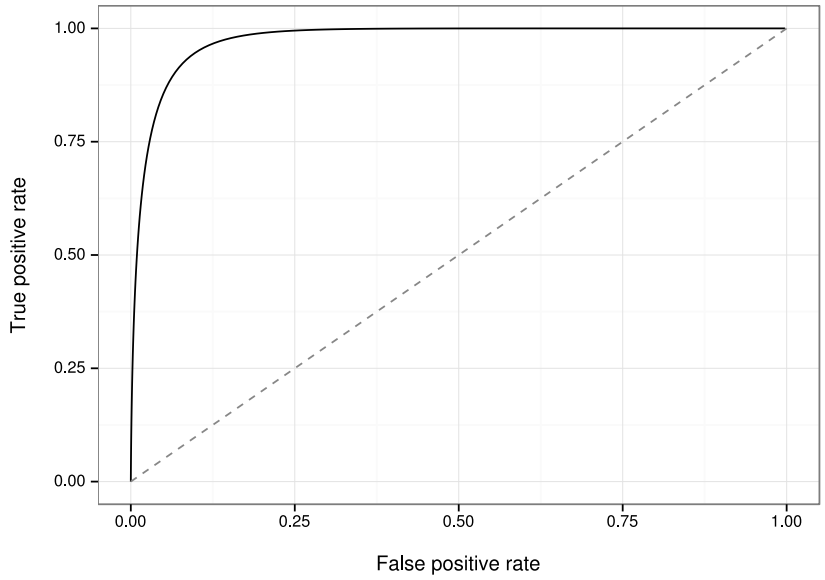


Figure 3.14: ROC curve which exhibits examples with excellent (solid) and worthless (dashed) performance. The dashed curve is also used as a performance reference.

Interpreted visually, a small EER implies that the ROC curve is shifted towards the upper left corner, and thus attains a high AUC.

The equal error rate is also referred to in the literature as the crossover error rate (CER) [67].

Precision@N and mean average precision

From a list of ranked distortion scores, the precision@N (P@N) metric measures the average detection precision of a keyword k_i among the top N_i hits, where N_i is the total number of occurrences of the i^{th} keyword in the test string [68]. The P@N value is calculated as the number of distortion scores affiliated with the keyword k_i among the top N_i hits, normalized by N_i,

$$\text{P@N}_i = \frac{\#\{k_i \text{ occurrences in top } N_i \text{ hits}\}}{N_i} \tag{3.25}$$

A simple approach of combining the average precision score of multiple keywords $\mathcal{K} = [k_1, k_2, \ldots, k_N]$, is using the mean average precision (MAP) score. This metric considers the ranking of each keyword, as opposed to the P@N measure which ignores a keyword target k_i ranked lower than N_i. The MAP score computes the average precision for each keyword and finds the mean across the dictionary of

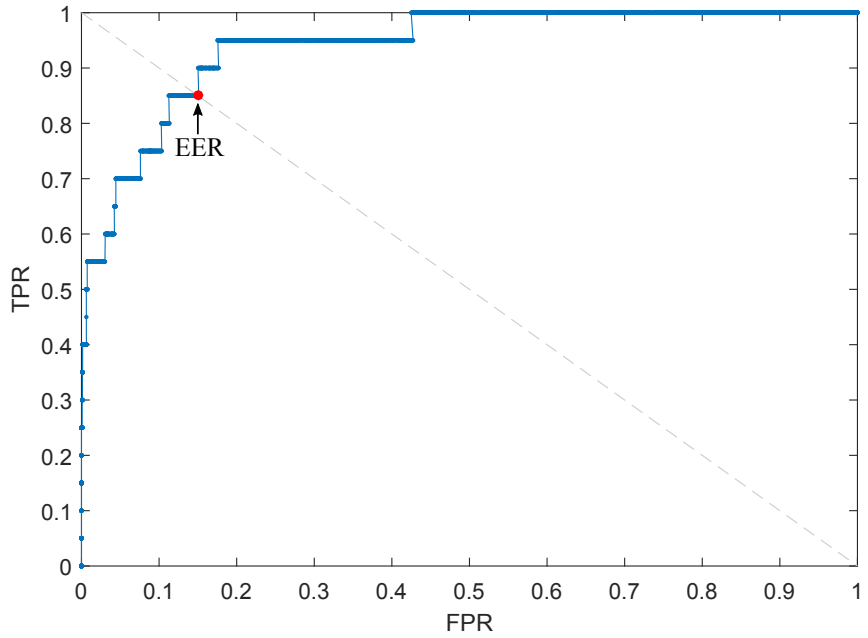


Figure 3.15: Visually, the EER can be interpreted as the intersection between the ROC curve and the off-diagonal line.

keywords [68]:

$$\text{MAP} = \frac{\sum_{i=1}^{N} \left(\frac{1}{N_i} \sum_{r=1}^{N_i} \frac{r}{p_{i,\,r}} \right)}{N} \tag{3.26}$$

where N is the size of the keyword vocabulary, N_i is the total number of repetitions of the i^{th} keyword term contained in the test input, $p_{i,\,r}$ represents the position of the r^{th} keyword target in the ranked score list for the i^{th} keyword.

Following a complete system description, and a short review of relevant metrics for evaluating detection performance, the next step is to analyze and simulate the system behavior. The simulations are carried out under different training and testing conditions, to mimic real case scenarios of the keyword spotter in use, and verify the system work as intended under a given set of conditions. Eventually the spotting performance is compared to contemporary KWS systems using the above-mentioned metrics as a frame of reference.

Chapter 4

Experiments and Results

This chapter aims to evaluate the unsupervised KWS framework presented in Chapter 3. The system is implemented and tested in MATLAB, the complete code can be found in the appendix CD and an overview of the CD content is given in Appendix B. The code for the implementation is mainly written by the author, but is in its entirety a combination of the author's code with public available scripts and toolboxes. In particular, the *voicebox* toolbox [69] has been applied for feature extraction, while a classical DTW implementation found in [70] has been modified to a STDW setting and used for pattern matching.
The experimental data used for both training and testing are acquired through speech recordings from a number of test persons. All recordings are made in a controlled acoustical environment, to generate as noise free speech signals as possible. This implies that recordings are conducted in an acoustically isolated room to avoid interference from external noise sources, and the distance between microphone and speaker is relatively short. Recordings have been carried out with both a conventional studio microphone and with a microphone embedded in a hearing aid. Details on the experimental setup are provided in Appendix A. Hence the initial system evaluation aims to assess the spotting performance in a controlled acoustical environment. Generally, this chapter seeks to examine the KWS system under different test scenarios and conditions, which reflects real case applications. Initially, a short description of the training an test data is provided in Section 4.1. Next, the experimental setup used in the application of single word detection is described in Section 4.2, along the the resulting detection performance. A more comprehensive keyword detection experiment is presented in Section 4.2.3, which explores detection performance in a multiple speaker, multiple keyword setting. In Section 4.3, a similar experiment is carried out for testing (false) keyword rejection capabilities of the keyword spotter. Later, in Chapter 5, an investigation into the system performance is carried out under non-ideal acoustical conditions.

4.1 Training and Test Data

Before going into details regarding the system evaluation, a short description of the test and training data is given. An essential part of evaluating every classifier is to separate the available data into a training and data set. As the name indicates, the training data is a set of speech recordings allocated solely for training purposes, and here in particular for the GMM to learn the partitioning of a feature vector into a set of components. After the system model (i.e. GMM) has been learned based on the training data, the keyword spotter is evaluated against the test data. The set of test data helps to investigate how well the system generalizes when being exposed to unseen data. Therefore it is of critical importance that any data applied during the training phase is excluded from the test set, and vice versa. A fundamental principle in machine learning applications pertaining to the distribution the training and test data, assumes the training data to be representative of the test data. This implies that the phonetic distribution of the training examples is somewhat identical to that of the unseen test examples. Violations of this basic assumption typically result in a poor classification accuracy, because the test data will behave differently from the training data, and therefore the performance of the learned system on the test data will degrade [71].
Since the test set contains values for the attributes we want to predict, i.e. the true classification of the test instances is known on beforehand, and therefore it can easily be determined whether the system model has correctly classified a test instance or not. In a KWS context, the test input is a speech signal labeled with the exact location of the keyword occurrences, and therefore it is apparent to verify from the detection output whether keyword regions are classified properly according to the signal labels.

In general speech processing applications, a database of prerecorded audio files and corresponding text transcriptions are often used for system evaluation. These databases referred to as speech corpora, consist of hours of speech data recorded from a number of test subjects covering both male and female speakers. However, testing and system evaluation as presented in the subsequent sections will rely on own recordings. Using such an approach gives the freedom to define and regulate the acoustical settings for the recordings, this pertains e.g. to the noise conditions in the room, the distance to the microphone, the type of microphone, the location of the microphone (e.g. behind the ear as in the case of a HA) etc. As a first step, the acoustical settings are adjusted to facilitate the operating conditions of the keyword spotter, and thereby evaluate its detection performance under acoustical favorable conditions. Furthermore, there is a limitation in the existing speech corpora in relation to the variety of keywords as well as the number of repetitions for each keyword. Conducting own recordings, allows for customizing the training

and test data from a set of self-determined speech recordings.

Details on the speech recordings are provided in Appendix A, where the recording procedure along with the experimental setup is described. A total of nine different keywords shown in Table 4.1 are recorded from five male and five female speakers, with every keyword repeated 30 times. Additionally, all test subjects were asked to read the same newspaper article to collect some general speech data. The duration of each keyword recording, i.e. 30 repetitions of the particular keyword is approximately 1 minute, while the article takes around 5 minutes to complete, resulting in roughly 15 minutes of speech data from each test person. All are recorded with with two types of microphones; initially a studio microphone was used for the recordings, and afterwards the recordings were repeated with a microphone embedded in a commercial hearing aid (HA). The selection of the studio microphone pertains to the aforementioned objective of creating an ideal (acoustic) setting for initial evaluation purposes. Due to its cost, size, features, complexity, etc, the studio microphone is expected to produce recordings of higher quality (as compared to microphones embedded in some mobile device), e.g. in the sense of achieving higher SNR values of the recordings. Whereas, the HA microphone closer resembles the microphone expected to be found in a mobile device, and thus better represents the achievable performance in a real-case scenario. In the result sections, the studio microphone and the HA microphone will be referred to as micHQ (high quality microphone) and micHA (hearing aid microphone), respectively.

#	Keyword
1	Oticon Alta
2	Oticon Epoch
3	Oticon Agile
4	Program 1
5	Program 2
6	Program 3
7	Volume up
8	Volume up
9	Go to sleep

Table 4.1: Recorded keywords for experimental use.

The collected data is apportioned into a training and test set, of which the keyword repetitions are reserved in a ratio 2:3 for training while the remaining third is used for testing, i.e. the majority of the keywords are allocated for the GMM to learn the respective keywords. As for the article recordings, they are mainly used for GMM

training, but also to create customized test sentences at which keyword instances are inserted at certain points, preferably in silence regions.
From a practical perspective, personal keyword recordings from the system user need to be acquired somehow. One such approach is to introduce a preliminary training session, in which the system user is prompted to utter the keyword(s) in isolation a number of times while the system is recording. Subsequently, the captured keyword examples are stored and divided into a training and a test group according to the aforementioned distribution.

The experiments mainly serve the objective of evaluating the keyword spotter in two types of scenarios:

1. *Keyword detection*: This is the most typical scenario, where a speaker n (referred to as the native system-user) intends to trigger his or her personal mobile device by uttering a wake-up word. In that case, the system model is trained on the speech recordings of the n^{th} speaker, likewise the keyword test inputs originate from the native system-user. The specificity of the keyword spotter and in particular the GMM may vary, as a of result of using more or less keywords. For our experimental framework, the maximum number of different keywords is nine (see Table 4.1), while the minimum is one.

2. *Keyword rejection*: Conversely, if a test utterance is spoken by a non-native user (i.e. any of the other $n-1$ test speakers), the keyword detector is expected to ignore that keyword utterance, as triggering the system is reserved to none but the native user. This scenario is easily tested by composing keyword recordings from multiple non-native system speakers into a test input. The system model is still trained on recordings from the n^{th} speaker, i.e. the native system-user, but the spoken keywords originates from the remaining $n-1$ speakers. Thus, the keyword rejection scenario can help to clarify how speaker dependent the keyword spotter is, according to its ability of averting non-native speakers' attempt to trigger the system. In wider terms, this scenario investigates the ability of the detector to refuse all other words, phrases, noises, and sound inputs besides the keyword(s) uttered by the native system-user.

4.2 Keyword Detection

In the following, the keyword spotter will be evaluated according to its ability of detecting a single keyword. This resembles one of the simplest scenarios in the discipline of continuous keyword spotting, where a prerecorded keyword template represented in the feature domain is matched against a test utterance in an

online classification routine. Nonetheless, this particular setting is highly relevant in many practical applications, one such prime example is to trigger a device running in a sleep-mode (or low power mode). Upon successful detection of the wake-up word, i.e. the keyword, the device can safely resign from the sleep-mode. Therefore, it is desired to device a system which is highly specialized and precise in detecting a single keyword. Such a system can be accommodated through different training strategies of the GMM.
In the approach used in [45], the TIMIT corpus [72] is used to train a GMM with 50 components based on a set of 3,696 speech sentences recorded from 462 speakers of both sexes. Using such a training strategy, the GMM becomes a general speech model that represents a broad phonetic space. In the following experiments a similar approach is embraced; the GMM is trained on general speech data in the form of article recordings. However, since the quantity of general speech data is rather restricted in our case, it must be ensured that the GMM learns the partitioning of the keywords otherwise, such that the keywords are represented in the phonetic space modeled by the GMM. To that end, an adaption phase is introduced where the GMM learns a keyword through training examples of that particular keyword. To reflect a realistic scenario of the training phase, the system model for a test speaker n is not trained on the article recording conducted by the speaker himself (or herself), as longer general speech recordings of the system user is unlikely to be available in any practical case. Rather, the system model will be solely trained on the article recordings of the remaining $n-1$ speakers. As for the keyword adaption, 20 repetitions of the respective keyword recoded by system user will be used. This is indeed a realistic approach, since the keyword examples from the system user (here denoted speaker n) are acquired through the above-mentioned training phase.

For testing, a sentence is constructed for each speaker, containing a total of 10 keyword repetitions throughout the whole test string. For this purpose, a VAD is utilized for detecting silence regions in the test string at which the keywords are inserted. Using the VAD, automates the work of constructing test inputs, and furthermore provides a more natural way of combining an already existing speech recording with keyword insertions such that they do not overlap in any sense. The keyword chosen for this initial experiment is the first one in Table 4.1. Having ten different speakers, give rise to ten corresponding test strings that each is evaluated individually.

4.2.1 Procedure

The experimental procedure is initiated by applying a VAD on speech segments of 25 ms duration and with 10 ms overlap. Each extracted signal frame is then

represented by 13 MFCCs including the zero coefficient, with first and second order derivatives (delta and delta-delta coefficients), resulting in a 39-dimensional feature vector per frame. All feature vectors generated from the training set are used to train a 50 component GMM, with exception of the keyword template. Upon completion of the training procedure, the GMM is used to decode test and template frames to generate Gaussian posteriorgrams. Finally pattern matching between the posteriorgrams is carried out via SDTW, using the parameter setting found in [45] to exhibit the best performance, i.e. the smoothening factor β is fixed to 0.0001 while the window size R is set to 6. The minimum distortion scores found from each alignment window are then collected and sorted in ascending order for computing the performance metrics.

4.2.2 Experimental Results

In this initial evaluation case, it assumed that only one keyword template is available for detection. Under this assumption, ten keyword repetitions from each of the ten speakers are included in the experiment, yielding a total of 100 trial searches. The optimal alignment scores from the search routine are plotted in Figure 4.2 for the micHQ recordings, while Figure 4.3 shows the corresponding results for micHA. The results from each test speaker are shown in separate warping path vs. distortion score plots, these are depicted with green regions indicating the time interval for the true keyword occurrences, expressed in terms of warping path coordinates. Obviously, the red regions mark the span of warping coordinates at which no keyword is present.

Figure 4.2 clearly shows that good matches between the keyword template and test keywords are consistently found for all keyword repetitions. Within each plot, the distortion scores affiliated with the background words lies roughly in the same score range and thus appear as a large cluster, meaning the majority of the non-keyword segments are classified in similar manner. Likewise, keyword scores are clearly separated from the non-keyword scores by a large margin, and appear as abrupt deflections at their respective time of occurrence. Having a clear and distinct partitioning of true and false classes is a desired property in many classification tasks, and particular in KWS where it facilitates the task of selecting an appropriate decision threshold. Thus a keyword spotter which possesses significant discrimination capabilities produces a broad margin between the classified entities and thereby attains higher performance. The virtue of having a large margin is e.g. measured by the ROC curve which sweeps a threshold across the distortion scores and computes the FPR and TPR at each instance. By virtue of having a broad margin, false positives and false negatives are less likely to occur. The idea behind is illustrated in Figure 4.1, where a set of keyword (green) and non-keyword (red) distortion scores are depicted along the possible classification

outcomes. From the figure it becomes clear that a classifier which discriminates well, yields a broader FP and FN margin, which in turn improves the FPR and TPR of the ROC curve.
The ROC curves generated from the micHQ distortion scores are shown in Figure 4.4. These plots signifies optimal detection performance as the AUC is 1 in all cases, and therefore the resulting EER values are consistently zero for micHQ. The EER values across all the speakers are reported in Table 4.2 for both micHQ and micHA.

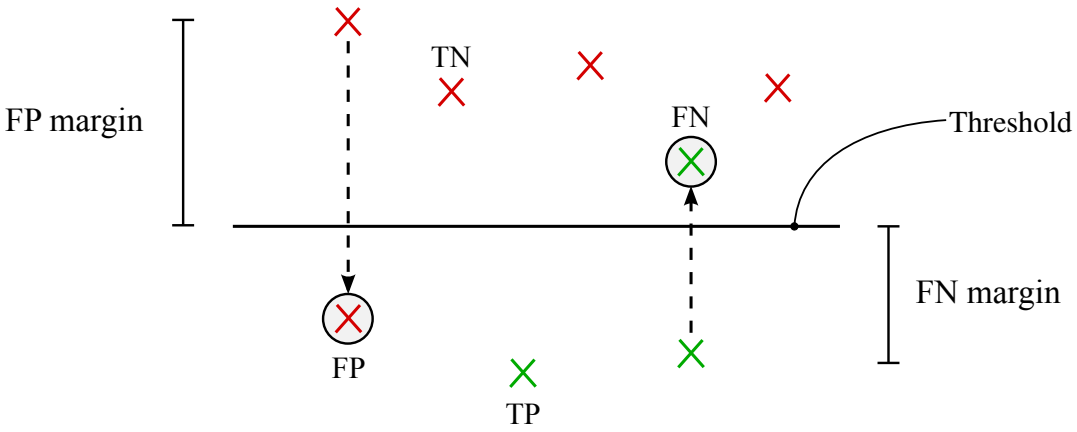


Figure 4.1: Keyword (green) and non-keyword (red) distortion scores are shown in the four possible categories they can be classified into. The figure expresses the virtue of having a broad margin, namely to avert false positives and false negatives. Eventually, such differences in the margins become evident from the ROC curve.

The results shown in Figure 4.3 follows more or less the same trend as that of the plots in Figure 4.2, i.e. apparent score deflections occur in keyword segments, while the distortion scores in the non-keyword regions are mostly grouped together at the higher end of the distortion score scale. However, there are specific result sets, like the plots for test person 2 and 8, where keyword occurrences are not detected perfectly. This drop in performance can be observed as a dip in the respective ROC curves shown in Figure 4.5, as the number of false negatives increases at the majority of examined threshold levels during the sweep. This in turn causes the TPR to drop until the threshold value reaches above the undetected keyword scores. Consequently, the EER metric will increase due to the notch in the ROC curve. This is because the intersection of the off-diagonal line with the ROC curve shifts in the right direction and thereby attains higher EER value.

Another important aspect of the observed distortion scores is the affiliation of those warping indices which attain minimal distortion (i.e. whether they are generated from a keyword or background word), as they will be ranked highest on the final score list. Given N repetitions of a keyword k in a test string, the precision of the

system will be 1 equivalent to 100 % if all N keyword instances appear among the top N hits, i.e. $\text{P@N}_k = 1$, and declines gradually as more keyword scores falls outside the top N. The precision metric is reported for each speaker and each microphone in Table 4.2, together with the average precision (across the speakers) for each microphone. A quick inspection of the plots in Figure 4.2, easily reveals that the vast majority of the distortion scores lying in the lowest end of the scale appear in the green regions, and are therefore affiliated with keyword occurrences. When ranking the scores, it becomes clear that the precision is optimal or near optimal in all cases, more specifically the average precision is 0.98 for micHQ and 0.89 for micHA. The results generated from the micHA data generally attains a bit lower performance, primarily due to the scoring results of speaker 2 and 8. In the worst case, i.e. for speaker 8, only half the keyword trials are ranked among the top 10 hits, giving a precision equal to 0.5. But, overall the micHA recordings generate decent results that are comparable to micHQ.

	Speaker 1	**Speaker 2**	**Speaker 3**	**Speaker 4**	**Speaker 5**
P@N - micHQ	1	0.9	1	1	1
P@N - micHA	1	0.8	1	1	0.8
EER - micHQ	0	0	0	0	0
EER - micHA	0	0.05	0	0	0.06

	Speaker 6	**Speaker 7**	**Speaker 8**	**Speaker 9**	**Speaker 10**	**Average**
P@N - micHQ	1	1	1	1	0.9	0.98
P@N - micHA	1	1	0.5	0.9	0.8	0.89
EER - micHQ	0	0	0	0	0	0
EER - micHA	0	0	0.1	0.005	0	0.02

Table 4.2: Performance metrics for single word spotting for micHQ and micHA.

The results and the corresponding evaluation metrics presented in this section suggest the keyword spotter works as intended; it is capable of extracting meaningful patterns and subsequently detect any occurring keyword instances, however, under certain conditions. The first condition concerns the aim of creating an ideal

acoustical setting for the recordings which are described in Appendix A, such as to avoid any type of noise artifacts in the recordings. To that end, a close match between the keyword template and test examples of the keyword is to be expected, because they are recorded immediately in succession and in the same acoustic environment. Conversely, if the recording conditions between template and test instances differ, a degradation in the detection performance is expected. Secondly, the system is highly specialized in modeling only a single keyword, e.g. a wake-up word, which eventually facilitates the task of detection. Additionally, it should be noted that evaluation is based on a restricted quantity of speech data with a limited amount of test keyword occurrences. The effects from this restriction are visible on the ROC curves (see Figure 4.4 and 4.5), which have a rough curvature. A more comprehensive experiment is carried out in Section 4.2.3, where multiple keywords are included in the test string, thereby the number of test instances are increased by an amount proportional to the number of included keywords. Nonetheless, the results including the generated ROC curves from this initial experiment are promising, and demonstrate a KWS algorithm that hold great potential to work in practice, even in a constrained mobile environment such as a HA platform. This initial performance indication is challenged further in Section 4.2.3, where a more thorough test is conducted, which in addition explores how the presented keyword spotter performs compared to contemporary KWS systems.

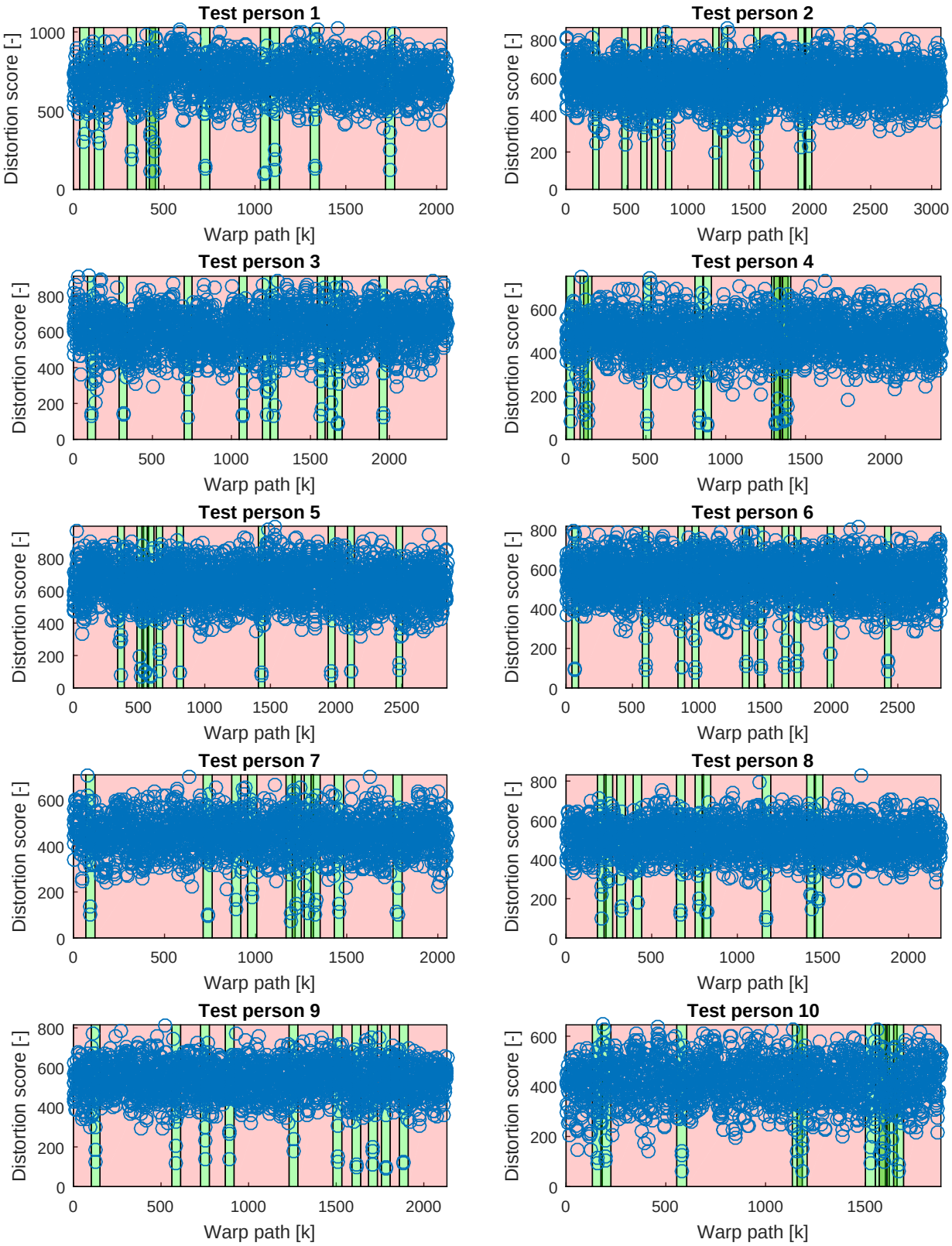


Figure 4.2: micHQ: Distortion scores resulting from single keyword detection for all test subjects.

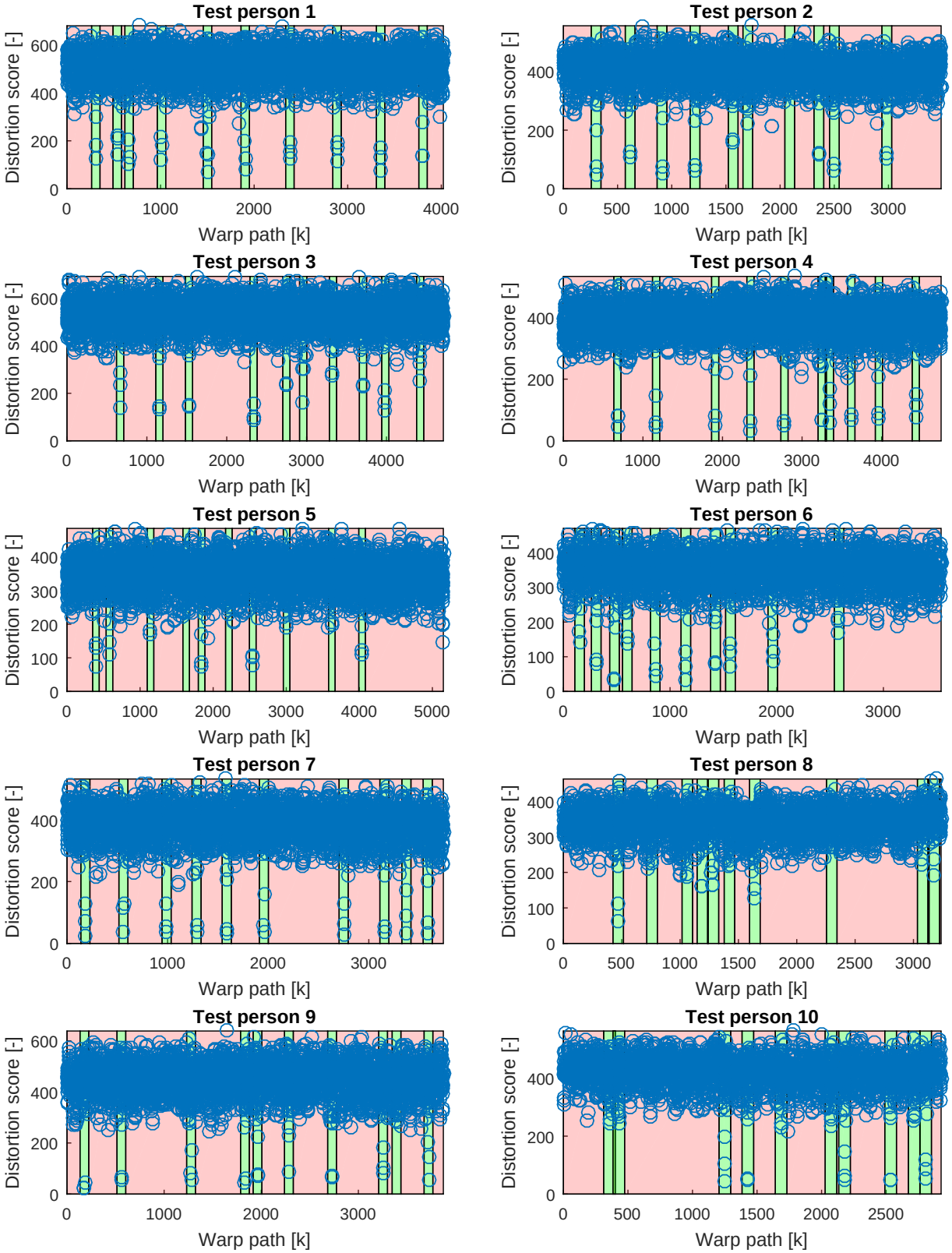


Figure 4.3: micHA: Distortion scores resulting from single keyword detection for all test subjects.

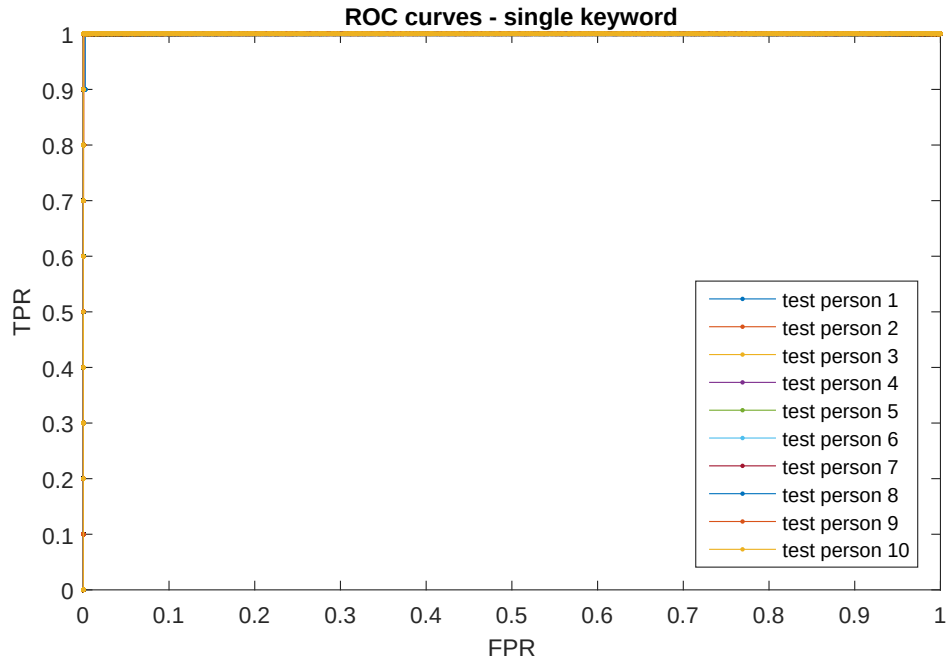


Figure 4.4: micHQ: Generated ROC curves from the initial keyword detection experiment for all test subjectst.

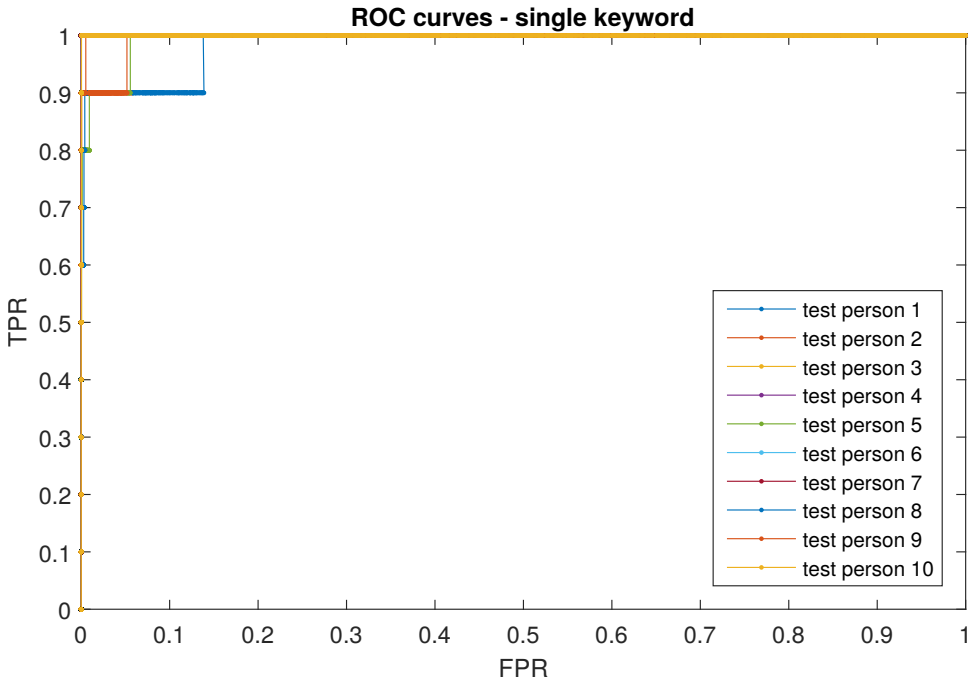


Figure 4.5: micHA: Generated ROC curves from the initial keyword detection experiment for all test subjects.

4.2.3 Multiple Keyword Detection

An extended experimental evaluation of the keyword spotter will be carried out in the following. The main objective is to perform a comprehensive system evaluation, in line with the test setups used in the evaluation of contemporary KWS systems [45][47][87]. This implies the use of multiple keywords each with multiple keyword repetitions. In the following a similar test setup will be adopted, such that the performance of the KWS system can be compared against contemporary state-of-the-art KWS systems. The first keyword spotter included in the comparison is described [47], and highly resembles the KWS architecture outlined in Figure 3.1, apart from the GMM which is replaced by a dedicated phonetic recognizer. Since the approach in [47] combines phonetic posteriorgrams with SDTW, it will be abbreviated PP+SDWT. The second KWS framework applied in the comparison is described in [68], and uses a posterior feature variant denoted acoustic segment model (ASM) posteriorgrams. The ASM posteriorgrams are derived from a set of HMMs (which are the acoustic segmental models) that are obtained in a unsupervised manner. In [68] a combination of ASM posteriorgrams and SDTW is applied, which subsequently will be denoted ASM+SDTW, additionally a second KWS architecture is proposed which forms a fusion of GMM and ASM posteriorgrams with SDTW, this will be referred to as ASM+GMM+SDTW. Finally, the KWS architecture from Figure 3.1 is abbreviated GP+SDTW, as it combines Gaussian posteriorgrams with SDTW.

The keywords used for the extended experiment are keyword 4-9 in Table 4.1. A test sentence is constructed for each speaker in which ten keyword repetitions of each keyword occur. Given ten different speakers and six keywords with each ten keyword repetitions, yields a total of 600 trial searches in the experiment. The parameter setting described in Section 4.2.1 has been reused for this experiment. All results are reported in terms of the P@N, EER, and MAP metrics. Regarding the P@N and EER metrics, one value is reported for each speaker for each keyword. In order to obtain a single expression of performance for each metric, the P@N and EER values are averaged over each word, and next over the different speakers. The MAP metric is slightly different in this regard, because it has an inherent approach of averaging over the different keyword, therefore it is only necessary to average the reported MAP values over the different speakers. The resulting performance metrics from the extended experiment are shown in Table 4.3

The reported metrics in Table 4.3 shows a significant degradation in the detection performance, relative to the initial experiment results presented in Table 4.2. From inspecting the resulting distortion scores, it appears that the performance degradation mainly is due to the obvious similarities between the selected keywords, e.g. “program 1”, “program 2”, and “program 3”. The resulting scores from the

	P@10	P@N	MAP	EER
GP+SDTW - micHQ	59.3	59.3	57.2	6.31
GP+SDTW - micHA	52.8	52.8	50.3	6.98
PP+SDTW	63.3	52.8	-	16.8
ASM+SDTW	56.6	40.6	40.4	-
ASM+GMM+SDTW	59.2	42.1	43.1	-

Table 4.3: Extended experiment results, with a comparison to contemporary state-of-the-art KWS systems.

matching routine when applying a keyword template corresponding to any of the three keywords, say "program 1", will exhibit low distortion for almost all test occurrence of the three keywords. For this example, it means that the "program 1" template will provide a good match in the test sentence to "program 2", and "program 3" occurrences. Hence the top-ranked distortion scores are often a confusion of the three keywords. The same intuition is behind the confusion of the keywords "Volume up" and "Volume down". Hence, the precision is expected to increase if the keywords are replaced with other words which have a more distinct phonetic content.

Considering the relative performance between the different KWS systems in Table 4.3, shows that the GP+SDTW is competitive to the other systems regarding the P@10 metric. Since there are always ten keyword test occurrences of each keyword in the considered GP+SDTW test setup, P@N and P@10 will always be equal in this case. This is not the case for the remaining KWS systems, as a varying number of keyword occurrences is used for each keyword. Hence, the most fair assessment is given based on the P@10 metric (as opposed to P@N), when comparing the GP+SDTW to the other methods. Additionally, the GP+SDTW system outperforms the other KWS systems, regarding both the EER and MAP metrics (for those KWS systems where these metrics have been reported). However, the results need to be considered with caution, as the training and test data typically differ in relation to the applied speech corpus, and e.g. in the number of keywords and keyword repetitions.

4.3 Keyword Rejection

Next, the keyword spotter is tested by its ability to reject keyword utterances spoken by non-native system users. A training setup identical to the one in Section 4.2 is used, while the test setup for obvious reasons differs from the previous one. Instead of inserting true keyword instances (i.e. keyword examples recorded by the native system-user) into the test string, these are substituted by keyword recordings from all the non-native system users. So, a practical scenario is simulated where group of speakers utter the keyword used to trigger the personal device some speaker. This experiment is repeated in ten different settings, such that each of the ten test speakers play the role of the native system user in one setting, while representing a non-native system user in the remaining nine settings. A single test string (representing one setting) contains in total ten keyword examples uttered by the non-native system users, one of these test subjects is selected at random to have two keyword repetitions included in the test string while the remaining test subjects have only one. In addition to interchanging the roles of the native versus non-native system user between the test speakers, the experiments will also be conducted using both types of microphones, i.e. micHQ and micHA. In this respect, there lies a notable difference in the configuration of the microphone recordings for this particular experiment. In the micHQ recordings, the keywords are recoded by speaking directly into a microphone placed immediately in front of the speaker. Whereas micHA recordings resembles a more realistic approach, where the speaker (playing the role of non-native system user) is displaced 1.2 meter, equivalent to a speaking distance, from the microphone while attempting to trigger the detector. In this recording setup, the HA microphone is mounted on the ear of a head and torso simulator (HATS) to make the recordings more authentic from a practical point of view.
The simulation procedure and the parameter setting follow the description in Section 4.2.1.

4.3.1 Experimental Results

The resulting distortion scores from the keyword rejection experiments are shown in Figure 4.6 for micHQ and Figure 4.7 for micHA. These plots represents the results from the ten test settings described above, which are distinct in their distribution of the native and non-native user roles among the test speakers. Furthermore, the green regions are now replaced by orange colored regions to emphasize that these regions contain false keyword occurrences, and that instances of FPs can potentially appear across these warping path indices. Comparing the keyword detection results from Figure 4.2 and 4.3 to the results in Figure 4.6 and 4.7, it becomes evident that the false keyword occurrences in the two later figures do not exhibit as strong deflections as true keyword occurrence do in the two former figures. In

fact, the majority of the false keyword insertions show either little or no evidence of detection, meaning the keyword detector to some degree, can discriminate keyword occurrences originating from the native system user from those that do not.

In certain cases the results strongly suggest that FPs will inevitably occur, this pertains especially to test person 2 and 7 in Figure 4.6, where several close matches are found between the keyword template and instances of false keyword occurrences. Less significant, though still clear matches are derived from the micHA results as indicated in Figure 4.7, for e.g. test person 2, 6, and 9. In terms of the rejection capabilities of false keyword terms, there is no striking difference between the resulting scores for the two types of microphones. However, the micHA results seem to provide a minor improvement in rejection performance, since the number of apparent FP instances are fewer and less severe in their distortion scores (i.e. they generally attain higher distortion).

Since no true keywords appear during testing, calculating the TPR is pointless and so becomes the computation of ROC curves. In fact, the absence of true keyword occurrences disqualifies the use of any of the described performance metrics. Therefore we resort to a more simplistic approach, where the FPR is plotted as a function of the threshold levels, as shown in Figure 4.8 and 4.9. These plots signifies that selecting a global threshold value is not straightforward, as the curve characteristics differ from speaker to speaker. On the other hand, there appears to be a threshold value at 200, where the FPR is close to zero across all speakers. A comparison of the plots in Figure 4.8-4.9 to Figure 4.2-4.3, seems to verify that 200 is a suitable threshold value for this particular experiment. At this exact threshold value, a desired balance is attained at which the threshold is sufficiently low to avoid false positives and therefore reject keyword occurrences uttered by a non-native system user (i.e. false keyword occurrences), while the threshold is sufficiently high to maintain a good detection rate of the true keyword occurrences.

In summary, the presented KWS system do exhibit a decent degree of speaker dependency for the majority of the test speakers, which is a highly favorable property in the context of KWS for a personal mobile device. This is mainly verified through inspection of the distortion score results in Figure 4.6-4.7, as there exist no explicit performance metrics to measure keyword rejection capabilities. From the plots, there is also evidence that FPs most likely will occur at several time instances. To that end, it is important to keep in mind that recordings are carried out in an acoustically controlled environment, which creates favorable conditions for keyword detection, but has the opposite effect in the case of keyword rejection. In fact, such a recording setup facilitates the conditions for a non-native system user to trigger the mobile device, in comparison to a real-life scenario where environmental noise is present, etc.

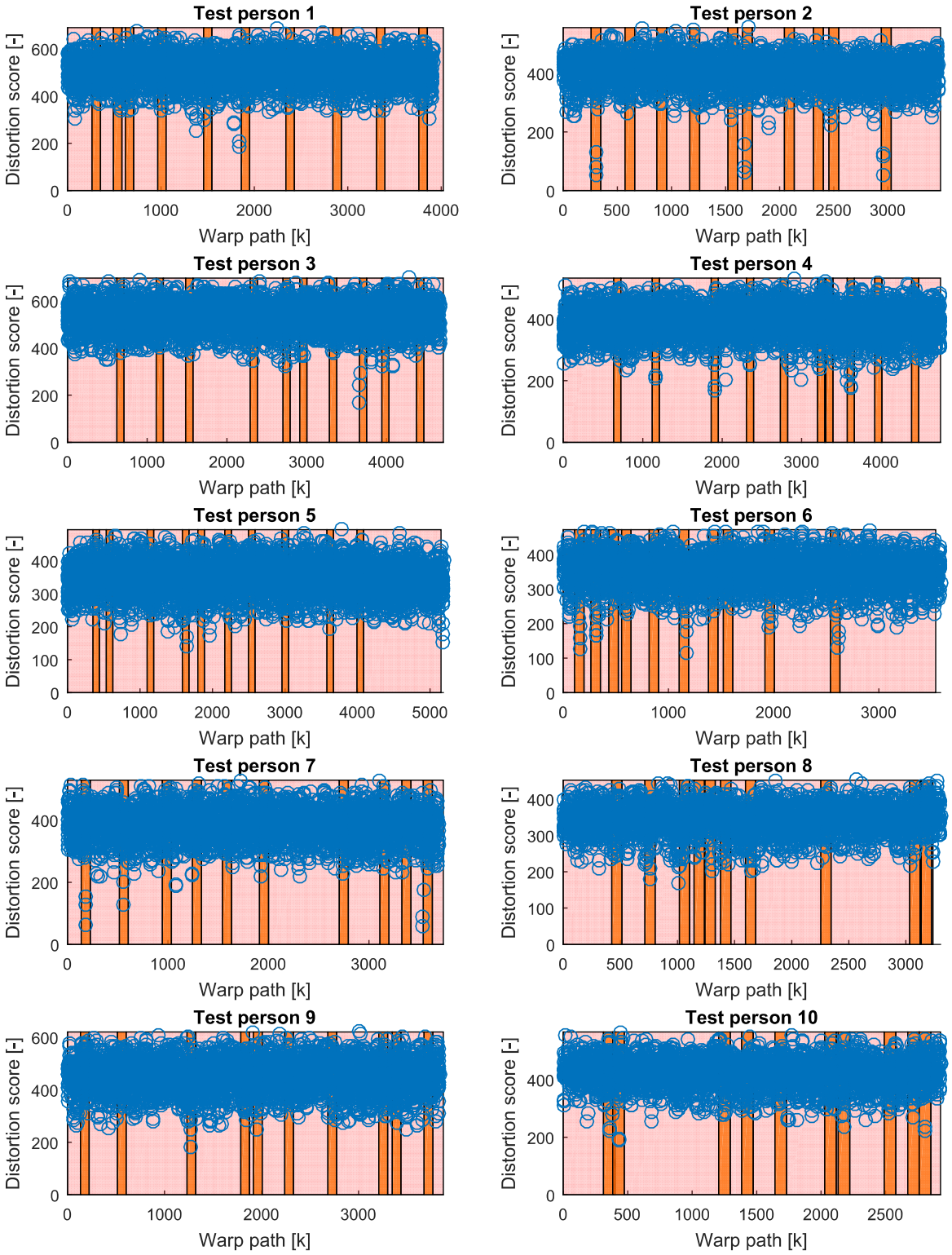


Figure 4.6: micHQ: Distortion scores resulting from single keyword rejection for all test subjects.

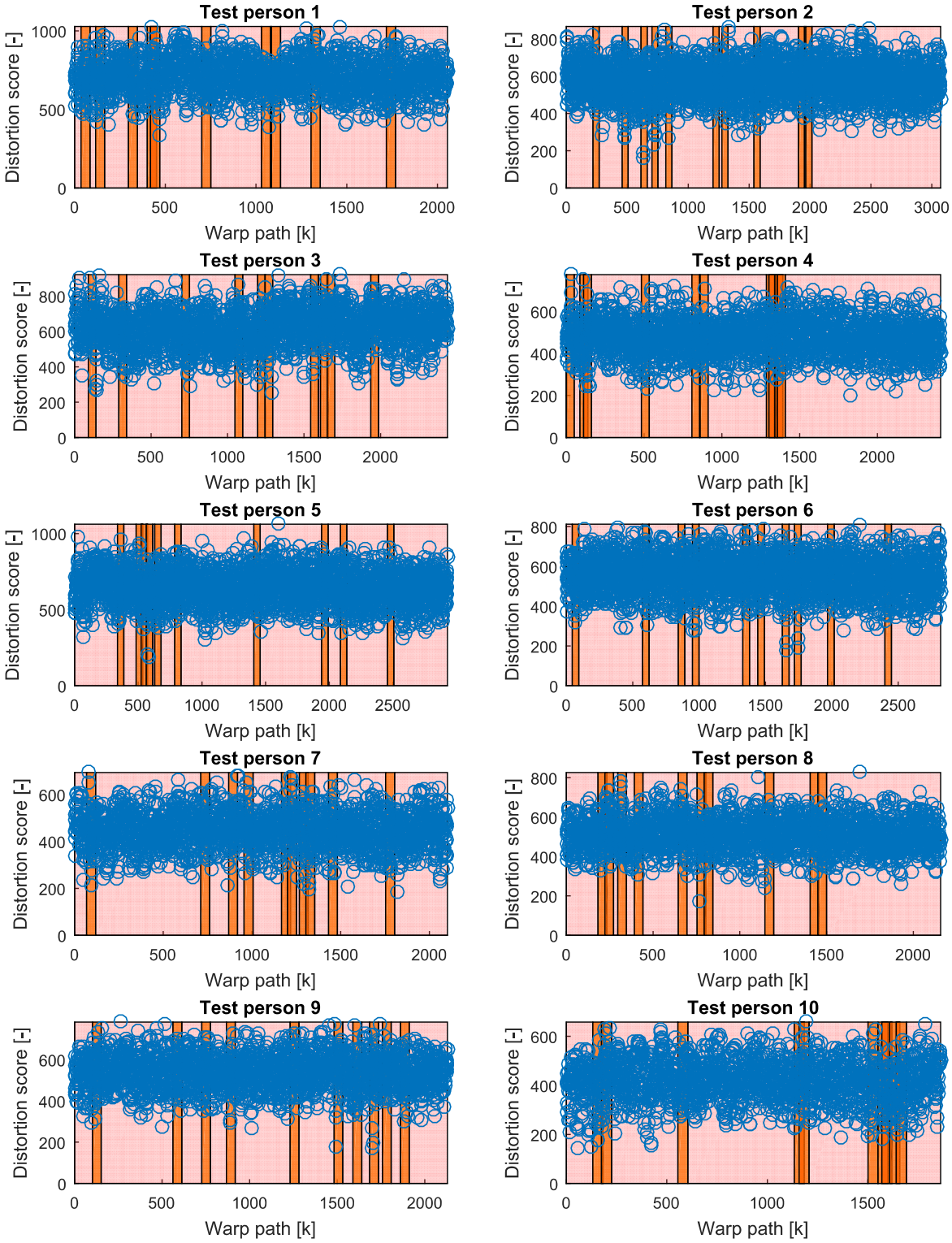


Figure 4.7: micHA: Distortion scores resulting from single keyword rejection for all test subjects.

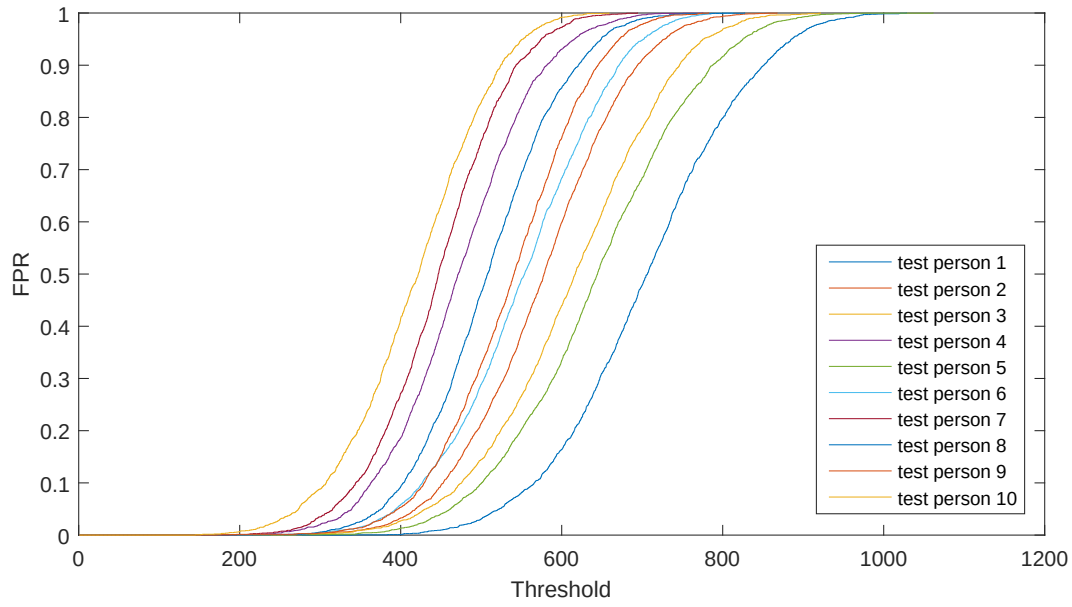


Figure 4.8: micHQ: FPR versus threshold plot.

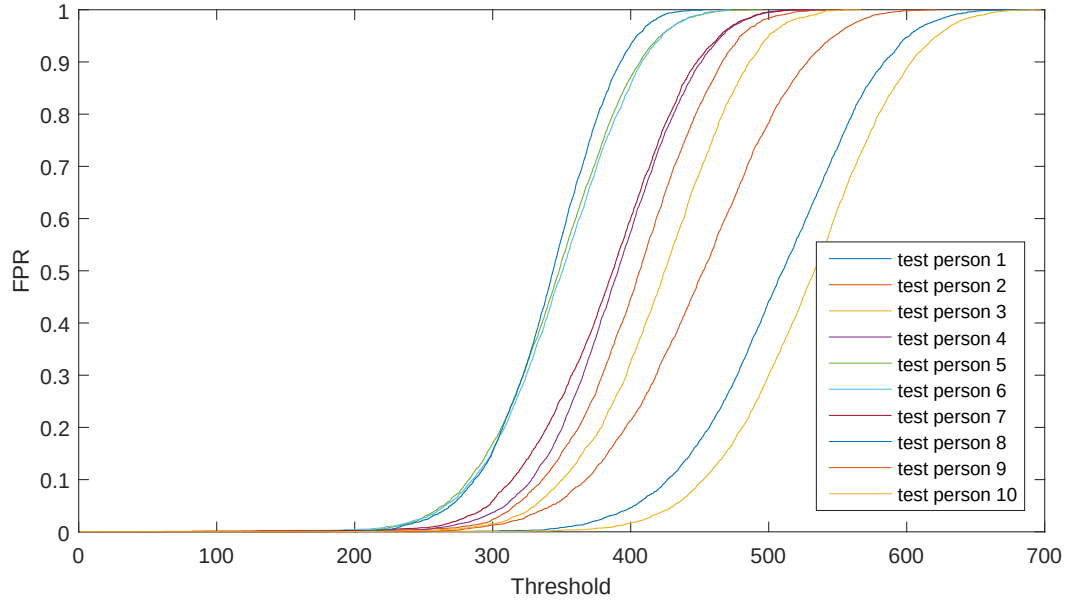


Figure 4.9: micHA: FPR versus threshold plot.

Chapter 5

Noise Compensation and Evaluation

In the previous chapter, recognition performance of the KWS system has been evaluated under an acoustically controlled and particularly noise-free environment. In close resemblance to real world operating conditions, it is desired to investigate the resulting effect of introducing various noise and disturbances into the tests. In particular, different sources of noise will be considered, i.e. both real life noise measurements obtained from the DEMAND (Diverse Environments Multichannel Acoustic Noise Database) noise corpus [83], and secondly babble noise and speech shaped noise (SSN).
As a first step, is important to clarify how detection performance is affected in the presence of various noise sources at different signal-to-noise ratios. In this regard, Section 5.2 evaluates the performance of the KWS system when the test data is contaminated with noise at various pre-defined noise sources and SNRs. Subsequently, different noise compensation strategies will be proposed to the application of keyword spotting, and more specifically in relation to the KWS architecture described in Section 3.1. The class of compensation methods considered in the following are known as *back-end* (or model domain) techniques, as they incorporate noise robustness into the system architecture. Finally, the spotting performance of the KWS system with noise compensation implemented, will be compared against state-of-the-art speech enhancement algorithms, as well as to the baseline system (i.e. without noise compensation). The classical speech enhancement are also referred to as *front-end* techniques, as they attempt noise compensation outside the folds (and independently) of the specific system architecture.

5.1 Training and Test Data Mismatch

It is well known that detection performance of conventional ASR systems drops dramatically, when recognition is carried out in noisy environments, due to the

resulting mismatch between training and test conditions [73]. Especially template-based KWS systems are prone to the mismatch problem, because the noise-contaminated test keywords will differ from the template keywords to a greater or lesser extent, depending on the noise type and the severity of the noise. This statement will be substantiated by experimental results in Section 5.2. Various noise compensation methods exist, with the goal of minimizing the effects of such mismatches, to obtain a detection performance close to that of the outset performance under matched conditions. Noise compensation techniques cope with the mismatch effects by compensation in either the test data, training data or in both. In the following, mainly two types of solution categories to the mismatch problem will be considered;

1. The application of classical speech enhancement methods as a pre-processing step, to remove or attenuate noise contaminants from noisy observations of the test data, thereby improving the SNR.

2. Incorporating noise compensation into the modeling stage, typically through machine learning techniques. This often implies adapting the back-end model to a specific noisy environment. In this context, the term *back-end model* refers to the (acoustic) model found within the system architecture, which is typically an HMM, GMM, etc.

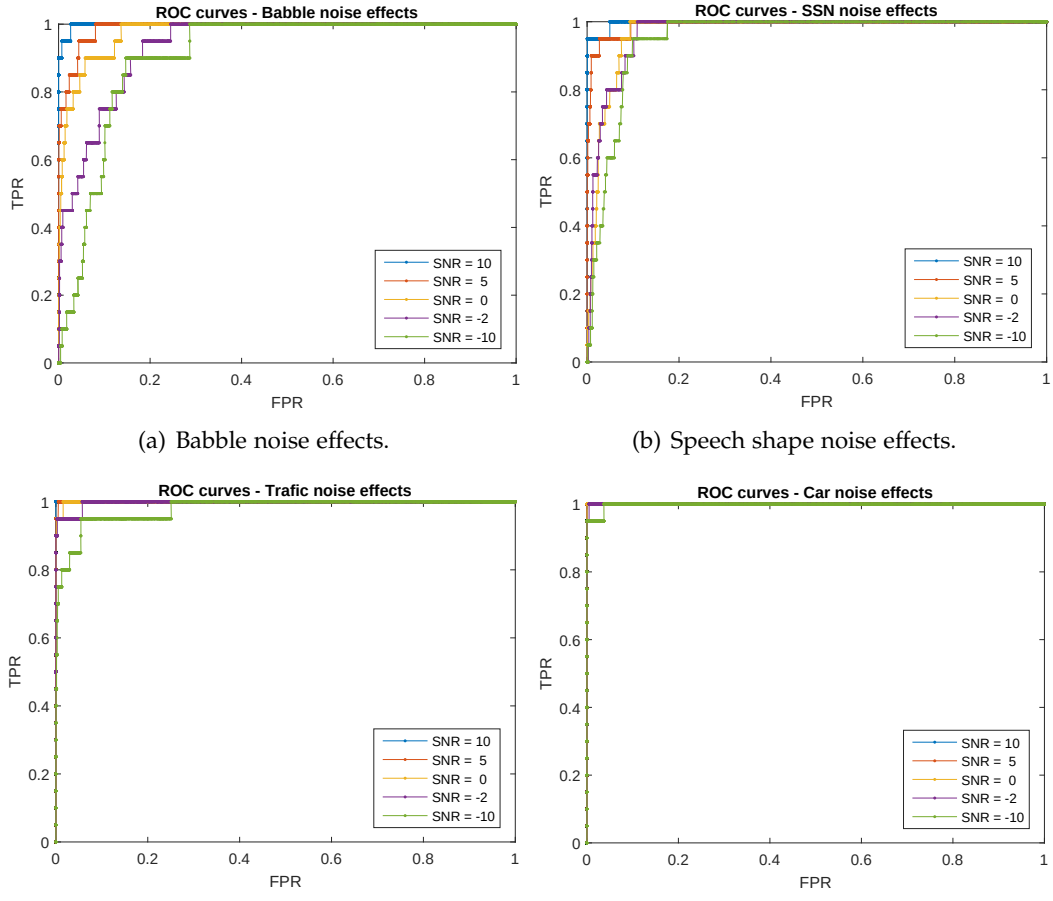
Solutions belonging to the first category are considered front-end techniques, i.e. they attempt to increase the intelligibility and quality of the noisy test signals prior to feeding them into the keyword spotter, and hence does not require any advance knowledge about the particular keyword spotter. This category includes a number of classical speech enhancement algorithms such as spectral subtraction [31] and Wiener filter [31], but also non-linear estimators such as the spectral amplitude MMSE estimator [31]. Contrary to the classical speech enhancement algorithms, the solutions belonging to the second solution category require insight into the particular KWS system, since noise compensation is attempted by incorporating robustness into the back-end model, e.g. through various training strategies. For this reason, model based methods can be considered a sort of back-end techniques, because they embrace the test input in its noisy form and instead perform compensation within the core of the recognizer. As opposed to the methods in the first solution category which attempt compensation directly on the noisy test signal, regardless of the applied keyword spotter. In the following sections both categories shall be considered; various noise compensation using model adaption will be explored and evaluated, while selected state-of-the art algorithms from the first solution category will be used as a reference for comparison.

5.2 Noise Sources and Performance Degradation

When evaluating machine learning based KWS algorithms, it of great importance to select suitable noise sources used for constructing training and test data. The chosen noise source must be selected such that they fit into the application which is desired to investigate, and not least be realistic. A commonly used approach is to apply a noise corpus, and for the subsequent experiments the DEMAND database will be used, which contain real-world noise recordings from a variety of environments. In particular, car and traffic noise will be used for evaluation, since they both represent the type of environmental noise encountered many typical real-life applications of a mobile device. A typical use-case where such noise sources are likely to be encountered, is in a hands-free dialog system in a car, which allows for controlling the vehicle functions (e.g. adjust temperature, radio, etc.) via voice. Furthermore, they are both stationary noise types, and will be considered in relation to non-stationary noise types. Thus, besides using noise recordings from the noise database, babble noise and SSN noise will additionally be considered for evaluating purposes. These have each distinct stochastic properties; the SSN noise is stationary by nature and is generated by filtering a sequence of white Gaussian noise with a all-pole filter in the form of Equation (2.2). The generated SSN sequence has a PSD similar to that of a long-term PSD of speech. On the other hand, babble noise is non-stationary, and is generated by mixing the speech recordings of different speakers, thereby simulating the scenario of having multiple simultaneously competing speakers. Keyword detection in the presence of babble noise is considered a difficult task, since the noise source highly resembles the target speaker, and additionally is non-stationary meaning the PSD of the noise can potentially change from frame to frame.

A general concern in machine learning applications when constructing training and test data, is to ensure that training and test data are unique in their speech signal as well as their noise signal, while being sufficiently large to carry out a proper system evaluation. On the contrary, using the same noise material for training and test purposes can generate misleading results, as the classifier might learn the noise sequence during the training phase, and thereby result in a artificially high classification accuracy. By the same token, the generalization capabilities are not evaluated properly when using such an approach. Furthermore, training and testing of the KWS system will be carried out without the use of looped noise, but only unique and unseen noise realizations. Looped noise here, refers to the process of replicating the same noise sequence a number of times to fit the length of the training and test set, i.e. until all clean training and test instances are contaminated with the particular type of noise. Hence all noise contaminated training and test in the following are constructed from non-looped noise realizations.

The effective impact of the different noise sources on the detection performance is shown in Figure 5.1, for a set of pre-defined SNR levels. The experimental setup described in Section 4.2 has been reused, but the test inputs have been scaled to the appropriate SNR level for the given noise source. Further an ideal VAD is employed for the experiments, i.e. the VAD is applied on the clean speech signals, to remove any effects that may be caused due to misclassification of speech and noise/silence segments. Additionally, the number of keyword occurrences in the test string has been doubled to improve the resolution of the ROC curves, i.e. the available keywords are allocated in a ratio 1:3 for training, while 2:3 of the keywords are reserved for testing.



(a) Babble noise effects.

(b) Speech shape noise effects.

(c) Traffic noise effects.

(d) Car noise effects.

Figure 5.1: Noise effects on detection performance of the KWS system, for babble noise, SSN, trafic and car noise each evaluated at the SNR levels $(10,\ 5,\ 0,\ -2,\ -10)$.

From the resulting ROC curves in Figure 5.1, it is clearly observed that babble noise degrades detection performance the most, whereas e.g. car noise is much less severe and only has a small impact in some extreme SNR scenarios. Generally, the effect of the measured noise sources is less visible as compared to babble noise and SSN, especially at lower SNR values where they have little or no impact. The observed results from Figure 5.1 are expected due to the difference (of the noise sources) in respect to the spectral and spectro-temporal similarities to speech [84]; regarding SSN, its spectral characteristics highly resembles that of regular speech, and therefore comprises a greater challenge to the keyword detector, than e.g. car and traffic noise. As for babble noise, it is highly non-stationary in resemblance to speech signals, and therefore its spectrum changes much faster over time as compared to other noise sources. Therefore babble noise will cause a relatively larger distortion to the information contained in the speech signal. With regards to car noise, most of its energy is concentrated at lower frequencies, and therefore its characteristics changes slower with time as compared to speech [84]. Intuitively, larger SNR is obtained for noise classes with characteristics different from speech, e.g. car and traffic noise, which is also confirmed in Figure 5.1.

5.3 Classical Speech Enhancement

As initially stated, the main objective of the speech enhancement algorithms is to increase intelligibility and quality of a noisy signal, by alleviating or eliminating noise artifacts in the observed noisy signal. The focus in the subsequent sections is directed towards state-of-the-art STFT-based speech enhancement algorithms, where two of the most well-known algorithms are described, i.e. spectral subtraction and the Wiener filter. These are considered front-end techniques in the context of ASR, as they attempt noise compensation independent of and outside the folds of the KWS system.

Initially, the signal model considered in the following will be stated, i.e

$$y(n) = x(n) + n(n) \tag{5.1}$$

where $y(n)$ is the noise-contaminated input signal, $x(n)$ is the clean speech signal, and $n(n)$ is the additive noise signal. All entities in the signal model are regarded realizations of a stochastic process, while the signal and noise realizations are assumed to be independent. Taking the STFT on both sides of in Equation (5.1), and given the linear property of the Fourier transform, the corresponding time-frequency (T-F) domain representation is obtained,

$$Y(n,\, \omega) = X(n,\, \omega) + N(n,\, \omega) \tag{5.2}$$

where $Y(n,\ \omega)$, $X(n,\ \omega)$, and $N(n,\ \omega)$ expresses the complex-valued T-F representation of the signal model entities, while n-index is used as a representation of the time frames rather than a single sample. The majority of the classical speech enhancement algorithms depend on either or both the clean signal spectrogram and noise spectrogram, and estimates of these will consequently have a significant impact on the performance of the applied speech enhancement algorithms. Though, it proves that the clean speech signal can be estimated from the speech and noise PSDs [44], these are rarely available in isolation in any practical case, and therefore need to be estimated. However, this is beyond the scope of this thesis, in the following explanation of existing speech enhancement algorithms, the noise and speech PSDs are simply assumed to be known. This assumption is only applied in the following review of the algorithms, and not in the actual implementations.
It should be noted that speech enhancement is essentially a trade-off; on one hand, one desires to maintain the information contained in the speech signal, and on the other hand avoid that noise artifacts remains after the enhancement process [31]. Thus, one must be aware of the existing trade-off between speech distortion and noise suppression.

5.3.1 Spectral Subtraction

Mainly due to its relative simplicity of principle and implementation, spectral subtraction is undoubtedly among the best known acoustical noise suppression algorithms. Assuming additive noise in accordance with the speech model outlined in Equation (5.1), an estimate of the clean speech T-F representation $X(n,\ \omega)$ can be obtained by subtracting the noise T-F representation $N(n,\ \omega)$ from $Y(n,\ \omega)$. The noise-free speech and the noise T-F domain estimates will be denoted $\hat{X}(n,\ \omega)$ and $\hat{N}(n,\ \omega)$, respectively. Hence $\hat{N}(n,\ \omega)$ can be used to estimate $\hat{X}(n,\ \omega)$ as,

$$\hat{X}(n,\ \omega) = Y(n,\ \omega) - \hat{N}(n,\ \omega). \tag{5.3}$$

An alternative representation using the polar form may be used to define $\hat{X}(n,\ \omega)$, by rewriting the T-F representations of $Y(n,\ \omega)$ and $\hat{N}(n,\ \omega)$ on polar form:

$$Y(n,\ \omega) = |Y(n,\ \omega)| \exp\left\{j\phi_y(n,\ \omega)\right\} \tag{5.4}$$

where $|Y(n,\ \omega)|$ is the magnitude spectrum and $\phi_y\ (n,\ \omega)$ is the phase of the noisy speech signal. A similar polar representation of the noise signal can be expressed:

$$N(n,\ \omega) = |N(n,\ \omega)| \exp\left\{j\phi_n(n,\ \omega)\right\}, \tag{5.5}$$

where $|N(n,\ \omega)|$ is the magnitude spectrum and $\phi_n\ (n,\ \omega)$ is the phase of the noise signal. In practical applications, it has been found that the noise phase can be used as a reasonable estimate to the phase of the noise-contaminated speech signal,

i.e. $\phi_y(n,\ \omega) \approx \phi_n(n,\ \omega)$ [40]. Applying this approximation to Equation (5.3) in combination with Equation (5.4) and (5.5) the following expression in obtained,

$$\hat{X}(n,\ \omega) = \left(|Y(n,\ \omega)| - \left|\hat{N}(n,\ \omega)\right|\right) \exp\left\{j\phi_y(n,\ \omega)\right\}. \tag{5.6}$$

An alternative formulation referred to as *power spectral subtraction* [31], estimates the clean speech power spectrum directly from the noise PSD and the observed signal PSD:

$$\left|\hat{X}(n,\ \omega)\right|^2 = |Y(n,\ \omega)|^2 - \left|\hat{N}(n,\ \omega)\right|^2. \tag{5.7}$$

As a finale measure to obtain the clean speech signal, the inverse STFT is applied to Equation (5.6). One of the main disadvantages of spectral subtraction, is the potential of introducing speech distortion. When too much subtraction is applied, speech information is lost during the enhancement process, whereas to little subtraction leaves noise contaminants in the enhanced signal. Another drawback of spectral subtraction, is that estimation errors can cause the expression in (5.6) to become negative, which is meaningless for a magnitude spectrum. A simple solution when encountering such negative values, is to set $\left|\hat{X}(n,\ \omega)\right|$ to zero. Unfortunately, using such a non-linear approach causes a type of distortion referred to as *music-noise* [31], which is visible as isolated peaks in the spectrum occurring at different locations within each time frames. In the time domain, music-noise causes random distortion of the tones at the analysis frame rate. Approaches to overcome music-noise are discussed in [31].
Even though it is relatively easy to understand the underlying principles behind spectral subtraction, as well as implementing the algorithm is affordable, the method has shown to possess several shortcomings. This pertains mainly to the introduction of speech distortion and music-noise.

5.3.2 Wiener Filter

In this section, the Wiener filter will be formulated for the application of speech enhancement, which serves the same purpose as the spectral subtraction algorithm discussed in Section 5.3.1, i.e. to attenuate the additive noise considered in Equation (5.1). Whereas the spectral subtraction method is not derived in a theoretical optimal way, the Wiener filter is optimal in a mean-squared error (MSE) sense under certain statistical assumptions as discussed in the following. The Wiener filter model will shortly be reviewed in the following, together with a description of its asymptotic SNR behavior.
Since the Wiener filter is restricted to be a linear type of filter, the model can be derived from either an infinite impulse response (IIR) filter or an finite impulse response (FIR) filter. For the sake of convenience and due to the fact that FIR filters are inherently stable, the following description will be based on the FIR filter. The Wiener filtering problem is illustrated in Figure 5.2, annotated with relevant model

entities. The filter depicted in the figure is a M-tap FIR filter, hence a delay line of length M is used for the input signal. The pertinent signals in the Wiener filter can be summarized as follows:

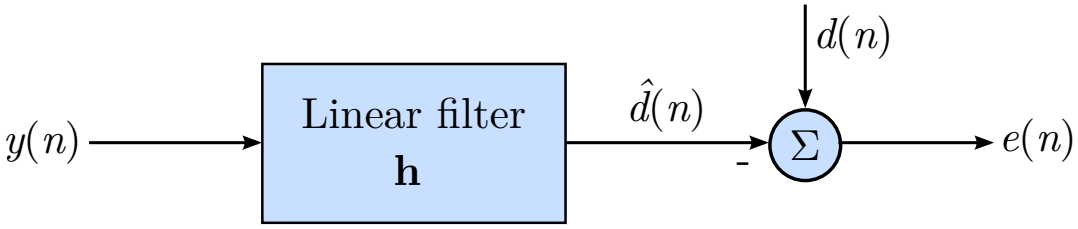


Figure 5.2: Block diagram of the linear model applied in the Wiener filter.

- Input signal,

$$\mathbf{y}(n) = [y(n) \ \dots \ y(n-M+1)]^{\mathsf{T}} \tag{5.8}$$

- M-tap FIR filter with coefficients,

$$\mathbf{h} = [h_0 \ \dots \ h_{M-1}]^{\mathsf{T}} \tag{5.9}$$

- Filter output signal,

$$\hat{d}(n) = \sum_i h_i y(n-i) = \mathbf{y}^{\mathsf{T}}(n)\mathbf{h} \tag{5.10}$$

- Desired zero-mean wide sense stationary (WSS) signal, $d(n)$

- Error signal

$$e(n) = d(n) - \hat{d}(n) = d(n) - \mathbf{y}^{\mathsf{T}}(n)\mathbf{h} \tag{5.11}$$

A basic assumption in the derivation of the Wiener filter, is that the input signal $\mathbf{y}(n)$ and the desired signal $\hat{d}(n)$ are jointly WSS, by definition this implies that they are individually WSS and the cross-correlation sequence is simply a function of the time-lag.
When applying the Wiener filter for speech enhancement, the input signal to the Wiener filter model is the noisy speech input $y(n)$ introduced in Equation (5.1). This further implies that the desired signal in this case corresponds to the noise-free signal, i.e. $d(n) = x(n)$. Under assumption that the noise signal has zero mean, and is uncorrelated with the desired signal, the solution to the Wiener filter can be defined as [31],

$$\mathbf{h} = (\mathbf{R}_{xx} + \mathbf{R}_{nn})^{-1} \mathbf{r}_{xx} \tag{5.12}$$

where $\mathbf{R}_{xx}$ is the auto-correlation matrix of $x(n)$, $\mathbf{R}_{nn}$ is the auto-correlation matrix of $n(n)$, and $\mathbf{r}_{xx}$ is the cross-correlation sequence. According to [74], the Wiener filter solution can be rewritten as a function of the SNR [74]:

$$\mathbf{h} = \left[\frac{\mathbf{I}}{\text{SNR}} + \hat{\mathbf{R}}_{nn}^{-1}\hat{\mathbf{R}}_{xx}\right]^{-1} \hat{\mathbf{R}}_{nn}^{-1}\hat{\mathbf{R}}_{xx}\mathbf{u}_1 \tag{5.13}$$

where the SNR is given by $\frac{\sigma_x^2}{\sigma_n^2}$, $\hat{\mathbf{R}}_{xx} \triangleq \frac{\mathbf{R}_{xx}}{\sigma_x^2}$, $\hat{\mathbf{R}}_{nn} \triangleq \mathbf{R}_{nn}\sigma_n^2$, σ_x^2 is the signal variance, σ_n^2 is the noise variance, and $\mathbf{I}$ is the $M \times M$ identity matrix, of which is first column is denoted $\mathbf{u}_1$. The asymptotic behavior of $\mathbf{h}$ in the cases where the SNR converges to zero and to infinity, can be expressed as follows [74]

$$\lim_{\text{SNR} \to 0} \quad \mathbf{h} = 0 \tag{5.14}$$

and,

$$\lim_{\text{SNR} \to \infty} \quad \mathbf{h} = \mathbf{u}_1 \tag{5.15}$$

The expressions in Equation (5.14) and (5.15), represents the two extremes scenarios of the SNR level; when the SNR is inclined towards infinity, the Wiener filter has very little effects as, $\mathbf{y}^\intercal(n)\mathbf{h} = \mathbf{y}^\intercal(n)\mathbf{u}_1 = \mathbf{y}$, and therefore does not provide any attenuation to the noisy speech signal. Conversely, when the SNR is inclined towards zero, the input signal is highly attenuated and becomes close to zero, resulting in loss of speech information of the input signal.

So far, the Wiener filter has been considered only in the time domain. However, a corresponding T-F representation of the Wiener filter can be expressed, as it is more often implemented in the STFT domain. Such a definition is found in [31],

$$H(n,\, \omega) = \frac{P_{xx}(n,\, \omega)}{P_{xx}(n,\, \omega) + P_{nn}(n,\, \omega)} \tag{5.16}$$

where $P_{xx}(n,\, \omega)$ is the PSD of the noise-free signal, and $P_{nn}(n,\, \omega)$ is the PSD of the noise signal. As a next measure, the Wiener filter can be expressed in terms of the a priori SNR value $\xi(n,\, \omega)$ [31],

$$H(n,\, \omega) = \frac{\xi(n,\, \omega)}{\xi(n,\, \omega) + 1} \tag{5.17}$$

where $\xi(n,\, \omega)$ is the a priori SNR defined as,

$$\xi(n,\, \omega) = \frac{P_{xx}(n,\, \omega)}{P_{nn}(n,\, \omega)}. \tag{5.18}$$

The final T-F representation of the filter output defined in Equation (5.10), is given by,

$$\hat{X}(n,\, \omega) = H(n,\, \omega)Y(n,\, \omega) = \frac{\xi(n,\, \omega)}{\xi(n,\, \omega) + 1}Y(n,\, \omega). \tag{5.19}$$

Note that the asymptotic behavior in the frequency domain given in Equation (5.19), is in line with the asymptotic behavior observed in the time domain according to Equation (5.13). That is, extremely low SNR regions, $\xi \rightarrow 0$, forces $\hat{X}(n, \omega) \approx 0$, and extremely high SNR regions, $\xi \rightarrow \infty$, forces $\hat{X}(n, \omega) \approx 1$.

5.4 Noise Adaptive Model and Template Training

Apart from the investigated state-of-the-art speech enhancement algorithms, we shall explore the use of machine-learning based compensation techniques for the use in KWS under noisy conditions. In particular, an approach similar to that of [75] is considered in the following. Current ASR architectures usually include all the training material in the training of the back-end model, which models a certain phonetic space. However, detection performance is known to degrade significantly in the presence of environmental noise in the speech data, especially when the ASR system is trained exclusively on clean speech data. The introduced noise has two major effects; it generates a distortion of the speech signal in its corresponding representation space, and secondly the random nature of the noise causes a loss of speech information. As for the first effect, it causes a mismatch between the clean training data and the test data, which have become noise-contaminated. Consequently, the acoustic model trained on clean speech data does not model speech data acquired under noisy conditions accurately, which eventually leads to degradation in detection performance. Regarding the second effect, it will also cause a distortion of the speech information, even in the case of optimal mismatch compensation [85].

One approach to mitigate noise mismatches between training and test data is by corrupting the training data with noise sources of similar nature to that of the test data, before the training procedure is applied. This intuition has led to several noisy training strategies [76][77], also known as *multi-style training*. The main motivation behind these methods relates to the principle that machine recognizers typically attain the best performance for a degraded speech input only when the training material is degraded as well [78]. In addition to the noise class, another factor influencing detection performance is the SNR [75]. In order to account for potential SNR variation in the environmental noise, a combined SNR and noise-type dependent training (SNT) strategy will be considered in the following. Adapting SNT into the KWS architecture in Figure 3.1, implies training a distinct set of GMMs, each representing a particular noise type at a specific SNR value. Noise is artificially added at a particular SNR to the training set for each GMM, such that it is trained for its corresponding noise class. Hence, a GMM database is generated which contains a set of pre-trained GMMs covering a set of pre-defined noise types and SNR levels. This is expected to give sharper PDFs within each GMM, as they

are tailored to one specific noise setting, e.g. compared to having a single GMM to handle all types of noise at the different SNRs. Given a set of different noise type and SNR models, keyword detection may be carried out in either two ways:

1. The first option implies a parallel multi-detection structure, where results are generated from all the GMMs. Subsequently, the detection hypothesis is selected based on the GMM which provides the best match, corresponding to minimum distortion.

2. Alternatively, a single GMM is selected for keyword detection according to the noise class and the SNR value estimated from a background noise classifier (BNC).

In similar studies of noisy speech in a car environment [86], the later approach has shown to have much lower computational complexity than the first approach, and a recognition performance which is similar, and in some cases even better.

For the same reasons as pointed out in [86], a similar strategy is adopted which combines the use of a BNC for the selection of a GMM from the model database. Subsequently this method shall be referred to as SNT-KWS. The architecture of the noise compensation framework is presented in Figure 5.3. This method assumes the availability of an ideal background noise classifier (BNC). Since various of such noise classifiers exist already [79][80][81], this will not be our focus point, and we shall simply assume an ideal classifier to be available. Rather, this section serves to investigate the SNT-KWS strategy as a method to mitigate the performance degradation outlined in Figure 5.1, which follows from the corruption of test data for various noise types and at different SNR values.

In addition to the noise adaptive model training applied for the GMMs, the use of noise compensation in the keyword templates will be considered as well. For this purpose an identical training strategy to nullify SNR mismatches between keyword templates and keyword test input will be applied. The basic idea is to expand the existing database of keyword templates outlined in Figure 3.1, with a number of templates corresponding to the number of pre-defined noise scenarios, each of which is corrupted with a particular noise type and SNR (representing one of the pre-defined noise scenarios). The selection of the keyword template for pattern matching is carried out in a similar way of selecting the appropriate GMM, namely according to the noise class and SNR of the noisy test input as estimated by the BNC. This idea also appears from the SNT-KWS architecture depicted in Figure 5.3. As for the SNT-KWS approach, we assume here an ideal BNC is available.

In [75] the impact of SNR mismatches on the ASR performance has been thoroughly studied for an HMM-based recognizer. In this regard the optimal SNR

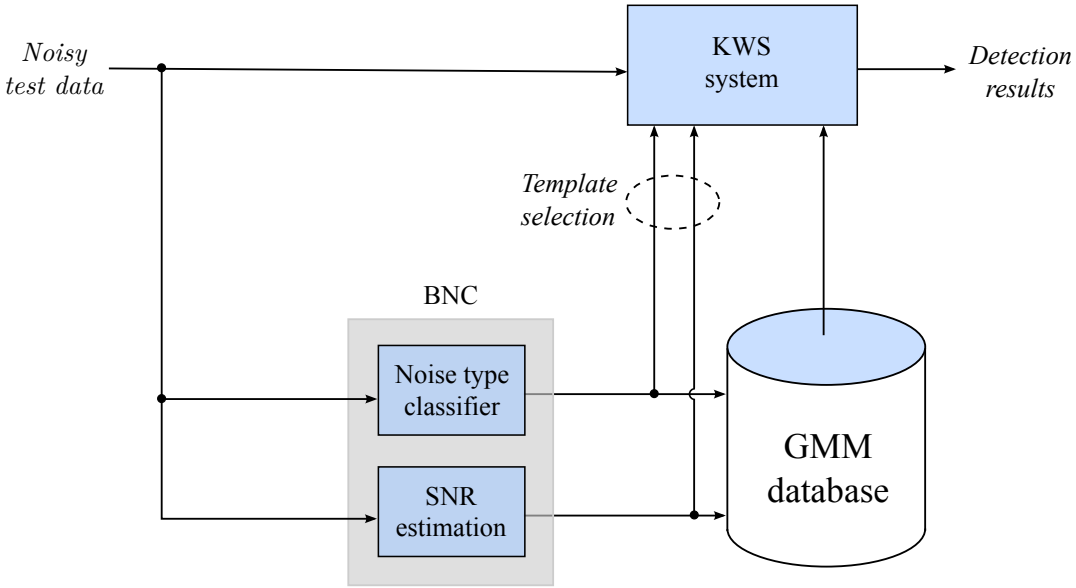


Figure 5.3: SNT-KWS system architecture.

compensation values to be used in the SNT-training procedure have been explored for a number of test noise types and SNR values. The results generally exhibit a linear relation between the optimal SNR compensation values used in the training set and the SNRs of the test set. Hence the optimal performance is achieved where the SNR of the training data is in the close vicinity to that SNR used for the test data, i.e. within a few dB. This further implies that minor mismatches in the SNR of training and test data does not impact detection performance significantly. For low SNRs in particular, the optimal SNR compensation lies above its reference, thereby suggesting that employing a GMM which is trained on higher SNR values may be advantageous in such situations, despite the apparent SNR mismatch.

The proposed SNT-KWS noise compensation method will be subject to a further investigation; as a next step we shall evaluate its relative performance in alleviating the performance degradation due to the mismatch problem. In particular, two distinct implementations of the SNT-KWS will be considered. In the first approach, GMM noise adaption is omitted, rather noise compensation is solely applied for the template instances contained in the keyword template database. This is a considerably simpler and cheaper approach than applying noise compensation to the GMM training data, and does not require the storage of multiple statistical models (here in the meaning of GMMs). Secondly, a merged approach will be considered where template noise compensation is combined with noise adaption in the GMMs such as shown in Figure 3.1. Practical implementations of the Wiener filter and spectral subtraction will be used as a benchmark against the proposed SNT-KWS

framework.

5.4.1 Experiments and Results

For assessment of the discussed front-end (classical speech enhancement) and back-end (model-domain adaption) noise compensation methods, the results in Figure 5.1 will be used as as a frame of reference. Hence it is desired to asses the performance of the noise compensation methods relative to the baseline system, i.e. with no noise compensation, and secondly examine the resulting performance of those methods which attempt noise compensation in one way or another. For this reason, an identical experimental setup to that of Section 5.2 is applied, for the exact same noise sources and SNR values. The implementation of the Wiener filter and the spectral subtraction algorithms are found in [31], for the experiments an iterative Wiener filter algorithm is considered which runs two iteration. For the SNT-KWS framework, a proportional SNR compensation strategy is applied, i.e. the training data is scaled to an SNR equal to that of the noisy test data. This approach is followed in both implementation of the SNT-KWS. In summary, the following five methods will be subject investigation and thus comparison:

- No noise compensation, i.e. simply the baseline system (shortened: No comp.).
- SNT-KWS compensation using keyword templates exclusively (shortened: SNT-KWS-TE).
- SNT-KWS compensation using a combination of keyword templates and GMM training (shortened: SNT-KWS-TR+TE).
- Spectral subtraction (shortened: SS)
- Wiener filter (shortened: WE)

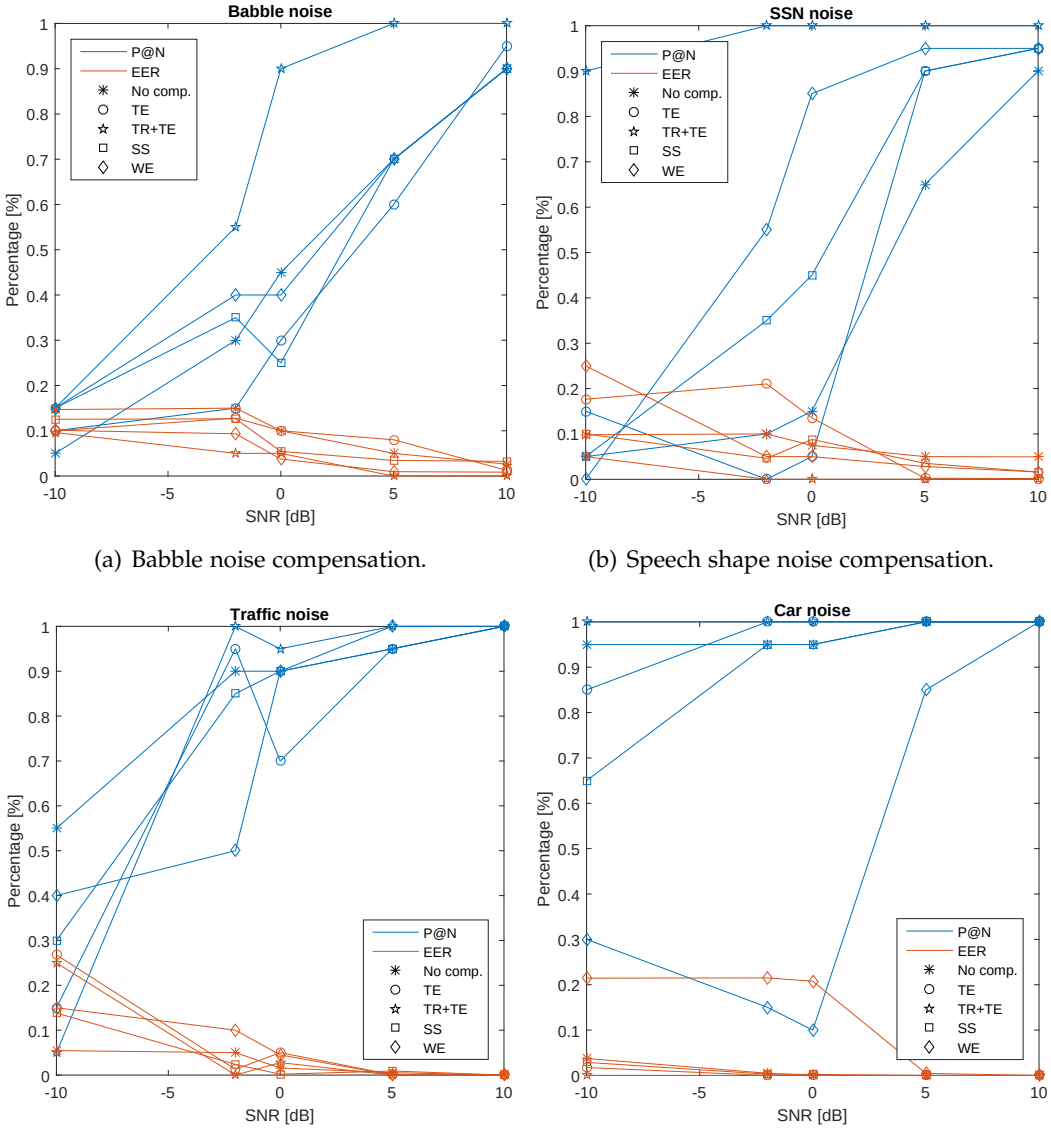
The generated results are shown in Figure 5.4, of which noise compensation in the presence of babble noise, SSN, traffic and car noise are evaluated in Figure 5.4(a), 5.4(b), 5.4(c), and 5.4(d), respectively. Note that SNT-KWS-TE and SNT-KWS-TR+TE in the figures are further shortened to TE and TR+TE, respectively. Each noise scenario, i.e. a certain setting of a noise type plus a SNR value, is quantified by the EER (red) and P@N (blue) metrics. Whereas P@N measures precision of the detected keywords, the EER specifies the point at which the FPR is equal to the FNR, and therefore is an expression of the performance quantified by the ROC curve. Since there is a strong relation between the EER metric and the ROC curve, the plots of the ROC curves will be left out for now.

As evident from the results, there is a considerable difference in the performance of the SNT-KWS methods; the SNT-KWS-TR+TE variant is consistently superior

to the SNT-KWS-TE in all cases. In most cases the SNT-KWS-TE has similar or even worse performance to that of the baseline system. These results signifies that noise adaption of the keyword templates considered in isolation, has no beneficial effects in terms of aiding the keywords spotter in noise robustness. One explanation might be that the loss of speech information in the test input can not simply be compensated by causing a similar loss of information in the keyword template, since the phonetic distortion due to the noise may differ. This presumption pertains especially to non-stationary noise types such as babble noise, which statistical properties changes fast over time. Therefore two sequences of babble noise (such as the one appearing in the test data versus the noise in the keyword template) are unlikely to be similar. This can possibly explain the tendency observed in Figure 5.4(a), where the overall performance of SNT-KWS-TE for the case of babble noise is inferior to the baseline system.

In regard to the SNT-KWS-TR+TE method, it appears to outperforms all the investigated methods in almost all the considered noise scenarios, as it typically attains the highest precision rates while maintaining a relatively low EER. This suggests that the SNT-KWS-TR+TE method has the potential to boost the detection performance in a broad variety of noise environments relative to the baseline KWS system. An example of the relative improvement in detection performance of the SNT-KWS-TR+TE method is expressed in Figure 5.5 using the ROC curve, for a specific noise scenario with babble noise at -2 dB SNR. This clearly shows how SNT-KWS-TR+TE manages to lift the baseline performance, signified as an increase in the AUC, and in this case also outperform the other noise compensation methods. However, at extremely low SNR values, there is no clear indicator of which compensation algorithms that attains the best performance, sometimes it turns more useful to apply no noise compensation at all, this pertains especially to the measured noise types as outlined in Figure 5.4(c) and 5.4(d). Hence the baseline KWS system seems to have an inherent robustness to particular noise types, which may be a property of the selected feature domain representation.

Another interesting finding, is that the classical speech enhancement algorithms, performs better when they are exposed to SSN noise, as opposed to the non-stationary babble noise. This observation fits the underlying theory behind these methods, as they highly depend on the estimation of the noise spectrum. Intuitively, a good noise spectrum is easier to estimate in the presence of stationary noise, like SSN, as compared to a non-stationary noise environment. In addition, the Wiener filter exhibits a relatively large degradation of performance for the measured noise types and mainly in the lower half of the investigated SNR range. As opposed to the Wiener filter performance in the study of SSN, where it performs relatively well and in fact is second best after the SNT-KWS-TR+TE. Hence the per-



(a) Babble noise compensation.

(b) Speech shape noise compensation.

(c) Traffic noise compensation.

(d) Car noise compensation.

Figure 5.4: Results from various noise compensation strategies; i.e. the SNT-KWS-TE and SNT-KWS-TR+TE, as well as spectral subtraction and Wiener filter. The experiments are carried out for babble noise, SSN, trafic and car noise each evaluated at the SNR levels $(10,\ 5,\ 0,\ -2,\ -10)$.

formance of the Wiener filter seem to be highly dependent on the noise type and SNR, and therefore it is not expected to generalize well for different noise settings.

As a concluding remark of the results in Figure 5.4, it should be emphasized that the SNT-KWS-TR+TE methods stands out from the other included methods in

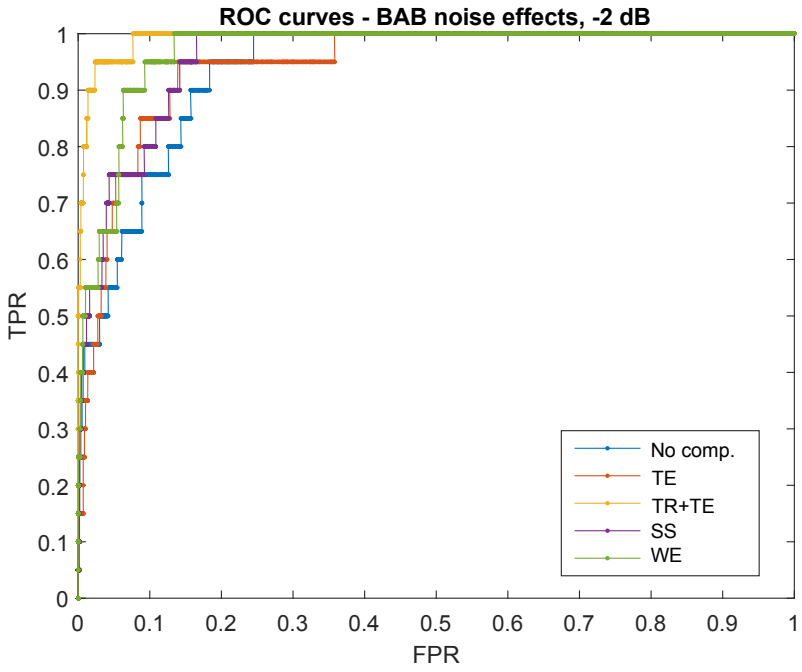


Figure 5.5: Noise compensation performance for the investigated front-end and back-end techniques. Evaluation is carried out for babble noise at -2 dB SNR, quantified by the ROC curve.

three principal aspect:

- SNT-KWS-TR+TE appears to consistently boost noise-robustness of the baseline KWS system under almost any of the investigated noise scenarios. For specific noise scenarios, the SNT-KWS-TR+TE increases precision with up to 50 percentage points relative to the baseline.
- More generally, the experiments have shown that SNT-KWS-TR+TE outperforms both the classical state-of-the-art front-end speech enhancement and SNT-KWS-TE, in more or less all considered noise scenarios.
- As opposed to the other methods, such as the Wiener filter, which attains good performance in specific noise scenarios, the SNT-KWS-TE has proven to generalize well for all considered noise scenarios. Further, it might even have the potential to generalize for unknown noise types, which is a subject that requires further investigation.

The results also suggests that SNR noise compensation is more effective when applied to both the back-end model and the keyword templates, than simply applying isolated keyword template compensation. The involvement of storing multiple statistical models in the combined compensation strategy implies an increased cost in memory footprint, which is an constrained resource for mobile devices such as

an HA. Secondly, SNT-KWS-TR+TE comes at an increased computational cost of training such multiple models. However, this is considered less critical as back-end model training is not a an online procedure, in fact, the models are trained offline and only once, before finally being stored. In the online detection procedure (i.e. the matching of a test input with a keyword template), it is simply a matter of selecting the appropriate GMM based on the estimated noise labels from the BNC.

Chapter 6

Conclusion

This thesis has looked into the field of keyword spotting for the application to personal mobile devices, and with a particular emphasis on the application to hearing aids. To that end, a suitable contemporary KWS system described in [45] has been selected, and evaluated on HA recordings according to its capabilities in mainly two aspects:

1. keyword detection; i.e. to evaluate its detection performance.
2. keyword rejection; i.e. to investigate aspects of speaker dependency, and more generally the rejection capabilities to background words.
3. noise robustness; it has been investigated how existing machine-learning based noise compensation techniques can be adopted into the selected KWS framework to boost noise robustness. The performance of the proposed methods are evaluated relative to the baseline system (i.e with no noise compensation) and additionally to state-of-the-art spectral-based speech enhancement algorithms.

Based on the discussions and results throughout the paper, a concluding remark will be provided for each of the six items in the objectives. For the sake of convenience the objectives are reposted in the following.

Objectives

- Review the fundamental principles behind speech production and speech representation used in the context of keyword spotting.
- Propose a KWS system that takes advantage of state-of-the-art techniques and accommodate the low-power constraints of a personal mobile device.

- Evaluate the selected keyword spotter for a practical mobile communication device by using real-life hearing aid recordings, and compare its performance relative to contemporary KWS systems.
- Investigate the degree of speaker dependency of the selected keyword spotter.
- Suggest various methods to improve noise robustness of the KWS system.
- Evaluate the effect of the proposed noise compensation strategy on the spotting performance, and compare it against state-of-the-art speech enhancement algorithms.

Objective 1

The reviewed speech fundamentals have been described in Chapter 2, based on scientific literature within the field. This covers both temporal and spectral aspects of speech, and treats subjects such as speech production, stationary properties, speech representation domains, etc. Fundamental principles behind speech signals and their characteristics have been reviewed to better understand the intuition behind the methods involved in the different stages of a conventional ASR system. This pertains e.g. to MFCC features which have proven to be a compact representation for aspects of the speech signal which are relevant to speech understanding. Thus prior knowledge of speech fundamentals to some extent is a prerequisite to understand such concepts. The learned principles from speech fundamentals are similarly applied in speech modeling, and for our application to the GMM which attempts partitioning of speech represented in the MFCC domain into its underlying phonetic units.

Objective 2

For the application of a personal (low-power) mobile device, more specifically a HA, a feasible KWS framework has been selected among contemporary KWS algorithms. The selected KWS system described in [45] is based on the template matching paradigm, and generally satisfies the platform constraints of a mobile device as outlined in Section 1.3. The selected architecture uses an unsupervised learning framework that without any transcription information trains a GMM to decode training and test instances into Gaussian posteriorgrams. The Gaussian posteriorgrams are then matched via segmental DTW, and the detection results are obtained from ranking the distortion scores returned by the matching routine.

Objective 3

The detection performance of the KWS system has been evaluated under two different recording setups, of which the first aims at creating an ideal setting for the

recordings and hence involves the use of a studio microphone, whereas a HA microphone is used in the second setup and thus represents the expected recording conditions of a practical mobile communication device. Under controlled acoustic conditions and using a simple test setup of only one keyword, the results have been overwhelmingly positive, in terms of the reported performance metrics (Table 4.2). The micHQ results, exhibits 98 % accuracy among the classified distortion scores, while the precision observed in the micHA case is 88.8 % and thus slightly inferior, but nonetheless still attains high precision. Further, the micHQ results attain optimal ERR equal to zero, also in this aspect the micHA results show slightly degraded performance, with an EER of 2 %. These results are also observable from the shape of reported ROC curves, as the AUC in these cases are close to or equal to one.

A more comprehensive experiment conducted in Section 4.2.3, where a more challenging test setup is applied in line with the evaluation procedure applied in some of the most recent keyword spotters [45][47][87]. This implies the use of multiple keywords each with multiple occurrences in the experiment. The results (Table 4.3) show that the selected KWS system is competitive to that of the state-of-the-art keywords spotters; in terms of the P@N, the micHQ is only surpassed by the phonetic posteriorgram+SDTW approach, while both micHQ and micHA have the lowest EER among all the considered methods. Furthermore, the considered KWS system exhibits superior performance in terms of the MAP metric.

Objective 4

Assessing the speaker dependency of the KWS system is a bit more challenging than keyword detection as there exist no explicit evaluation metric for this purpose. The evaluation metrics currently considered depend on true keyword occurrences (i.e. keywords in the test string uttered by the native system user), e.g. to measure the keyword detection precision among the top ranked distortion scores as in the case of the P@N and MAP metrics. Whereas the extraction of ROC curves (and therefore also the EER) depend on true keyword occurrences to compute the TPR, which is not possible in the absence of true keyword occurrences in the test string. However, a comparison between the results from the keyword rejection experiments with that of the keyword detection results, suggests that a threshold can be found which avoids false positives and therefore rejects keyword occurrences uttered by a non-native system user (i.e. false keyword occurrences), while maintaining a good detection rate of the true keyword occurrences.

Objective 5

A keyword spotting algorithm applied in a real-world scenario will inevitably be exposed to a large variety of noise sources at different SNRs. Therefore it was explored how noise robust the keyword spotter is to a set of pre-defined noise types and SNR values, and assess the resulting effect on the detection performance. To compensate for the performance degradation due to the noise, several noise compensation strategies have been proposed. In this thesis we have explored a back-end noise compensation strategy to cancel SNR mismatches between training and test data, i.e. by using noise adaptive GMM and template training. This method is denoted SNT-KWS. The GMMs and keyword templates are adapted to a specific noise type and SNR, and subsequently stored in a database. An ideal noise classifier is assumed to be available for estimating the noise source and SNR of the noisy input speech, such that an appropriate keyword template and GMM can be selected based on the estimated noise label.
Two variants of the SNT-KWS method have been considered; a simple and training-wise cheap approach which implements only template compensation, i.e. SNT-KWS-TE, and secondly a merged approach is considered where template noise compensation is considered in combination with noise adaptive GMM training, i.e. SNT-KWS-TR+TE. The computational complexity of the online detection procedure is the same in the two cases, however SNT-KWS-TR+TE comes with an increased cost in the memory footprint for storing the various GMMs. Hence, one should exercise caution regarding the number of considered noise scenarios, as memory is a constrained resource in mobile platforms.

Objective 6

The resulting plots from the noise compensation experiments (Figure 5.4) show that SNT-KWS-TR+TE outperforms both the classical state-of-the-art front-end speech enhancement algorithms and SNT-KWS-TE, in more or less all considered noise scenarios. As opposed to the other methods, such as the Wiener filter, which attains good performance in specific noise scenarios, the SNT-KWS-TE has proven to generalize well for all considered noise scenarios. Further, it might even have the potential to generalize for unknown noise types, which is a subject that requires further investigation.
In summary, the SNT-KWS-TR+TE appears to consistently boost noise-robustness of the baseline KWS system under almost any of the investigated noise scenarios. For specific noise scenarios, the SNT-KWS-TR+TE increases precision with up to 50 percentage points relative to the baseline.

Final Remark

Throughout this thesis, the application of keyword spotting has been explored for the application of hearing aids. Through an experimental approach, it was found that the selected KWS system described in [45], has the feasibility and an adequate detection performance to be implemented in a practical mobile device, under the recording circumstances of which the system has been evaluated. To simulate the KWS system behavior in a real-life recording scenario using a mobile device, HA recordings have been conducted, on which the system performance is evaluated. The results (Table 4.2 and 4.3) have revealed that no significant performance degradation followed from the use of micHA recordings as compared to the micHQ recordings. Furthermore, the KWS system has shown to be competitive to recent keyword spotters for the P@N measure, while it is superior regarding the EER and MAP metrics. However, these results need to be considered with caution, as the training and test data typically differ in relation to the applied speech corpus, and e.g. in the number of keywords and keyword repetitions.

Based on these observations, the investigated KWS framework is considered practically applicable under the acoustic circumstances considered in the experiments. However, it is strongly expected that the considered keyword spotter can maintain a decent performance in broader variety of acoustical environments besides those considered in the experiments, by incorporating elements of noise robustness into the system architecture. In particular, the SNT-KWS-TR+TE method has shown to mitigate the performance degradation followed by noise corruption of the test data for a range of different noise scenarios. The main concern with SNT-KWS-TR+TE from a practical point of view is that the number of pre-defined noise scenarios must be kept at a minimum, in consideration of the memory footprint. The results observed so far (Figure 5.4) indicates that the SNT-KWS-TR+TE generalizes well for known noise types, therefore it is worth investigating the generalization ability for a broader range of known and unknown noise scenarios when using only a restricted number of GMMs. This can potentially be accommodated by modifying the current training strategy, such that each GMM is trained for a range of differnt noise sources and SNRs. However, this is a subject for further investigation. Additionally, a practical application requires the ideal BNC to be replaced with an actual implementation. In this respect, exact noise labeling of the SNR levels is not considered an absolute must, since the findings in [75] allow for a deviation in the estimated SNR level without any considerable loss in performance. In summary, the methods investigated throughout the thesis is believed by the author to have great potential for a practical implementation, and successfully be able to detect keyword utterances in real-life applications.

Bibliography

[1] D. O'Shaughnessy, "Interacting With Computers by Voice: Automatic Speech Recognition and Synthesis," *Proceedings of the IEEE*, vol.91, no.9, pp.1272-1305, Sept. 2003.

[2] Apple Inc., "iOS 9 Siri," `http://www.apple.com/uk/ios/siri/`, visit date: 09/11-2015.

[3] Microsoft, "Microsoft Cortana," `http://www.microsoft.com/en-us/mobile/experiences/cortana/`, visit date: 09/11-2015.

[4] Google, "Google Now," `https://www.google.com/landing/now/`, visit date: 09/11-2015.

[5] Amazon, "Amazon Echo/Alexa," `http://www.amazon.com/Amazon-SK705DI-Echo/dp/B00X4WHP5E`, visit date: 09/11-2015.

[6] S. Irtza, K. Rehman and S. Hussain, "Urdu Keyword Spotting System using HMM," *Conference on Language and Technology 2014* (CLT 14), Karachi, Pakistan.

[7] G. Chen, "Low Resource Keyword Spotting," Dept. of Electrical and Computer Engineering, Johns Hopkins University, Dec. 2014.

[8] M. Benzeghib, et. al, "Automatic speech recognition and speech variability: A review," Elsevier, *Speech Communication* 49 (2007) 763–786.

[9] Joseph Keshet, David Grangier, and Samy Bengio, "Discriminative keyword spotting," *Speech Commun.* 51, 4 (April 2009), 317-329.

[10] J. Davis and M. Goadrich, "The relationship between Precision-Recalland ROC curves," in *Proceedings of the 23rd international conference on Machine learning ICML 06*, vol. 10, no. 2, 2006, pp. 233–240.

[11] J. S. Bridle, "An efficient elastic-template method for detecting in given words in running speech," *British Acoustic Society Metting*, pp. 1-4, 1973.

[12] H. Sakoe and S. Chiba, "A Dynamic Programming Approach to Continuous Speech Recognition," in *Proceedings of the Seventh International Congress on Acoustics*, Akadémiai Kiadó, Budapest, pp. 65-69, 1971.

[13] A. J. K. Thambiratnam, "Acoustic Keyword Spotting in Speech with Applications to Data Mining," Queensland University Of Technology Brisbane, Queensland, March 2005.

[14] Guoguo Chen, Parada, C. and Sainath, T.N., "Query-by-example keyword spotting using long short-term memory networks," in *Acoustics, Speech and Signal Processing (ICASSP)*, 2015 IEEE International Conference on, pp.5236-5240, 19-24 April 2015.

[15] Guoguo Chen; Parada, C.; Heigold, G., "Small-footprint keyword spotting using deep neural networks," in *Acoustics, Speech and Signal Processing (ICASSP)*, 2014 IEEE International Conference on, pp.4087-4091, 4-9 May 2014.

[16] R.C.Rose and D. B. Paul, "A hidden Markov model based keyword recognition system," in *Acoustics, Speech, and Signal Processing, 1990. ICASSP-90.*, 1990 International Conference on, p.129-132 vol.1, 3-6 Apr 1990.

[17] M. Weintraub, "LVCSR log-likelihood ratio scoring for keyword spotting," in *Acoustics, Speech, and Signal Processing, 1995. ICASSP-95.*, 1995 International Conference on, pp.297-300 vol.1, 9-12 May 1995.

[18] A. Mandal , K. R. Prasanna Kumar and Pabitra Mitra, "Recent developments in spoken term detection: a survey," in *International Journal of Speech Technology*, June 2014, Volume 17, Issue 2, pp 183-198.

[19] D. Hakkani-Tur and G. Riccardi, "A general algorithm for word graph matrix decomposition," in *Acoustics, Speech, and Signal Processing*, 2003. Proceedings. (ICASSP '03). 2003 IEEE International Conference on, pp.I-596-I-599 vol.1, 6-10 April 2003.

[20] L. Mangu, E. Brill and A. Stolcke, "Finding consensus in speech recognition: word error minimization and other applications of confusion networks," in *Computer Speech & Language*, Volume 14, Issue 4, October 2000, Pages 373-400.

[21] M. Weintraub, "LVCSR log-likelihood ratio scoring for keyword spotting," in *Proc. ICASSP*, pp. 129–132, 1995.

[22] A. Sangwan and J.H.L. Hansen, "Keyword recognition with phone confusion networks and phonological features based keyword threshold detection," in *Signals, Systems and Computers (ASILOMAR)*, 2010 Conference Record of the Forty Fourth Asilomar Conference on, pp.711-715, 7-10 Nov. 2010

[23] K. Ng and V. W. Zue., "Subword-based approaches for spoken document retrieval", *Speech Communication* 32, 3 (October 2000), 157-186.

[24] I. Szoke et al., "Sub-word modeling of out of vocabulary words in spoken term detection," in *Proceedings of Spoken Language Technology (SLT) Workshop*, IEEE. 2008, pp. 273–276.

[25] J. Holmes and Wendy Holmes, "Speech Synthesis and Recognition," CRC Press, 2.nd ed., 2001.

[26] J. Liénard, "Speech and Voice Perception: Beyond Pattern Recognition," in *Speech Processing, Recognition and Artificial Neural Networks: Proceedings of the 3rd International School on Neural Nets*, Springer London, pp.85-112, 2010.

[27] H. Lundqvist, I. Más, and G. Karlsson, "Edge-Based Differentiated Services," Proc. IWQoS, Passau, Germany, June 2005.

[28] P. Denes and E. Pinson, "The Speech Chain: The Physics and Biology of Spoken Language," New York: Worth Publishers, 1993.

[29] K. Hayward, "Experimental Phonetics ," Taylor And Francis, 2000.

[30] T. F. Quatieri, "Discrete-Time Speech Signal Processing: Principles and Practice," Prentice Hall PTR, 2002.

[31] P. C. Loizou, "Speech Enhancement: Theory and Practice," CRC Press, 2007.

[32] A. O'Cinneide, D. Dorran, and M. Gainza (2008), "Linear Prediction: The Problem, its Solution and Application to Speech," DIT Internal Technical Report, 2008.

[33] Online etymology dictionary, "OED," `http://www.etymonline.com/index.php?term=format&allowed_in_frame=0`, visit date: 09/04-2016.

[34] X. Huang, A. Acero , and H.-W. Hon (2001), "Spoken Language Processing - A Guide to Theory, Algorithm and System Development," Prentice Hall, 1. edition.

[35] K. Kumar, C. Kim and R. M. Stern, "Delta-spectral cepstral coefficients for robust speech recognition," *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Prague, 2011, pp. 4784-4787.

[36] J. W. Picone, "Signal Modeling Techniques in Speech Recognition," Proceedings of the IEEE, Vol. 81, No. 9, pp. 1215-1247, September 1993.

[37] J. Jensen, and Z.-H. Tan, "Minimum mean-square error estimation of mel-frequency cepstral features - a theoretically consistent approach," *IEEE/AMC Transactions on Audio, Speech and Language Processing*, Volume:23, No: 1, pages 186–197, 2015.

[38] M. Woelfel and J. McDonough, "Distant Speech Recognition," John Wiley & Sons, 2009.

[39] U. Shrawankar and V. Thakare, "Techniques for Feature Extraction In Speech Recognition System : A Comparative Study," *CoRR*, abs/1305.1145 (2013).

[40] J. R. Deller, J. H. Hansen, and J. G. Proakis, "Discrete-Time Processing of Speech Signals," Wiley Interscience, 1. edition, 1993.

[41] S. Young, "A review of large-vocabulary continuous-speech," in *IEEE Signal Processing Magazine*, vol. 13, no. 5, pp. 45-, Sept. 1996.

[42] B. Logan et al., "Mel frequency cepstral coefficients for musicmodeling," in *ISMIR*, 2000.

[43] J. W. Picone, "Signal modeling techniques in speech recognition," in *Proceedings of the IEEE*, vol. 81, no. 9, pp. 1215-1247, Sep 1993.

[44] R. C. Hendriks, T. Gerkmann, and J. Jensen, " DFT-Domain Based Single-Microphone Noise Reduction for Speech Enhancement: A Survey of the State of the Art", Synthesis Lecture on Speech Processing and Audio Processing, 2013.

[45] Zhang, Yaodong, and James R. Glass, "Unsupervised Spoken Keyword Spotting via Segmental DTW on Gaussian Posteriorgrams," in *Proceedings of the 2009 IEEE Workshop on Automatic Speech Recognition & Understanding (ASRU 2009) IEEE*, 2009. 398–403.

[46] Z. H. Tan and B. Lindberg, "Low-Complexity Variable Frame Rate Analysis for Speech Recognition and Voice Activity Detection," in *IEEE Journal of Selected Topics in Signal Processing*, vol. 4, no. 5, pp. 798-807, Oct. 2010.

[47] T. J. Hazen, W. Shen and C. White, "Query-by-example spoken term detection using phonetic posteriorgram templates," *Automatic Speech Recognition & Understanding, 2009. ASRU 2009. IEEE Workshop on*, Merano, 2009, pp. 421-426.

[48] G. Aradilla, J. Vepa and H. Bourlard, "Using posterior-based features in template matching for speech recognition," *Proc. Interspeech-2006*, 2006.

[49] S. Madikeri, "Speaker Verification and Keyword Spotting Systems for Forensic Applications," Ph.D. thesis, IIT Madras.

[50] D. A. Reynolds, "Automatic speaker recognition using gaussian mixture speaker models," The Lincoln Laboratory Journal, pp. 173-192, 1995.

[51] D. A. Reynolds and R. C. Rose, "Robust text-independent speaker identification using Gaussian mixture speaker models," in *IEEE Transactions on Speech and Audio Processing*, vol. 3, no. 1, pp. 72-83, Jan 1995.

[52] D. A. Reynolds, "Speaker identification and verification using Gaussian mixture speaker models," Speech Commun. 17, 1-2 (August 1995), 91-108.

[53] C. M. Bishop, (8 th printing, 2009), "Pattern Recognition and Machine Learning,", Springer., 1. edition.

[54] CMU Statistics, "Old Faithful Geyser Data," `http://www.stat.cmu.edu/~larry/all-of-statistics/=data/faithful.dat`, visit date: 05/05-2016.

[55] G. Aradilla, H. Bourlard and M. Magimai-Doss, "Posterior featuresapplied to speech recognition tasks with user-defined vocabulary," in *Proc. ICASSP*, Taipei, 2009.

[56] E. Tsiporkova, "Dynamic Time Warping Algorithm for Gene Expression Time Series", `http://www.psb.ugent.be/cbd/papers/gentxwarper/DTWAlgorithm.ppt`, visit date: 09/05-2016.

[57] M. Müller, "Dynamic time warping," in *Information Retrieval for Musicand Motion*, Berlin, Germany: Springer, 2007, pp. 69–84.

[58] A. S. Park and J. R. Glass, "Unsupervised Pattern Discovery in Speech," in *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 16, no. 1, pp. 186-197, Jan. 2008.

[59] M. Blondel, "Introduction to Dynamic Time Warping," (2009), `http://www.mblondel.org/journal/2009/08/31/dynamic-time-warping-theory/`, visit date: 09/05-2016.

[60] H. Sakoe and S. Chiba, "Dynamic Programming Algorithm Optimization for Spoken Word Recognition," in *IEEE Transactions on Acoustics Speech and Signal Processing*, 26(1):43-49, March 1978.

[61] F. Itakura, "Minimum prediction residual principle applied to speech recognition," in *IEEE Transactions on Speech and Audio Processing*, 23:52–72.

[62] E. Keogh and C. A, Ratanamahatana, "Exact indexing of dynamic time warping", in *Knowledge and Information Systems*, March 2005, Volume 7, Issue 3, pp 358-386.

[63] K. F. Man and K. S. Tang, "Genetic algorithms for control and signal processing," *Industrial Electronics, Control and Instrumentation, 1997. IECON 97. 23rd International Conference on*, New Orleans, LA, 1997, pp. 1541-1555 vol.4.

[64] K. R. Rao and P. Yip, "Discrete Cosine Transform: Algorithms, Advantages, Applications," Academic Press, Boston, 1990.

[65] D. Zhang, "Automated Biometrics: Technologies and Systems," Seattle,WA, USA: Kluwer, 2000.

[66] Manish Gupta and Michael Bendersky, "Information Retrieval with Verbose Queries," Foundations and Trends in Information Retrieval, vol. 9, no. 3-4, pp. 209–354, 2015.

[67] Eric Conrad, Chapter 1 - Domain 1: Access Control, In Eleventh Hour CISSP, Syngress, Boston, 2014, Pages 1-21.

[68] H. Wang, T. Lee and C. C. Leung, "Unsupervised spoken term detection with acoustic segment model," *Speech Database and Assessments (Oriental COCOSDA), 2011 International Conference on*, Hsinchu, 2011, pp. 106-111.

[69] M. Brookes, "VOICEBOX: Speech Processing Toolbox for MATLAB," `http://www.ee.ic.ac.uk/hp/staff/dmb/voicebox/voicebox.html`, visit date: 15/05-2015.

[70] D. Ellis, "Dynamic Time Warp (DTW) in Matlab," `http://labrosa.ee.columbia.edu/matlab/dtw/`, Lab for Recognition and Organization of Speech and Audio (LabROSA), Columbia University, visit date: 15/05-2015.

[71] B. Liu, "Web data mining: exploring hyperlinks, contents, and usage data," Springer, Berlin, 2011.

[72] TIMTI Corpus, "TIMIT Acoustic-Phonetic Continuous Speech Corpus"`https://catalog.ldc.upenn.edu/LDC93S1`, Linguistic Data Consortium, University of Pennsylvania.

[73] Y. Chung and J. H. L. Hansen, "Compensation of snr and noise typemismatch using an environmental sniffing based speech recognition so-lution," J. Audio, Speech, Music Process., vol. 2013.1, pp. 1–14, 2013.

[74] J. Chen, J. Benesty, Y. A. Huang, and S. Doclol, "New insights into the noise reduction wiener filter," in *IEEE Transactions on Audio, Speech and Language Processing*, 14(2):1218–1234, 2006.

[75] H. Xu, P. Dalsgaard, Z. H. Tan and B. Lindberg, "Noise Condition-Dependent Training Based on Noise Classification and SNR Estimation," in *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 15, no. 8, pp. 2431-2443, Nov. 2007.

[76] S. Das, A. Nadas, D. Nahamoo, and M. Picheny, "Adaptation techniques for ambience and microphone system,"in *Proc. IEEE Int. Conf.Acoust., Speech, Signal Process.*, 1994, vol. 1, pp. 21–23.

[77] R. P. Lippmann, E. A. Martin, and D. B. Paul, "Multi-style training forrobust isolated-word speech recognition," in *Proc. ICASSP'87*, 1987,pp. 705–708.

[78] Richard P. Lippmann, "Speech recognition by machines and humans," in *Speech Communication*, Volume 22, Issue 1, July 1997, Pages 1-15.

[79] F. Saki and N. Kehtarnavaz, "Background noise classification using random forest tree classifier for cochlear implant applications," *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Florence, 2014, pp. 3591-3595.

[80] C. Couvreur et. al, "Automatic Classification of Environmental Noise Events by Hidden Markov Models," Applied Acoustics, Volume 54, Issue 3, July 1998, Pages 187-206

[81] J. M. Kates, JM, "Classification of background noises for hearing aid applications," The Journal of the Acoustical Society of America, 97(1):461-70, 1995.

[82] U. B. P. Das, "Performance Evaluation of Wiener Filter and Kalman Filter Combined with Spectral Subtraction in Speaker Verification System," International Journal of Innovative Technology and Exploring Engineering (IJITEE) ISSN: 2278-3075, Volume-2, Issue-2, January 2013.

[83] DEMAND Corpus, "Diverse Environments Multichannel Acoustic Noise Database," `http://parole.loria.fr/DEMAND/`, visit date: 30/05-2015.

[84] J. Benesty, J. Chen, Y. Huang, I. Cohen, "Noise Reduction in Speech Processing," vol. 2 of Springer Topics in Signal Processing, Springer, Berlin Heidelberg, 2009.

[85] A. D. L. Torre et. al, "Speech Recognition Under Noise Conditions: Compensation Methods," in Robust Speech Recognition and Understanding, pp.460, I-Tech, Vienna, Austria, June 2007.

[86] M. Akbacak and J. H. L. Hansen, "Advances in acoustic noise trackingfor robust in-vehicle speech systems," in *Advances for In-Vehicle and Mobile Systems*,

H. Abut, J. H. L. Hansen, and K. Takeda, Eds.NewYork: Springer, 2007, ch. 10, pp. 109–122.

[87] A. Muscariello, G. Gravier, F. Bimbot, "Zero-resource audio-only spokenterm detection based on a combination of template matching techniques," INTERSPEECH2011: 12th Annual Conference of the International Speech Communication Association, Aug2011, Florence, Italy. 2011.

Appendix A

Measurement Report

The purpose of this experiment is to conduct speech recordings for the evaluation of the KWS system described in Chapter 3. The experimental data will be used in relation to both test and training purposes. All recordings are made in a controlled acoustical environment, to generate as clean speech signals as possible. This implies that recordings are conducted in a acoustic isolated room to avoid interference from external noise sources, the distance between microphone and speaker is relatively short. In particular, recordings are first carried out using a conventional studio microphone, and secondly with the microphones in a commercial hearing aid (HA). The experimental setup used for the HA recordings attempts to simulate real case scenarios of a speaker with a personal mobile device.

All recordings are carried out for an equal amount of male and female speakers, with a total of number of 10 different speakers. Each speaker was asked to repeat each of the nine keywords in Table A.1 thirty times. Subsequent to the keyword recordings, the speakers were asked to read out the same newspaper article (containing 680 words) in order collect some general speech data. The duration of each of the 9 keyword recordings (with each 30 repetitions) is approximately 1 minute, while the article takes around 5 minutes to complete, resulting in roughly 15 minutes of speech data from each test person.

The experiments described in the following aim to evaluate the KWS system through the data acquired in the recordings. First and foremost, it is investigated how well the keyword spotter serves its main purpose of detecting spoken keywords uttered by the native system-user. While the second objective is to explore the ability of the keyword spotter to reject instances of keywords uttered by a non-native user.

#	Keyword
1	Oticon Alta
2	Oticon Epoch
3	Oticon Agile
4	Program 1
5	Program 2
6	Program 3
7	Volume up
8	Volume up
9	Go to sleep

Table A.1: Selected keywords for the recordings.

A.1 Measurement Equipment

For the recording setup, the following equipment have been used:

Equipment	Description
Microphone	RØDE NT2000
Microphone stand	-
Hearing aid	Oticon Vigo recording set
Laptop	Lenovo T450s
USB sound card	Edirol UA-25EX
Head and torso simulator	-

Table A.2: Measurment equipment.

A.2 Experimental Setup

In the following, two different setups are described, one used for regular microphone recordings, while the second setup is used for hearing aid recordings.

A.2.1 Microphone Setup

The setup used for the regular microphone recordings is shown in Figure A.1, where the microphone is placed on a regular table 30 cm from the speaker's mouth. The recorded signals are sampled through an external sound card connected to the microphone through an analogue input port. All signals are then collected in MATLAB running on a laptop. In between the speaker and the sampling equipment, a wall partition[1] is placed to attenuate any type of interference between the two.

[1] i.e. a movable sort of wall that is build from sound absorbing material, but is commonly is used to divide the interior space of a room.

All signals are sampled at 44.1 kHz with a resolution of 16 bit. To reduce computational complexity of running the KWS algorithms, the signals are down-sampled to 8 kHz in MATLAB.

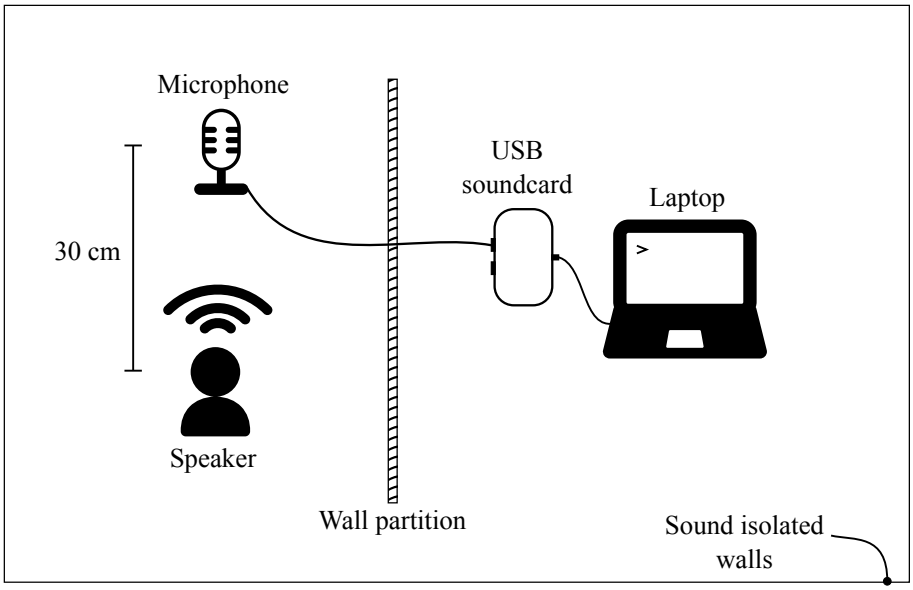


Figure A.1: Experimental setup for the first test scenario.

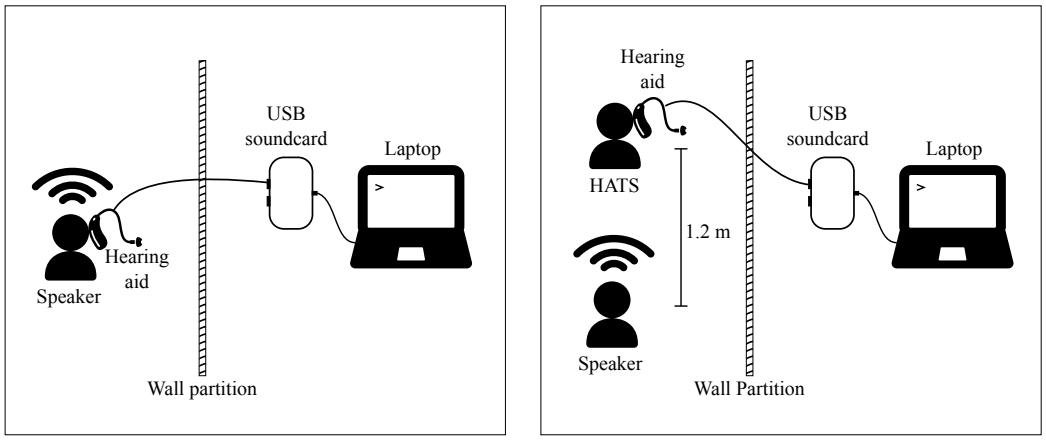
A.2.2 Hearing Aid Setup

A somewhat similar setup to the one in Figure A.1 is used for the hearing aid recordings. But, instead of using a single setup for both keyword detection and rejection testing, the HA setup is split in two corresponding to the number test scenarios, these are shown in Figure A.2. The HA experiments closer resemble a real case scenario mainly in three aspects:

- Location of the microphone
- Quality of the microphone
- Distance to the microphone

The first test scenario depicted in Figure A.2(a), is used for evaluating the detection performance. In this case, the HA microphone is used for the recordings, with the HA placed behind the ear of the speaker during the recordings of the keywords and the newspaper article. The second setup shown in Figure A.2(b), is used to mimic an attempt of a non-native user to trigger the keyword detector. A head and torso simulator (HATS) is equipped with the HA, and is displaced within a speaking distance from the speaker, more specifically 1.2 meter is used in the

experiment. In this case the HATS plays the role of the native system-user, while the speaker mimics the non-native user attempting to trigger the keyword detector by uttering the keyword. Therefore only keyword recordings are carried out in the second test scenario.



(a) Keyword detection; HA experimental setup. (b) Keyword rejection; HA experimental setup.

Figure A.2: Hearing aid experimental setup for the two test scenarios.

A.3 Error Sources

The main source of error in the recordings, are due to the speaker pronunciation and rate variability. During the experiments it was clear that certain test speakers unintentionally changed the pronunciation of the keywords significantly, due to the many consecutive repetitions of the same keyword. Consequently, keyword templates generated from the recordings may differ greatly from the generated keyword test set, thereby causing a mismatch between template and test instances. In more general terms, too large variability of the recorded signals can cause the training data to differ significantly from the test data. Furthermore, several problems have been encountered when the speaker talks either too fast or too slow. This pertains mainly to the process of partitioning the recorded keywords into individual audio files via the VAD. In particular, pronouncing the keywords too fast leaves just a narrow time gap between each repetition, and therefore can cause the VAD to detect multiple keyword repetitions as one. Conversely, speaking too slow can cause the VAD to cut the keywords in halves, this applies especially to composed keywords like “go to sleep”. So far, this problem has been solved by manually tuning the VAD parameters for each speaker and each keyword such that it separates the keyword repetitions correctly. However, in a practical scenario the issues of pronunciation and rate variability during the recording session of the

keywords need to be dealt with. To this end, the by far simplest solution is to define clear guidelines for how the recordings must be conducted by the system user, e.g. demanding a pause of a specific duration between each keyword repetition and instructing the system user to speak loud and clear during the recordings. Lastly, it should be mentioned that sound reflections from either the table or the wall partition may be captured by the microphone.

Appendix B

CD content

The content on the CD includes:

- Digital copy of the thesis
- MATLAB scripts used during the project
  - `KWS_experiment.m`: Contains the baseline keyword spotter.
  - `Evaluation.m`: Evaluates the performance of the results obtained by `KWS_experiment.m` and plots an ROC curve.
- MATLAB toolboxes used for the KWS algorithm and evaluation