Efficient Recursive Speaker Segmentation for Unsupervised Audio Editing

Thor Bundgaard Nielsen

Kongens Lyngby 2013
M.Sc.-2013-62

DTU Compute
Technical University of Denmark
Matematiktorvet, building 303B, DK-2800 Kgs. Lyngby, Denmark
Phone +45 4525 3031, Fax +45 4588 1399
compute@compute.dtu.dk
www.compute.dtu.dk M.Sc.-2013-62

Summary

Today nearly everyone carries a microphone every waking moment. The world and particularly the internet is awash with digital audio. This excess generates demand for tools, using machine learning algorithms, capable of organisation and interpretation. Thereby enriching audio and creating actionable Information.

This thesis tackles the problem of speaker diarisation, answering the question of "Who spoke when?", without the need for human intervention. This is achieved through the design of a custom algorithm that when given data, automatically designs an algorithm capable of solving this problem optimally.

Initially this thesis scans the field of change-detection in general. A diverse variety of methods are studied, compared, contrasted, combined and improved. A subgroup of these methods are selected and optimised further through a recursive design. Beyond this, the raw audio is processed using a model of the speech production system to generate a sequence of highly descriptive features. This process deconvolves an auditory fingerprint from the literal information carried by speech.

Given data from normal conversation, between an arbitrary number of people, the generated algorithm is capable of identifying almost 19 out of 20 speaker changes with very few false alarms. The algorithm operates 5 times faster than real-time on a contemporary PC and subsequently answers the "who" by comparing the speaker turns and assigning labels.

The work carried out in this thesis is of particular practical use in the field of audio editing.

Resumé

Næsten alle bærer en mikrofon hvert vågent øjeblik. Verdenen og især internettet er oversvømmet med digital lyd. Dette overskud genererer efterspørgsel efter værktøjer som, ved brug af maskine-lærings algoritmer, kan håndtere organisering og fortolkning. Derved beriges lyd og handlingsrettet information skabes.

Denne afhandling tackler spørgsmålet om "Hvem talte, hvornår?", uden behov for menneskelig indgriben. Dette opnås gennem design af en nyskabende algoritme, som ved brug af data automatisk designer en algoritme i stand til at løse dette problem optimalt.

Efter en dybere gennemgang af teorien bag ændrings-detektion i sin helhed, anvendes en mangfoldig række metoder. Disse metoder bliver undersøgt i detaljen, hvorefter de sammenlignes, kontrasteres, kombineres og forbedres. En undergruppe af disse metoder bliver valgt og optimeres derefter ved brug af et rekursivt design. Ud over dette, er den rå lyd forarbejdet ved anvendelsen af en model for tale-produktions-systemet. Denne anvendes til at generere en sekvens af højt beskrivende attributter, som affolder et auditivt fingeraftryk fra den bogstavelige informationen båret af talen.

Givet data fra normale samtaler, mellem et vilkårligt antal mennesker, er den genererede algoritme i stand til at identificere næsten 19 ud af 20 taler-skift. Disse identificeres med meget få falske alarmer og algoritmen operere 5 gange hurtigere end realtid på en moderne PC. Derefter besvares "hvem" ved at sammenligne udsagn og tildele etiketter.

Metoder udviklet i denne afhandling er af særlig praktisk anvendelse inden for lydredigering.

Preface

This Master's thesis was carried out at the department of Applied Mathematics and Computer Science in collaboration with the department of Electrical Engineering at the Technical University of Denmark, DTU. It is presented in fulfilment of the requirements for acquiring an M.Sc. in Engineering Acoustics. This thesis was prepared in the period from February 2013 to June 2013, under the supervision of Professor Lars Kai Hansen and Postdoc Bjørn Sand Jensen.

This thesis deals with the extraction of structure and information from speech containing multiple speakers. This is done through the extensive use and development of various machine learning methods, as well as the modelling of audio specific features in the cepstral domain.

This thesis is funded by CoSound – A Cognitive Systems Approach to Enriched and Actionable Information from Audio Streams [20].

Lyngby, 15-June-2013

[illegible]

Thor Bundgaard Nielsen

Acknowledgements

I would like to thank my supervisor Lars Kai Hansen, whose great course on Non-Linear Signal Processing inspired this thesis, for the insightful discussions along the way and for helping me through the mathematical deductions. My teacher Morten Mørup for introducing me to the topic of machine learning which very nearly escaped my notice. CoSound for sponsoring my ticket to Digital Audio – Challenges and Possibilities, event in Copenhagen on June 21 2013. My co-supervisor Bjørn Sand Jensen, particularly for helping me access the IMM cluster. The Acoustic Technology group for their support. Finally, for the extensive work of proofreading the thesis, I would like to thank my brother Emil and my father Ove for a diligent effort. Thanks to my family, this thesis has become much more pleasant to read.

Abstract

The problem of unsupervised retrospective speaker change detection continues to be a challenging research problem with significant impacts on automatic speech recognition and spoken document retrieval performance. The aim here is to design a much faster than real-time speaker diarisation software suite possibly for use in news audio editing. This thesis aims broadly, comparing a variety of well-known speaker segmentation methods based around vector quantization and Gaussian processes. These well established methods are compared to a novel statistical change-point detection algorithm based on non-parametric divergence estimation in the field of relative density-ratio estimation using importance fitting. All methods are optimized using a direct search method, initialized by a custom multi-step grid-search, in a recursive speaker change detection paradigm, built on Mel-Frequency Cepstral Coefficients. Methods are compared on the basis of their performance and their efficiency on the ELSDSR speech data corpus. It is found that an inexpensive Gaussian process based on the Kullback-Leibler distance when optimized in this recursive SCD paradigm, can compete in terms of performance with far more expensive methods while maintaining a very high efficiency. Further a recursive speaker change detection paradigm yields promising results. Beyond this, it is shown that a simple feature selection based on a theoretical model of the human speech production system yields a marked improvement in performance. Lastly this method is experimentally applied in the field of agglomerative hierarchical speaker clustering and compared to a more well established method based on the Baysian Information Criteria. Here a novel approach similar to the Kullback-Leibler distance called the Information change rate shows promising results. The system developed in this thesis could be implemented in digital audio workstations to greatly simplify the process of speaker segmentation by automatically answering the question of "Who spoke when?".

Contents

Nomenclature

Symbols

Greek

α_{cd}	M_{cd} gain
α_F	F-measure weight
α_{FAC}	M_{FAC} gain
α_R	Mixture-density
Γ	Gamma function
$\delta(x)$	Error term
$\theta(n)$	Impulse response
$\Theta(k)$	Spectral impulse response
λ	Regularization parameter (RuLSIF)
μ	Population mean
μ_k	Centroid position of cluster k
σ	Width of kernel $K(x^A; x^B)$
Σ	Population covariance matrix
υ	D.o.F

Roman

A_n	Preceding analysis window at time t_n
B_n	Succeeding analysis window at time t_n
$c_\theta(k)$	Cepstral impulse response
$c_e(n)$	Cepstral excitation sequence
c_k $(k = 1, 2, \ldots, K)$	k^{th} code-vector (cluster)
$c_s(n)$	Cepstral speech signal
$C_\theta(k)$	Spectral magnitude of impulse response
C^A	Codebook for A
C^B	Codebook for B
$C_e(k)$	Spectral magnitude of excitation sequence
C_n	The n^{th} MFCC
D	Number of dimensions
$e(n)$	Excitation sequence
$E(k)$	Spectral excitation sequence
f	Frequency (independent variable of spectrum)
f_{mel}	Mel-frequency scale independent variable
$f(y)$	Taylor expansion
fn	False negatives
fp	False positives
F	F-measure
F_{CDF}	Student's t CDF
$g(x\|\theta)$ $(\theta = \theta_1, \theta_2, \ldots, \theta_N)^T$	Density-ratio with parameters θ
H_0	Null hypothesis
K	K-means codebook size
$K(x^A; x^B)$	Gaussian kernel
K_{mel}	Mel filter bank size
$KL_{\text{divergence}}$	Kullback-Leibler divergence
KL_{distance} and KL	Kullback-Leibler distance

l_{aw}	Analysis window length
l_s	Analysis window Shift length
M_{cd}	Moving average threshold of change-detection algorithm
M_{FAC}	Moving average threshold of FAC algorithm
N	Sample size
$\mathcal{N}$	Gaussian (normal) distribution
p	Probability
$p(x)$	Example PDF
P	Independent parameters needed for full description
$P_A(x)$	A population PDF
$P_B(x)$	B population PDF
PRC	Precision
$q(x)$	Example PDF
q_{jk}	Entries of Q
Q	Sample covariance matrix (estimate of Σ)
r_α	Alpha-relative density-ratio
RCL	Recall
R_{len}	A random length drawn from a uniform distribution between a fixed lower and upper boundary.
R_{sample}	A random speech sample drawn from the pool of samples longer than R_{len}
r_{nk}	Vector signifying to which cluster x_n belongs
s	Estimate of population standard deviation
$s(n)$	Speech signal
$S(k)$	Spectral speech signal
S_k $(k = 1, 2, \ldots, K_{\mathrm{mel}})$	Outputs of the Mel filter bank
$\pm t_{95\%}$	The t-scores at the edges of the confidence interval
tn	True negatives
t_n	n^{th} potential change-point's temporal position
tp	True positives
t-score	The ratio of the departure of an estimated parameter from its notional value and its standard error
T_i	Change-point width, another change-point must be at least $\pm\frac{T_i}{2}$ away
$T_{\max}$	Defines the largest amount of data in all algorithms

$X = \{x_1, x_2, \ldots, x_N\}$	Feature vector sequence
$\overline{X}$	Sample mean (estimate of μ)
Z	Standard score of a raw score X

Glossary

AGGM	Adapted Gaussian Mixture Model
AHC	Agglomerative Hierarchical Clustering
AUC	Area Under the Curve
BIC	Bayesian Information Criterion
CDF	Cumulative Distribution Function
CMFAC	Combined Metric False Alarm Compensation
CMS	Cepstral Mean Subtraction
CPU	Central Processing Unit
DAW	Digital Audio Workstation
DCT	Discreet Cosine Transform
DFT	Discrete Fourier Transform
D.o.F	Degrees of Freedom
DTU	Technical University of Denmark
ELSDSR	English Language Speech Database for Speaker Recognition
EM-algorithm	Expectation-Maximization algorithm
FAC	False Alarm Compensation
FFT	Fast Fourier Transform
FSCL	Frequency-Sensitive Competitive Learning
GMM	Gaussian Mixture Model
HMM	Hidden Markov Model
ICA	Independent Component Analysis
ICR	Information Change Rate
IDFT	Inverse Discrete Fourier Transform
i.i.d	independent and identically distributed

IMM	Department of Mathematical Modelling (at DTU)
ISP	Intelligent Sound Processing toolbox
JIT	Just-In Time (accelerator)
KL	Kullback-Leibler distance
KL (FULL)	Kullback-Leibler distance using full Mel-range
KL (lower)	Kullback-Leibler distance using lower Mel-range
KL (upper)	Kullback-Leibler distance using upper Mel-range
KLEIP	Kullback-Leibler Importance Estimation Procedure
K-means	K-means clustering
LBG	Linde-Buzo-Gray algorithm
LOF	Local Outlier Factor
LSPs	Line Spectrum Pairs
LTI	Linear Time-Invariant (LTI-system)
MATLAB	MATrix LABoratory
Mel	The Mel scale
MFC	Mel-Frequency Cepstrum
MFCC	Mel-Frequency Cepstral Coefficient
MLLR	Maximum Likelihood Linear Regression
PCA	Principal Component Analysis
PDF	Probability Density Function
PE	Pearson divergence
PLP	Perceptual Linear Prediction
PRC	Precision (a constituent of the F-measure)
Q.E.D.	Quod Erat Demonstrandum "Which had to be demonstrated"
RCL	Recall (a constituent of the F-measure)
RFAC	Recursive False Alarm Compensation
ROC	Receiver Operating Characteristic (curve)
RuLSIF	Relative unconstrained Least-Squares Importance Fitting
SEM	Standard Error of the Mean
SCD	Speaker Change Detection
SDR	Spoken Document Retrieval
SNR	Signal-to-Noise Ratio
SOM	Self Organizing Maps
STE	Short-Time Energy
STFT	Short-Time Fourier Transform

TIMIT	Acoustic-Phonetic Continuous Speech Corpus
uLSIF	unconstrained Least-Squares Importance Fitting
VQ	Vector Quantization
VQD	Vector Quantization Distortion
WCSS	Within-Cluster Sum of Squares
WTA	Winner-Takes-All (in the context of K-means)
ZCR	Zero-Crossing Rate

Chapter 1

Introduction

The internet has become a vast resource for news pod-casts and other media containing primarily speech. This poses an interesting problem as traditional text-based search engines will only locate such content through information tagged onto the audio-files manually. Manual labelling of audio content is an extensive task and therefore necessitates automation. This in turn created the whole field of Speaker Change Detection, SCD, or speaker diarisation, involving methods from machine learning and pattern recognition.

This thesis will explore a range of available methods for SCD and compare them for use in audio editing. Audio editing involves a rather tedious process of familiarisation with the individual segments of the media content. The hope is that the aforementioned methods can ease and simplify this process, thus empowering Digital Audio Workstations, DAWs, by adding automated speaker diarisation.

Optimally a DAW using speaker diarisation would be able to search inside audio files for high level information, here referring to topics, speakers, environments, etc. This thesis will however focus primarily on SCD, as it builds on the knowledge gathered from the creation of Castsearch [87], a context based Spoken Document Retrieval, SDR, search engine. During the creation of Castsearch, Jørgensen et al. designed a system for audio classification [55]. This classification system includes the classes; Speech, music, noise and silence. This classification system was applied to allow raw data to be processed, the use of such a system in this thesis is discussed in section 2.1. To clarify this thesis will focus solely on audio content. In other words incorporation of audio meta-data, audio transcriptions, associated video content, etc., is outside the scope of this thesis.

1.1 Speaker Change Detection

SCD is the process of locating the speaker to speaker changes in an audio stream. This section will delineate the methods found in the field of SCD and describe their general application is this thesis. This section will also briefly touch on speaker clustering, a process which SCD enables.

The end goal is to hypothesize a set of speaker change-points by comparing samples before and after a potential change-point at regular intervals, see figure 1.1.

1.1.1 Real-time detection vs. retrospective detection

The process of detecting abrupt changes in an audio stream must be divided into two distinct sub-fields with disparate challenges and trade-offs involved. As the title suggests these sub-fields revolve around the proximity to real-time detection. As will be mentioned below the metric-based methods employed in this thesis require a certain amount of data after a potential change point in order to detect it. In addition to the requirements on data, there are the requirements on processing time. The aim of this thesis is to design a system that takes recordings, thus not real-time, and process these. This processing must however be comparable or preferably much quicker than real-time in order to retain its usefulness. The work here will therefore use optimised retrospective detection.

1.1.2 Supervised vs. unsupervised methods

Another way to bisect the field of SCD is a division into supervised and unsupervised methods. If the number of speakers and identities are known in advance, supervised models for each speaker can be trained, and the audio stream can

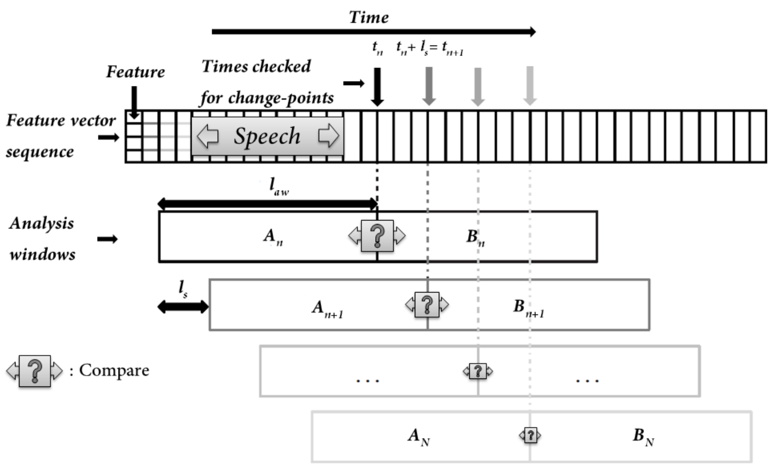

Figure 1.1: This figure is presented as a rough reference of the basic concept: Which is to compare segments of data before and after a potential change-point and judge whether it is different enough. The data, consist of a sequences of feature vectors describing the sound over a small interval. As seen the figure uses a variety of parameters, which will be described throughout this thesis as they become relevant, these include the analysis windows A and B, of length l_{aw}, as they are shifted forward in time. This occurs by regular increments of l_s, to the next potential change-point at time t_n. The reader is encouraged to review this figure at regular intervals. It should be noted that the proportions portrayed in this figure are greatly exaggerated.

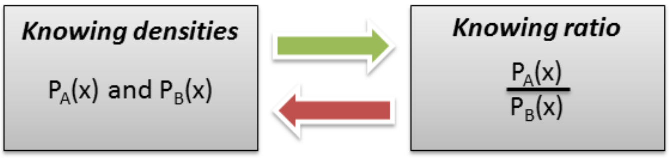

Figure 1.2: Rationale of direct density-ratio estimation. As seen a shortcut to the standard approach is taken. Rather than model the prior and the posterior data individually only to subsequently estimate the ratio, a more direct method is to directly model the ratio, see secion 3.2.3. Figure concept borrowed from [75].

be classified accordingly. If the identities of the speakers are not known in advance unsupervised methods must be employed. Due to the nature of the data; unsupervised SCD is a premise of this thesis and its approaches can roughly be divided into three classes, namely energy-based, metric-based and direct density-ratio estimation:

Energy-based methods rely on thresholds in the audio signal energy. Changes are found at silence-periods. In broadcast news the audio production can be quite aggressive, with only little if any silence between speakers, which makes this approach less attractive. Whereas metric-based methods basically model the data before and after a potential change point and subsequently measures the difference between these consecutive frames that are shifted along the audio signal.

Despite the established results of these metric based methods, they have a disadvantage; they try to estimate a distinct Probability Density Functions, PDFs, before and after a potential change point, rather than directly estimating the difference between these. Since this difference contains all required information, the intermediate step of gathering information only to discard it later is circumvented, see 1.2. This group of methods is called direct density-ratio estimation and is a fairly new idea in the field of SCD; with the method, RuLSIF [75], applied here designed some months ago, see section 3.2.3.

1.1.3 Precision in time vs. false positive rate

The process of SCD has an inbuilt trade-off that needs to be addressed. In order to detect short speaker segments it needs to be fairly constrained in time. This however leads to a smaller dataset per possible change point and naturally causes a higher amount of false positives. Turning SCD into an iterative process can mediate this trade-off between the ability to notice short segments vs. a higher false positive rate. This method has previously been called false alarm compensation, is applied in this thesis and is described in section 2.4.

1.1.4 Speaker change detection methods

Speaker change detection methods used in this project can be further grouped into three subgroups:

1. Gaussian Processes
2. Vector quantization
3. Direct density-ratio estimation

The first subgroup are the distance measures between separate multivariate Gaussians trained on data before and after the potential change-point. These include the Kullback-Leibler Distance, here termed $KL_{distance}$ or simply KL, and a simplification of it, the so-called Divergence Shape Distance, DSD, which focuses solely on locating covariance changes. For more details see section 3.2.2.

The second subgroup is the Vector Quantization, VQ, approach which incorporates a variety of approaches to 'discover' the underlying structure of a dataset through iteratively improved guesses. These guesses are in the form of a much smaller amount of representative data, the difference is then measured in the total movement of this representative data, called Vector Quantization Distortion, VQD. For more details see section 3.2.1.

Lastly, for the Direct density-ratio estimation a variant of Kullback-Leibler Importance Estimation Procedure, KLEIP, called Relative unconstrained Least-Squares Importance Fitting, RuLSIF, introduced by Liu et al. [75] is applied. The concept behind this method is slightly more abstract, but revolves around modelling the distortion of the prior data required to produce the posterior data and then condensing this distortion model into a single number. KLIEP is designed to be coordinate transformation invariant, this however has the disadvantage of an increased sensitivity to outliers. For this reason the variant, RuLSIF, is preferable in practical use. For more details see section 3.2.3.

All methods are metric-based; in essence this means that they supply a number for every point in time. The magnitude of this number is correlated with the likelihood of a change-point at that moment. These metrics therefore need to be thresholded to yield definite predictions, rather than a smooth scale of possibilities. These thresholds will be defined relative to a smoothed version of the metric itself, see figure 1.1. For more details see section 1.1 and 2.4

1.1.5 Overlapping speech

Since real dialogue does not always conform to the simple model of speaker turns the possibility of overlapping speech segments is a liability and the definition for 'babble noise' is vague in this sense. Overlapping speech naturally spreads a speaker change over time, this may even blur the speaker change to insignificance and a smooth transition to a different speaker altogether is a real liability. The methods discussed in section 1.1.3 can alleviate this issue assuming the notion of speaker turns remains valid. This issue naturally lowers the precision of the model, in a sense this issue will be regarded as a single speaker in speech noise. The data used in this thesis does not contain overlapping speech; therefore its imagined consequences are purely theoretical.

1.1.6 Speaker segment clustering

Once a dialogue has been separated into speaker turn segments, these segments can be clustered. This process will produce a reasonable guess as to the amount of speakers present in the dialogue and a notion of “who said what when”. The amount of background noise and other such limitations may impact the performance of this step.

In this thesis several approaches to speaker clustering within the field of Algomerative Hierarchical Clustering [8], AHC, have been compared; the general concept is to start by assuming that every speaker turn is a unique person. The algorithm then iteratively combines the most similar segment until only 2 segments remain. The correct amount of speakers is then found by looking for the combination where the constituents were the most dissimilar.

AHC naturally requires a metric by which to judge the dissimilarity between speakers. Here the tested metrics include all the metric described in section 3.2 along with a Bayesian Information Criterion, BIC, approach and a further improvement on this termed Information Change Rate, ICR, [42].

Unfortunately speaker clustering was started late in the thesis, as SCD naturally is a prior step. The scarcity of available computing resources at that point was enhanced by unforeseen circumstances, see section 4.3.1. This necessarily deprioritised a rigorous approach to speaker clustering. The relevant software was written, see appendix A.1.2, but were only lightly experimented with in the further work section 5.

This step could have an important role to play as information gathered here could have hinted at which segments contain missed change points and might have facilitated the possibility of the software storing a profile for a particular speaker for later recognition.

1.2 Toolboxes and other software packages

All software is developed using MATLAB and its accompanying toolboxes. In addition to this, custom toolboxes were employed including; the Intelligent Sound Processing, ISP, Toolbox, developed as part of the Intelligent Sound project by Jensen et al. [53] and Mike Brookes' VOICEBOX toolbox [13]. It should be mentioned that the ISP toolbox has been adapted to support a 64 bit windows based OS, along with a number of technical improvements, whenever unsupported features were required, see section 4.3.1.

1.3 System overview

This section presents a basic modular design of the proposed system given as a flow chart in figure 1.3. The process starts with a raw audio sample containing speech, with multiple speakers. This raw data is far too redundant and messy to reliably determine speaker changes. The raw data is therefore fed into a preprocessing mechanism that generates the raw audio feature, the MFCCs, see section 2.2.

The system subsequently blindly segments these MFCCs into 3 second segment, with a new segment starting every 0.1 seconds, marking a position to be checked for a possible change-point. This results in an organised data structure, see figure 1.1.

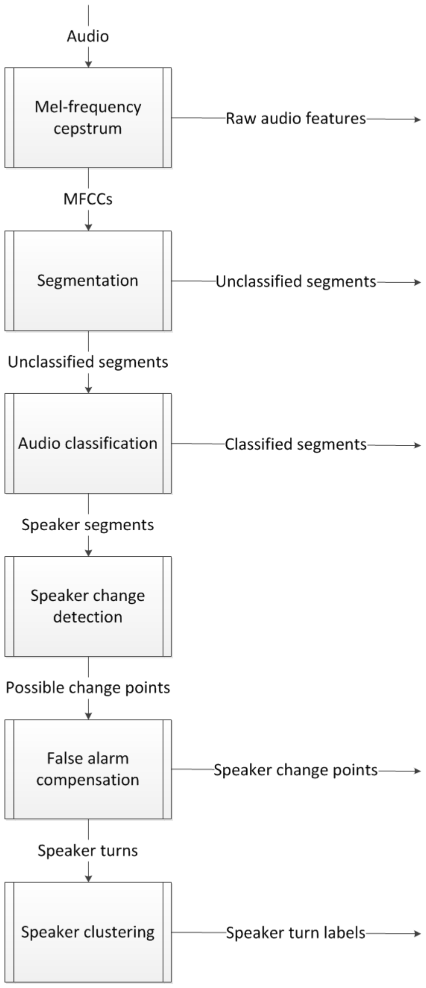

Figure 1.3: Simplified overview of the system developed in this thesis presented as a one-way flow chart, where arrows mark the outputs and inputs of the various modules. Section 1.3 is dedicated to describing this figure in detail.

Prior to checking the possible speaker change points, it is verified that all these segments contain primarily speech. Segments containing music, silence or classified as other are discarded.

All segments containing primarily speech are then fed into the speaker change detection module. This module, using only the 3 second segment before and after a possible change-point quickly sorts through the vast majority change-points.

The remaining possible change-points are then scrutinised using as much data as possible by the false alarm compensation module, which attempts to identify and remove the remaining false positives. This process concludes with a set of hypothesized speaker change points and the corresponding speaker turns.

The speaker turns are then handed to an unsupervised clustering module which assigns a label to each speaker and subsequently marks every speaker turns with the label of its speaker, using an agglomerative hierarchical clustering approach.

Chapter 2

Data pre-processing

This chapter will begin with a description of the available data corpora, why ELSDSR was selected in lieu of more common choices, and why raw news podcast data was not applied. Given this information the methods involved in applying the selected data will be explained, and a deeper analysis is conducted into potential weak links of the process.

The chapter will then proceed to an exhaustive search of the commonly used feature extraction techniques applied in SCD and related fields, speaker recognition, speaker diarisation, etc. The process concludes with the selection of Mel-Frequency Cepstral Coefficients, MFCCs, as the sole features used in this thesis.

This is followed by a thorough description of the theory, the methods, the reasons and the attributes of the MFCCs. This process concludes with a range of possible feature sets; these feature sets are compared in section 4.1.

Finally, this chapter will conclude with a description of the methodologies and practices applied to detect speaker changes and to reject false speaker changes. This final part is accompanied by the theory and concepts behind a novel hybrid approach based on combining unrelated SCD methodologies.

2.1 Data

A dataset of speech changes is required in order to train, test and compare the speaker change detection and speaker clustering methods. In addition to this a data set is needed to evaluate hyper parameters and finally a dataset is needed to evaluate the performance of the full system.

As the present work builds the preliminary work for CastSearch, by Mølgaard et al. [87], and is intended for use in news editing; it seems natural to acquire the CNN (Cable News Network) pod-cast dataset. This dataset presently consists of 1913 pod-cast, totalling about 6.6GB. Due to the size alone the dataset contains more than enough speaker variety and speaker changes.

The use of the CNN data does however pose some problems. Firstly since it is merely recordings of actual news shows it contains a mixture of speech, music, silence and other. All non-speech section would have to be filtered out, since this project is focused on detecting changes from one speaker to another, not from speaker to music, etc.

However even with the data preprocessed to only include speech a significant problem remains, namely the quality of the different speech segments. News anchors usually have expensive equipment and are situated in studios. Whereas with reporters in the field, background noise, bandwidth and bit-rates play a large role in the quality of the recording.

These are powerful cues as they almost always signify actual speaker changes. They are however not cues specific to the speaker and are therefore not actual cues at all, but merely a potentially powerful masking of cues correlated with actual cues. In addition they represent a loss of information and can therefore not be corrected for or filtered away.

And finally the CNN data is not annotated, requiring a tedious process of manual labelling, not always possible since the assumption of individual speaker turns does not always fully apply. All in all using actual raw data may hinder rather than serve the purpose of training a speaker change detection model. As such the only alternative is to design synthetic data that comes as close to the real data as possible, without the inherent problems mentioned above.

2.1.1 Synthetic data

Since the use of raw data for model training is not viable, the use of synthetic data is necessary. Several options, corpora, are readily available for this purpose. They are however generally accompanied by a rather large price tag. The department DTU Compute has two usable corpora on hand, the ELSDSR [31] and the TIMIT [37] corpora.

Since the ELSDSR corpus contains a sufficient amount of data and has long uninterrupted speaker segment, on the order of 20 sec, it is almost ideal for the purpose of this project. A few minor hassles have to be overcome though, these include how to string several segments together, how to handle the bias inherent in the distribution of speaker segment lengths and finally how to sample the data.

The TIMIT corpus could be applied, but suffers from all the issues of ELSDSR, but also has fairly short uninterrupted speech segments. It could be argued that using TIMIT, which is well recognised and widely applied, would enable a more direct comparison to other work. It could also be argued that the wider range of dialects available in TIMIT would slightly increase generalisability. However, these factors a considered minor compared to the advantage of longer uninterrupted segments that ELSDSR fields. Therefore as ELSDSR has more than sufficient data TIMIT will not be applied.

2.1.2 ELSDSR speech corpus

ELSDSR [31] is an English Language Speech Database designed for Speaker Recognition [30].

ELSDSR contains voice recordings from 23 speakers (13M/10F), age ranging from 24 to 63. The spoken language is English, all except one speaker have English as a second language.

The corpus is divided into a training and a test set. Part of the test set, which is suggested as training subdivision, was made with the attempt to capture all the possible pronunciation of English language including the vowels, consonants and diphthongs, etc. Seven paragraphs of text were constructed and collected, which contains 11 sentences. The training text is the same for every speaker in the database. As for the suggested test subdivision, forty-four sentences (two sentences for each speaker) were collected.

In summary, for the training set, 161 (7 Utterances $*$ 23 People) utterances were recorded; and for test set, 46 (2 Utterances$*$23 People) utterances were provided.

2.1.3 Splicing speech samples

The creation of synthetic data has an inherent problem associated, as these corpora do not contain speaker changes, they contain speech samples. These speech samples need to be spliced together and this is where the problem enters; how to distinguish between locating the spike, the sudden change in sound pressure, that the splicing creates and locating an actual speaker change.

2.1.3.1 Method

It was surprisingly difficult to find any work that mentioned this issue, let alone proposed methods to solve it. Even presenting the issue to a significant proportion of the departments DTU Compute and DTU Acoustic Technology at separate status presentations, failed to yield reference-able research into the issue. It is therefore necessary to invent a method. As this issue it probably minor, a relatively simple method is proposed in order to avoid creating unnecessary artefacts.

Authors note: It should be mentioned that upon revision; a patent issued in 1988 to Neil R. Davis [22] was located by the thesis supervisor, methods described in this patent did not make it into this thesis.

Several simple solutions made it onto the drawing board, revolving around three groups of methods:

1. Searching in the vicinity for suitably similar features of the signals, discarding data around the edges of the signal.
2. Warping the signals in the vicinity in order to reduce the broadband noise that a spike would create.
3. Overlapping the signals, thus smearing the speaker change over time.

The fact that the data is purely speech with almost no background noise means that the signal regularly crosses zero and has smooth derivatives. The method applied is a variant of method 1 and locates the nearest zero crossing with

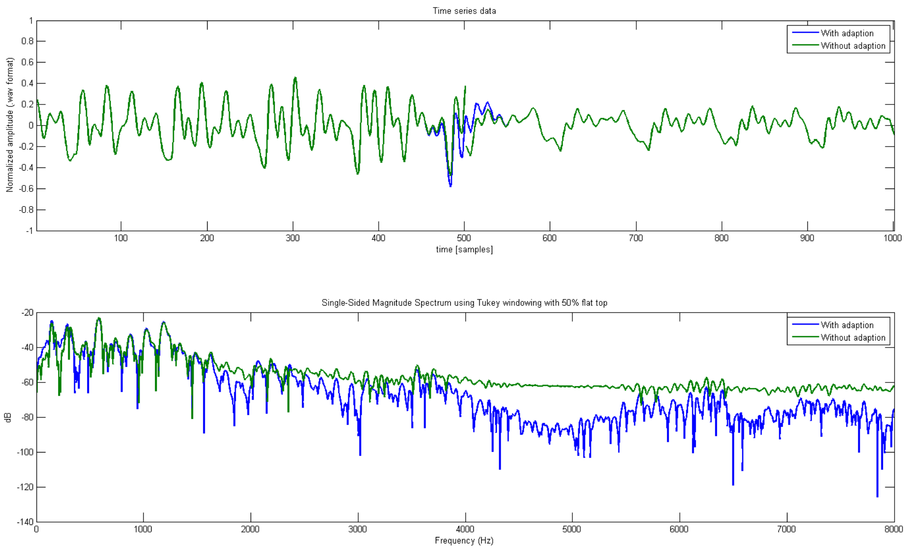

Figure 2.1: Visual illustration of the splicing issue when combining two audio files. In the graph two speech samples are directly combined and combined using a splicing method. In the lower graph the corresponding effects in the spectral domain are observed. As seen the splicing method used here substantially reduces the broadband noise that the sharp transition creates. In this particular example a version of the 2. group of methods is applied, though as mentioned in section 2.1.3.1 this is not the version that is used in this thesis. This is provided merely for visual reference. This splicing method simply pulls the ends together, which has the disadvantage of a number of free parameters. These free parameters control the locality of the splicing, in this case it converges exponentially towards the junction. See appendix A.5.2.

identical sign on the first derivative and merely discards the data in between. This slightly reduces the length of the speech segments, but this effect is on the order of milliseconds and is thus negligible.

As the MFCC calculation uses temporal windows with 50% overlap, only 3 feature vectors will be affected by the switch. See section 2.2 for details and See figure 2.2 for a graphical representation of the data using principal component analysis on a random change-point.

If these vectors are highly affected this could mean that up to 0.5% of the data could be outliers, since the analysis window on each side is 3 seconds long and each MFCC has a width of 20ms with 50% overlap. This is of cause assuming only 1 speaker change is inside the analysis windows, however with more than one speaker change this issue would be minor in comparison. Since some of the applied methods, KL and DSD, model the subsequent analysis windows with normal distributions, which are very sensitive to outliers [51, 117], these potential outliers could amplify or even dominate the difference between the analysis windows.

2.1.3.2 Analysis

Figure 2.3 displays a histogram of the local outlier factor, LOF, [12] on the border region compared to a histogram of the LOF of all other data within the analysis windows. This data is gathered from 200 change-points.

The LOF score is basically a local density estimation where the density is modelled using the Euclidean distance to the K^{th} nearest neighbour, in this case K is set to 3 since the outliers might be clustered together in which case they will be each other's 1^{st} and 2^{nd} neighbours.

From figure 2.3 two things are apparent; the LOF scores are fairly normally distributed and the region is definitely showing anomalous behaviour with a mean significantly different from the rest of the data. However, whether this is an inherent part of the data, a result of the method or whether this method is merely completely ineffectual in dampening the splicing effect is unknown and will require deeper analysis.

Figure 2.4 displays a similar representation of the data, this time without any modification to the border region between the speech samples, here a very similar result is seen. This seems to indicate that the method for splicing the data together is ineffectual or being masked by an actual trend in the data.

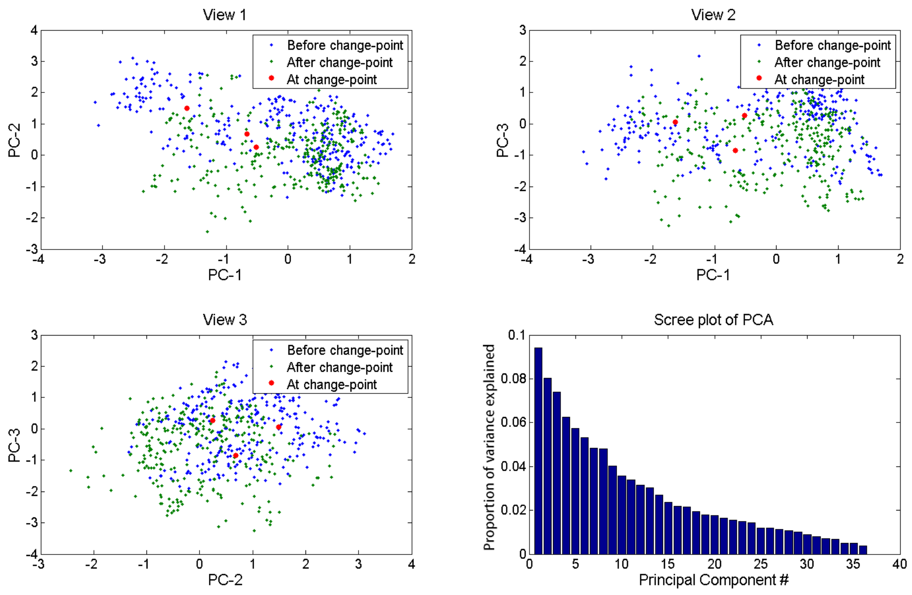

Figure 2.2: MFCCs visualised through a Principal Component Analysis, PCA, of data from 2 random subsequent analysis windows. The region potentially affected by the temporal effects of splicing the 2 audio files together is highlighted in red. In this example the audio files are joined without splicing. As is evident the MFCCs at the change-point are not obvious outliers. As is seen from the scree plot and as mentioned in section 2.2.4 the data is quite globular, even with this small subset of the data the first 3 principal components only account for 25% of the total variance. Meaning that the MFCCs might still clearly be outliers, but due to the dimensionality; the human visual system is inadequately capable of receiving the data efficiently.

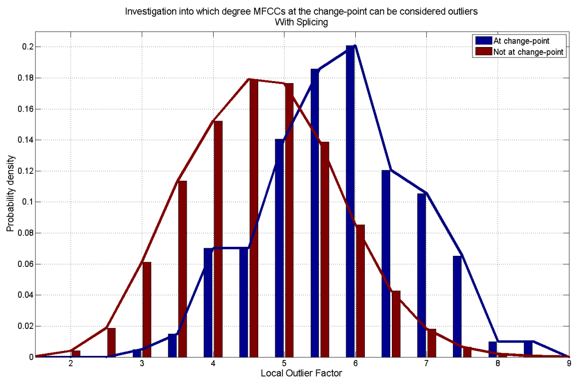

Figure 2.3: Investigation into which degree the MFCCs at the change-point can be considered outliers, in this case with the novel splicing method applied. As seen the distributions are clearly almost Gaussian and are clearly distinct. The x-axis represents the LOF score, a measure of the density around an MFCC. This density is measured in Euclidean distance.

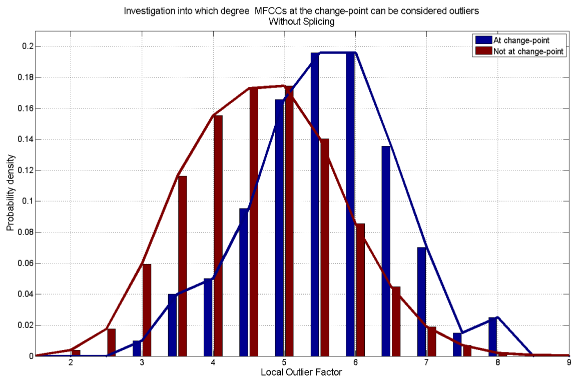

Figure 2.4: Investigation into which degree the MFCCs at the change-point can be considered outliers, in this case without a splicing method applied, i.e. the audio files were simply joined. As seen the distributions are clearly almost Gaussian and are clearly distinct. The x-axis represents the LOF score, a measure of the density around an MFCC. This density is measured in Euclidean distance.

2.1.3.3 Effectiveness

The hypothesis is that the splicing of sound files will cause the MFCCs at - and possibly next to - the change-point to exhibit outlier behaviour. Since the MFCC at the change-point exhibits outlier behaviour even when the sharp change of sound pressure is removed, the effectiveness of the applied method must be in how distinct the outlier behaviour is, in comparison to how distinct the outlier behaviour is without any splicing method applied. As seen in table 2.1 the distributions seen in figures 2.3 and 2.4 yield very high t-scores and as such are definitely outliers, whether one of them is less distinct will require further analysis.

2.1.3.3.1 Statistical hypothesis test

A way to quantify the degree to which these MFCCs exhibit outlier behaviour is through the use of the Welch's t-test [120], otherwise known as the unequal sample sizes, unequal variances and independent two-sample t-test:

$$t\text{-}score = \frac{\overline{X}_1 - \overline{X}_2}{s_{\overline{X}_1 - \overline{X}_2}} \tag{2.1}$$

Where X_i is the sample mean of the i^{th} sample and where

$$s_{\overline{X}_1 - \overline{X}_2} = \sqrt{\frac{s_1^2}{N_1} + \frac{s_2^2}{N_2}} \tag{2.2}$$

Where s_i^2 is the unbiased [27] estimator of the variance and N_i is the sample size. Unlike in Student's t-test [27], the denominator is not based on a pooled variance estimate.

When performing statistical hypothesis testing, the first step is to determine the null hypothesis, H_0. In this case it is that the MFCC at the change-point is not an outlier, that is, that the mean LOF score is equal to the mean LOF score of all the data:

$$H_0 : \mu_1 - \mu_2 = 0 \tag{2.3}$$

The hypothesis is identical in the no splicing method test.

Run #	1	2	3	4	5	6	7	8	9	10	Mean	Standard Error
T-Score (With splicing)	27.7	28.4	30.2	29.6	29.5	28.7	30.7	28.6	29.2	31.3	29.4	1.1
T-Score (Without splicing)	30.0	29.0	26.6	30.2	27.5	28.6	28.6	30.8	29.3	26.7	28.7125	1.4313

Table 2.1: Results from 10 runs of Welch's t-test between a sample of MFCCs LOF scores directly at its change-point and the LOFs of a collection of every other MFCC, except MFCCs adjacent or on change-points. The results clearly showing very high t-scores irrespective of whether a splicing method was applied. In addition it is seen that the mean t-scores are also very similar and overlap each others standard error, indicating that with or without splicing results in roughly the same outlier degree. A separate t-test is necessary to determine within which certainty the 2 trials run sets can be said to differ.

The procedure used is to first draw 1000 change-points, as described in section 2.1.5, except that the analysis windows are kept clean of any other change-points. Only the MFCC directly at the change-point is used, excluding the adjacent ones, in order to increase the signal-to-noise ratio.

Then to run the t-test between all the data and the data at the change-points, repeat this procedure 10 times to get a better estimate of the t-score and to estimate of the standard error [27] of the estimated t-score. See table 2.1 for the result with and without the splicing method.

And finally run a new t-test similar to the one performed in the previous step, but between the results with and without the splicing method. This yielded the result:

$$t\text{-}score = 1.1949 \tag{2.4}$$

This t-score is quite low and will require interpretation in order to determine at which significance level the null hypothesis of the proposed method making no difference can be rejected.

2.1.3.3.2 Interpretation

The difference is statistically significant at a specific confidence level if the t-score is outside the corresponding confidence interval about the hypothesized value of zero. If on the other hand the t-score is within the confidence interval the null hypothesis cannot be rejected at that confidence level and the difference could just be statistical variation. A common choice is a confidence level of 95%, meaning that the result has a one in twenty chance of being wrong, assuming the t-distribution is a good approximation.

To determine the confidence interval the degrees of freedom, D.o.F, must be estimated. For this purpose the Welch-Satterthwaite equation [103] is employed, in this case:

$$\upsilon = (N_1 - 1) + (N_2 - 1) \tag{2.5}$$

Where υ the D.o.F is simply the amount of observations minus one for each of the estimated means.

The t-score that corresponds to the edge of the confidence interval, is calculated by computing the inverse of Student's t Cumulative Distribution Function, CDF, F_{CDF}. The t inverse function in terms of the t CDF is [27]:

$$x = F_{\text{CDF}}^{-1}(p|\upsilon) = \{x : F_{\text{CDF}}(x|\upsilon) = p\} \tag{2.6}$$

Where [27]:

$$p = F_{\text{CDF}}(x|\upsilon) = \int_{-\infty}^{x} \frac{\Gamma(\frac{\upsilon+1}{2})}{\Gamma(\frac{\upsilon}{2})} \frac{1}{\sqrt{\upsilon\pi}} \frac{1}{(1+\frac{t^2}{\upsilon})^{\frac{\upsilon+1}{2}}} \, \mathrm{d}x \tag{2.7}$$

Where Γ is the gamma function [27] and the result, x, is the solution of the CDF integral given the D.o.F, υ, and the desired probability p. The result was calculated using the MATLAB function **tinv**.

The t-scores are expected to fall within:

$$\pm t_{95\%} = 2.1009 \tag{2.8}$$

It is therefore concluded that at this confidence level the difference might be due to chance and the null hypothesis cannot be rejected. Simply put more data is required, the applied method has no statistically significant impact on the outlier score at the change-point, given that:

$$t\text{-}score = 1.1949 < \pm t_{95\%} = 2.1009 \tag{2.9}$$

Using equation 2.6, the confidence level at which the null hypothesis can be rejected is:

$$\text{Null hypothesis rejected at a confidence level of } 75.24\% \tag{2.10}$$

The results from the final t-test on the t-scores gathered using the method compared to not using the method are quite interesting. It would appear that the data shows, to a confidence level of 75%, that applying the splicing method actually increases the outlier score of the MFCC at the change-point, since

$$\text{T-Score (With splicing)} \geq \text{T-Score (Without splicing)} \tag{2.11}$$

,seen in table 2.1.

If anything this suggests that the method should not be used. However since the method renders the click noise from the transition inaudible and the null hypothesis can only be rejected at a confidence level of 75%, the method must have some effect and is therefore applied in lieu of a better methodology.

2.1.4 Speech sample sizes

Since the dataset only contains a small amount of speech samples, 161 training and 46 test, the variation of speech sample lengths in also small, see figure 2.5. To ensure that the methods do not make use of this fact in some fashion, and to ensure that the methods are calibrated to take short segments into account, a process is applied to ensure a uniform distribution of speech segment lengths. This is achieved by swapping the uniform distribution of speakers with the non-uniform distribution of lengths as follows.

1. A random length, R_{len}, is drawn from a uniform distribution between a fixed lower and upper boundary.

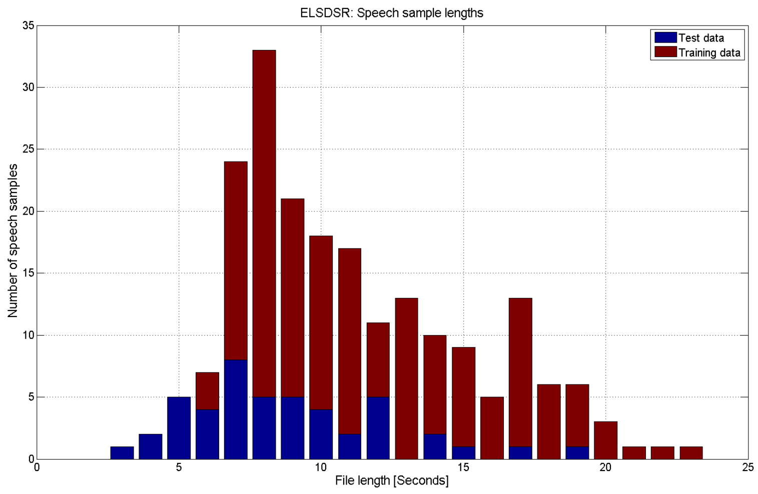

Figure 2.5: Distribution of speech samples lengths in the ELSDSR database. As seen the distribution is different for test and training data, far from uniform and samples of sizes smaller than 3 seconds, while common in news pod-casts are entirely missing. This clear bias could skew the results if not removed. Section 2.1.4 describes how the speech sample lengths are randomised to a uniform distribution.

2. A random speech sample, R_{sample}, is drawn from the pool of samples longer than R_{len}.

3. A random subsection with the length of R_{len} is drawn from R_{sample}.

Through this process a wide variety of speaker lengths are achieved, and by extension a huge number of speaker changes are possible. This process does however favour the middle part of the longer speech samples. Since the dynamics of speech, with respect to MFCCs, is constant independent of where in the sample it is drawn, this should not bias the data. The lower boundary of how short a speech segment can be is set to 1 second, as the change-detection algorithm by design filters away segments shorter than 1 second, see section 2.3, whereas the upper boundary of how long a sample can be is arbitrarily set to 15 seconds to ensure that the pool of samples longer than R_{len} has some speaker variety.

2.1.5 Data bootstrap aggregation

Bootstrap aggregation, also known as 'bagging' [113], is a technique that repeatedly samples - with replacement - from a data set using a uniform distribution. ELSDSR contains 161 files in the training set and another 46 in the test set. Looking at the training data alone this yields the possibility of $161 * 160 = 25760$ different speaker changes. With the method described in section 2.1.4, whereby changes occur at almost any point in any file, the amount of possible speaker changes in turn becomes astronomical. For this reason an exhaustive training of the model using every data combination is implausible, thereby requiring a method for data selection. This is where bootstrap aggregation comes into play; on average a sufficiently large dataset will contain about 63% of the data because each sample has a probability $1-(1-1/N)^N$ of being selected [113]. For large N this probability converges to [113]:

$$1-(1-1/N)^N \sim 1-1/e \approx 0.632 \tag{2.12}$$

2.2 Feature extraction

Feature extraction is the process of extracting relevant information prior to processing, while discarding irrelevant and redundant information.

In more detail feature extraction is the process whereby specific aspects of raw data is accentuated as a means to increase class separation, reduce noise, avoid redundancy, reduce dimensionality and tease apart the products of non-linear behaviours. This step is vital in order to avoid traps like the infamous curse of dimensionality [52].

More specific to speaker diarisation, the extracted features from the pressure wave should contain information able to distinguish speakers and possibly environments. As this project focuses on speaker change detection, features that distinguish speakers take absolute priority. The hope is that even if different environments trigger the speaker change detection algorithm, that the speaker clustering algorithm will be able to merge the relevant sections.

Lu et al. [78] comments that compensating for the effect of the channel or environment mismatch remains a difficult issue in speaker recognition research. They use the Cepstral Mean Subtraction, CMS, algorithm. They however press that CMS alone is insufficient.

2.2.1 Feature type selection

Kinnunen et al. [62] compiled a list of appropriate properties that features for speaker modelling and discrimination should have:

- Have large between-speaker variability and small within speaker variability.
- Be robust against noise and distortion.
- Occur frequently and naturally in speech.
- Be easy to measure from speech signal.
- Be difficult to impersonate/mimic.
- Not be affected by the speaker's health or long-term variations in voice.

In addition to these, a final system must operate computationally efficient and since the number of required training samples for reliable density estimation grows exponentially with the number of features, the number of features must be as low as possible in order to detect short speaker turns:

- Be computationally efficient.
- Small feature vector size.

A range of feature types have been employed in the field of speaker diarisation [86]:

- Short Time Energy, STE, by Meignier et al. [84].
- Zero Crossing Rate, ZCR, by Lu et al. [79].
- Pitch by Lu et al. [77, 78].
- Spectrum magnitude by Boehm et al. [9].
- Line Spectrum Pairs, LSPs, by Lu et al. [78, 79].
- Perceptual Linear Prediction, PLP, cepstral coefficients by Tranter et al. [114] and Chu et al. [19].

- Features based on phoneme duration, speech rate, silence detection, and prosody are also investigated in the literature Wang et al. [119].

Kinnunen et al. [62] categorise the features from the viewpoint of their physical interpretation:

1. Short-term spectral features.
2. Voice source features.
3. Spectro-temporal Features.
4. Prosodic features.
5. High-level features.

Then proceeds to recommend new researchers in the field of speaker change detection to use only the first type. Namely short-term spectral features, on the argument that they incorporate all the properties above, are easy to compute and yield decent performance. Referencing the results by Reynolds et al. [101].

Mel-Frequency Cepstral Coefficients, MFCCs, sometimes with their first and second derivatives are the most common features used (e.g. [23, 54, 79, 107]).

This project is built on the preliminary work for CastSearch by Jørgensen et al. [55] and Mølgaard et al. [87] which exclusively employs MFCCs. As the use of MFCCs is quite common, literally recommended for new researchers and mediates direct comparison with previous work, this project will employ MFCCs as the sole features. Section 2.2.2 will go into detail on attributes and parameters of the MFCCs used in this thesis, section 2.2.3 looks at the largest limitation of MFCCs and section 2.2.4 describes the underlying theory behind MFCCs.

2.2.2 MFCC attributes

For a description of the Mel-Frequency Cepstrum and the Mel-Frequency Cepstral Coefficients, see section 2.2.4 and 2.2.4.2.

Even though the use of MFCCs is very common, the range of MFCCs used remains diverse. The use of 24-order MFCCs seems quite common [4, 17, 18, 115] while Kim et al. [60] applies 23-order MFCCs. Using derivatives is also common, 13-order MFCCs along with their first-order derivatives are consistently applied by Kotti et al. [64, 66], while Wu et al. [122] employs 12-order MFCCs along with their first-order derivatives.

Understandably, in later work, various types of feature selection are being applied to mitigate this issue. Wu et al. [122] investigates several MFCC orders before the 12-order MFCCs along with their first derivatives are chosen. In Kotti et al. [65], an effort is made to discover an MFCC subset that is more suitable to detect a speaker change, with some performance gains, however they do remark that this may reduce generalisability to other datasets. To further this process this thesis applies several types of feature selection, see section 4.1 and 4.3.1.

Also, there is no consensus with respect to first-order MFCC derivatives. While first-order MFCC derivatives are claimed to deteriorate efficiency by Delacourt et al. [24], the use of first-order MFCC derivatives is found to improve performance by Wu et al. [122]. In this thesis this is found to depend on the applied method's sensitivity to dimensionality issues, see section 4.1.1.2.

Since no consensus seems evident and since the 'direct comparison with previous work' argument remains valid; this project will employ 12 MFCCs, with the use of first and second order derivatives and then perform forwards and backwards feature selection, see section 4.1 and section 4.3.1, respectively.

The majority of temporal parameters are borrowed from Jørgensen et al. [55], but it is unclear which specific MFCCs are used. For this reason only 13 Mel-filters are used rather than 20, ensuring that the 12 MFCCs cover the entire range. 13 rather than 12 Mel-filters are used as the first MFCC is discarded since it encodes the log energy of the signal and relying on the volume is obviously a poor indicator for a speaker change detection. In section 4.3.1 backwards feature selection will be applied to extend the 13 Mel-filters up to the 20 used in [54].

In addition, in line with Jørgensen et al. [54] among many others, 20ms windows are used for the Short-Time Fourier Transform, STFT, with 10ms overlap and a hamming window is applied to minimise spectral leakage. This choice seems arbitrary, it is however very common, the earliest example found is by Ahalt et al. [2]. In line with [55] and in line with the sampling frequency of the ELSDSR database a sampling rate of 16KHz is used. Unlike in [54] the MFCCs are standardized [67]:

$$Z = \frac{X - \overline{X}}{s} \tag{2.13}$$

Where Z is the standard score of a raw score X. $\overline{X}$ is the expected value of X, in this case a Gaussian is assumed reducing it to the sample mean, calculated using equation 3.9 and s is the standard deviation of X calculated using equation 3.10.

This has been shown to improve performance in Viikki et al. [118].

2.2.3 MFCCs and noise

Despite the de facto standardization of their use as front-ends, MFCCs are widely acknowledged not to cope well with noisy speech [16]. As written by Chen et al. [16], many techniques have been deployed to improve the performance in the presence of noise, such as Wiener or Kalman filtering [98, 116], spectral subtraction [10, 33, 47, 48, 49], cepstral mean or bias removal [38, 56], model compensation [35, 88], Maximum Likelihood Linear Regression, MLLR, [121] and finally the method applied by the inventors of the RuLSIF method, transfer vector interpolation [92].

The general concept in all of these is to use prior knowledge of the noise to mask, cancel or remove noise during preprocessing or to adjust the relevant parameters to compensate for the noise. However it was realised that applying any of these methods is beyond the scope of this thesis. The necessary source code for adding various types of noise was designed, but testing the method chosen in section 4.1 with it would require source code components that would simply take too long to design and is therefore relegated to further work, see section 5.

The results found in this thesis is necessarily based on a relatively noise free environment. In the context of news editing this should not pose a problem as speaker transitions are rarely from one noisy environment to another. The SCD method found as optimal in this thesis will easily detect a change from a speaker in a noiseless environment to a speaker in a noisy environment, but might have difficulties under changing noise conditions not coinciding with speaker transitions.

2.2.4 MFCC theory

As mentioned in section 2.2, this work is based solely on manipulation of MFCCs.

The Mel-Frequency Cepstral Coefficient, MFCC, features have been used in a wide range of areas. The Mel-Frequency Cepstrum, MFC, originates in speech recognition [100], but has increasingly been used in other areas as well. Such as music genre classification [3, 83], music/speech classification [28, 89] and many others, for instance [32, 123].

The implementation used here is from the Voicebox toolbox by Brookes et al. [13]. This section will go into greater detail on the underlying theoretical aspects, by first examining the cepstrum, extraction of coefficients using the Discreet Cosine Transform, DCT, and then moving onto a description of the Mel scale and its applications.

2.2.4.1 The Cepstrum

This section is to a large degree based on the book, "Discrete-time processing of speech signals" by Deller et al. [25].

Speech in general can be modelled as a filtering of the vocal excitation by the vocal tract, in other words a convolution in the time domain. The vocal excitation, produced by the vocal cords, controls the pitch and the volume of speech. Whereas the shape of the vocal tract controls the formants of speech which define literal semantics and nuances in the speech helpful for speaker discrimination. Taking this into account is shown to improve performance in section 4.3. The speech signal, $s(n)$, is therefore the vocal excitation, sometimes referred to as the excitation sequence [55], $e(n)$, which is convolved with the slowly varying impulse response, $\theta(n)$, of the vocal tract:

$$s(n) = e(n) * \theta(n) \tag{2.14}$$

An initial task in speech data preprocessing is therefore a deconvolution and separation of these different aspects of speech. This separation is useful for a number of reasons, mainly it enables the analysis of the separate aspects individually. This individual analysis is essential since the shape of the vocal tract control literal semantics and is thus useful in speech recognition, whereas the vocal excitation is speaker specific and is therefore used mainly in speaker recognition. Using all the data for SCD is shown to reduce performance in section 4.3.

This deconvolution is where the cepstrum comes into play, the cepstrum is a representation used in homomorphic signal processing, to convert signals combined by convolution into sums of their cepstra, for linear separation. If the full complex cepstrum is generated this process is termed ‘homomorphic deconvolution’ [94, 99].

In the field of speech analysis the cepstrum is particularly useful as the low-frequency excitation and the the formant filtering, which are convolved in the time domain and multiples in the frequency domain, are additive and in different regions in the quefrency domain. Quefrency being the independent variable of the cepstrum, analogues to frequency of the spectrum, in line with the anagrammatic naming convention of the field.

The complex cepstrum, as described by [93], contains all phase information and thereby enables signal reconstruction. However for application in speech analysis, only the real cepstrum is standard and consequently the version implemented in Voicebox [13] and in this thesis employs this version. The real cepstrum, $c_s(n)$, is defined as [99]:

$$c_s(n) = \mathrm{IDFT}\{\log|\mathrm{DFT}\{\mathrm{s(n)}\}|\} \tag{2.15}$$

Where DFT and IDFT are the Discrete Fourier Transform and the Inverse Discrete Fourier Transform respectively and n is the time-like variable in the cepstral domain. Through the use of the Fourier transform the spectrum of the signal becomes a multiplication of the components rather than a convolution:

$$S(k) = E(k)\Theta(k) \tag{2.16}$$

In practice a variation of the Short-Time Fourier transform, STFT, using overlapping hamming windows is applied, see section 2.2.2. The logarithm product rule enables a linear separation of the components; thereby the multiplication becomes an addition, simultaneously the absolute value is used to discard all phase information:

$$\begin{aligned}
\log|S(k)| &= \log|E(k)\Theta(k)| && (2.17)\\
&= \log|E(k)| + \log|\Theta(k)| && (2.18)\\
&= C_e(k) + C_\theta(k) && (2.19)
\end{aligned}$$

Finally to enter the quefrency-domain the IDFT is applied. Since IDFT is a linear operation it applies to each component according to the principle of superposition, giving the real cepstrum of $s(n)$, $c_s(n)$:

$$
\begin{aligned}
c_s(n) &= \mathrm{IDFT}\{C_e(k) + C_\theta(k)\} && (2.20)\\
&= \mathrm{IDFT}\{C_e(k)\} + \mathrm{IDFT}\{C_\theta(k)\} && (2.21)\\
&= c_e(n) + c_\theta(k) && (2.22)
\end{aligned}
$$

For a pictographical summary of the signal transformation process see figure 2.6.

2.2.4.2 Mel-frequency warping

The cepstrum described in section 2.2.4.1, is lacking a key feature often applied, namely a frequency warping used to model the human auditory system. The general idea is that by applying this frequency warping the information is more evenly distributed among the coefficients. In [106] different version are examined and contrasted, including the one applied in the ISP toolbox [53] and therefore the one used here. The use of the Mel scale has been shown to increase performance many times over and is standard practice in the field of speech analysis.

The calculation of the mel-cepstrum is similar to the calculation of the real cepstrum except that the frequency scale of the magnitude spectrum is warped to the mel scale.

Originally proposed by Stevens et al. [110] the mel scale is a perceptual scale of pitch. The scale is based on an empirical study of subjectively judged pitch ratios by a group of test subjects.

It should be mentioned that among others Donald D. Greenwood, a student of Stevens, found the existence of hysteresis effects in the mel scale in 1956. This is mentioned in [109] and was submitted in a mailing list in 2009 [41]. The mel scale does not take these hysteresis effects into consideration and is therefore slightly biased, but has been shown to increase performance nonetheless.

The mel scale is usually approximated by a mapping with a single independent variable, the corner frequency, i.e. the frequency to which the pitch ratio is measured against. This frequency have varied over the years, the currently most popular, and the version implemented in Voicebox [13], was proposed by [80] with a corner frequency of 700 Hz. The 700 Hz version superseded the 1000 Hz version first proposed by [29], on the conclusion that it provides a closer

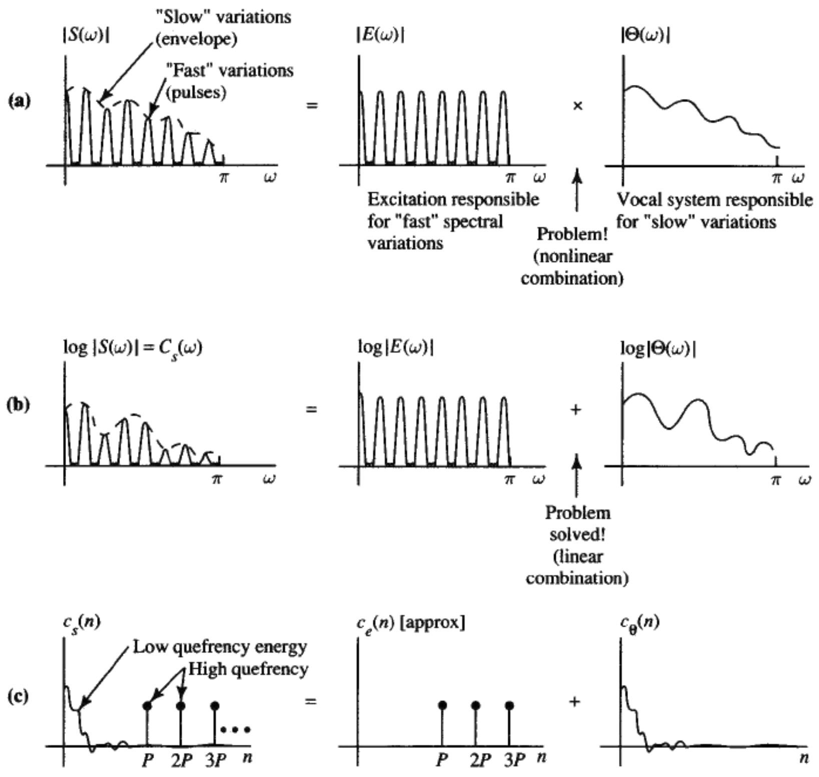
(a)
|S(ω)|
"Slow" variations (envelope)
"Fast" variations (pulses)
=
|E(ω)|
×
|Θ(ω)|
Excitation responsible for "fast" spectral variations
Problem! (nonlinear combination)
Vocal system responsible for "slow" variations
(b)
log |S(ω)| = C_s(ω)
=
log |E(ω)|
+
log |Θ(ω)|
Problem solved! (linear combination)
(c)
c_s(n)
Low quefrency energy
High quefrency
=
c_e(n) [approx]
+
c_θ(n)
π ω
P 2P 3P n

Figure 2.6: Shows how a speech signal is composed of a slowly varying envelope part convolved with quickly varying excitation part. By moving to the frequency domain, the convolution becomes a multiplication. Further taking the logarithm the multiplication becomes an addition. Thereby neatly separating the components of the original signal in its real cepstrum. Figure borrowed from [25].

approximation of the mel scale, especially at high and low frequencies [36]:

$$f_{\text{mel}} = 2595 \log_{10}(1 + \frac{f}{700}) \tag{2.23}$$

Where f is frequency.

Using this mel frequency mapping the MFCCs are calculated according to figure 2.7. In line with section 2.2.4.1, the only addition is the step for discretizing the cepstrum, the discrete cosine transform. The MFCCs are computed from the Fast Fourier Transform, FFT, power coefficients which are filtered by a triangular band pass filter bank. The filter bank used in this thesis consists of 12 triangular filters later expanded to 20, see section 2.2.2. The MFCCs, C_n, are calculated as [99, 124]:

$$C_n = \sqrt{\frac{2}{K_{\text{mel}}}} \sum_{k=1}^{K_{\text{mel}}} (log_{10} S_k) \cos\left(\frac{n(k-\frac{1}{2})\pi}{k}\right) \tag{2.24}$$

Where $S_k(k = 1, 2, \dots, K_{\text{mel}})$ is the outputs of the filter bank and $n = (1, 2, \dots, N)$ are the samples in a 20ms audio unit. The $\sqrt{\frac{2}{K_{\text{mel}}}}$ normalisation factor is added so that the inverse does not require an additional multiplicative factor, this is used to make the transform matrix orthogonal.

As mentioned in section 2.2.4.1, ideally only the vocal excitation part of the speech signal is wanted and therefore only the highest coefficients of the mel-cepstrum are extracted. As mentioned in section 2.2, this thesis initially uses all coefficients except the lowest from the 13 filters, and subsequently find improved results by using only the 12 upper coefficients from 20 filters. In addition this project includes the first and second order derivatives of these coefficients, as mentioned in section 2.2.

An interesting note is that some authors, including [23] have commented that the spectral basis functions of the cosine transform in the MFC are very similar to the principal components of the spectra, which were applied to speech representation and recognition much earlier by Pols et al. [97, 96]. Thereby giving credit to the whole notion of MFCCs as a means to accurately represent spectral content efficiently. Indeed it has even been found that MFCCs if produced from a very wide range of sounds, might be similar to an Independent Component Analysis, ICA, of sound perceived by mammalian ears in general [43][71].

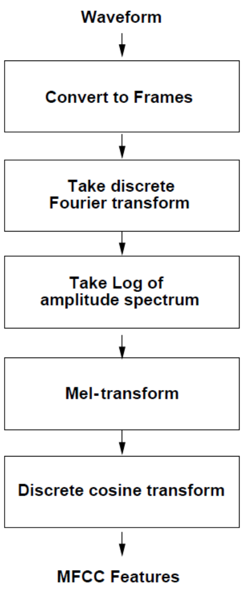
Waveform
Convert to Frames
Take discrete Fourier transform
Take Log of amplitude spectrum
Mel-transform
Discrete cosine transform
MFCC Features

Figure 2.7: Process by which the MFCC features are calculated. This figure is meant as an overview, not a summery, minor steps like windowing are overlooked for concision. Section 2.2.4 is dedicated to explaining the process portrayed in this figure.

2.3 Change-point detection

The end goal is to hypothesize a set of speaker change-points, all methods however only produce a metric in which peaks correspond to possible change-points. This in essence boils the problem of change-point detection down to a need for a 1D-peak detection algorithm. In Jørgensen et al. [55] a suitable algorithm is proposed, which includes a series of steps and a few free parameters.

A change-point is defined as a local, within T_i seconds, maxima above the threshold, M_{cd}. This threshold is the moving average over a region, $2 * T_{max}$ seconds, with a gain, α_{cd}, applied to the threshold.

$$M_{cd} = \alpha_{cd} \frac{1}{2 * T_{max} + 1} \sum_{i=n-T_{max}}^{n+T_{max}} \text{metric}_i \tag{2.25}$$

These free parameters T_i, T_{max} and α_{cd} need to be fixed or trained. This thesis defines a speaker turn as being above at least 1 sec. Therefore, in accordance with [54], T_i is set to 2 seconds (other peaks within ± 1 second are ignored).

In Jørgensen et al. [55], T_{max} is used for multiple purposes, in addition to defining a length for the moving average it defines the maximum use of data in the False Alarm Compensation algorithm, FAC. A discussion of why and how T_{max} is determined will be delineated in section 2.4, after the FAC method has been explained.

The threshold gain parameter, α_{cd}, is trained empirically through a search, in the free parameter space, for a maximal F-measure, see section 3.3.

2.4 False Alarm Compensation

The common approach to speaker change detection is that of a comparison between subsequent segments of a fixed length [54][111][105]. This approach however is saddled with an inherent problem, namely the ability to detect short speaker turns, requiring short analysis windows, versus the ability to avoid false positives, requiring long analysis windows, see section 1.1.3. The literature proposes several solutions to this dilemma.

- Algorithms that successively build larger segments rather than comparing equal sized segments [42].

- Methods for running several analysis window lengths in parallel and then combining the results [7].

- Iterative methods, where the algorithm is run recursively with successively longer analysis windows on the successively smaller amount of possible change-points in order to reject false alarms. This class of methods has been termed False Alarm Compensation, FAC, and is employed by [54]. Here the false alarm compensation step was run only once.

This project employs a variant of the FAC method employed in Jørgensen et al. [54], the difference being an investigation into the benefit of a multistep procedure, here termed Recursive FAC, RFAC, see section 2.4.1 and 5.2.

This approach has a number of free parameters; A buffer variable equal to T_i, see section 2.3 used as a buffer zone around a hypothesized change-point, a threshold gain parameter, α_{FAC}, and finally, T_{max}, the maximum data used from available data on each side of a hypothesized change-point.

α_{FAC} is a gain applied to the moving average threshold calculated on the metric, M_{FAC}, similar to the variable, α_{cd}, see section 2.3. α_{FAC} times this moving average threshold is the function below which a hypothesized change-point is rejected. Only two thresholds exist, the change-detection one which is the moving average of the metric with the gain α_{cd}, and the same moving average with the gain α_{FAC}.

$$M_{FAC} = \alpha_{FAC} \frac{1}{2 * T_{max} + 1} \sum_{i=n-T_{max}}^{n+T_{max}} \text{metric}_i \tag{2.26}$$

The threshold gain parameter, α_{FAC}, is trained empirically through a search, in the free parameter space, for a maximal F-measure, see section 3.3.

In Jørgensen et al. [55] it is argued that in order to maintain correlation between the FAC threshold and the change-point threshold, T_{max}, the maximum analysis window and length of the moving average must be identical across algorithms.

T_{max} has the additional purpose of maintaining speaker turn purity. I.e. if a speaker-change point is missed during the first iteration, the resulting speaker turn data would contain multiple speakers. T_{max} counteracts this issue by using the data closest to the hypothesized change-point in question, thus using as much data from the true speaker as possible and completely purifying the data if the true speaker turn is longer than T_{max}.

2.4.1 Hybrid method

This section will slightly modify the basic FAC method mentioned in section 2.4, by proposing a robust novel approach where metrics are combined to compensate for dissimilar weaknesses. A novel approach is proposed as a literature search failed to yield any solutions to the problems described in this section.

In section 4.1.3.1, it is shown that combining different metrics could potentially increase performance, the reason being that they might be able to complement each other on several key points. As mentioned in section 2.4, the idea of a false alarm compensation paradigm is to incrementally increase the amount of data available to the metric to progressively refine its estimate at a potential change-point, using the earlier iterations to determine how much data is actually available. During these iterations potential change-points that are found to be false alarms are deleted freeing up data for neighbouring potential change-points.

The metrics, see section 3.2, vary in computational complexity and performance, with the computationally heavy metrics generally outperforming the light metrics, see section 4.1.3. Using a light metric to quickly detect a large amount of potential change-points and then subsequently applying a heavy metric on the small data set to filter out the false alarms seems logical.

This combined metric approach however has a serious issue at its core. In the standard FAC paradigm the FAC threshold is merely a scaled version of the threshold applied in the change-detection step, see section 2.4. As the same metric is usually applied to both change-detection and FAC the only difference is in the amount of data the metric is working with. I.e. in the standard FAC paradigm it is assumed that the FAC metric assumes roughly the same range of values as the change-detection metric irrespective of the amount of data used and that peaks will merely be scaled versions depending on the data amount.

When combining metrics both of these assumptions fail and must be handled. The novel approach proposed here involves discarding the notion of using the moving-average of the change-detection step in the FAC step altogether.

Ideally the FAC metric is only evaluated at potential change-points, this however poses a problem as a reference is required in order to determine a threshold to which the FAC metric is compared. As one of the strengths of the combined metric approach is that they treat the data differently, i.e. accentuating different aspects, using a threshold based on a scaled version of the change-detection metric is invalid.

What is proposed instead is a heuristic for estimating a noise floor of the FAC metric through information gathered from the change-detection step. The basic assumption is that if the change-detection metric has a low threshold, thereby proposing a large amount of potential change-points, then a position far from a potential change-points is very unlikely to actually be a change-point.

The Combined Metric FAC, CMFAC, is therefore given the list of potential change-points along with a list of not-change-points. It is then saddled with the job of identifying which of the potential change-points are different enough from the not-change-points. These not-change-points are chosen as close to the center as possible between the proposed change-points.

This novel CMFAC method applies its particular metric, RuLSIF is used in section 5.2, to each of the not-change-points using as much data as possible up to T_{max}, in exactly the same fashion as the standard FAC method, see section 2.4. Thereby acquiring a list of values that probably correspond to the noise floor of the CMFAC metric. This list might however still contain change-points or other rare events, this is handled by taking the median of this list, thereby acquiring a single value very likely to be representative of the noise floor.

This single value is then used as a constant threshold to which the gain, α_{FAC}, is trained, see section 3.3. The hope is that the disadvantage of trading the moving average threshold of standard FAC for the constant median threshold of the CMFAC approach will be out weighted by the benefit of a more expensive and more thorough metric. A metric that does not have to correspond to the metric used in the change-detection step.

This median approach is chosen as that will enable the algorithm to apply even in the case of news editing, where only a few speaker changes occur. The CMFAC method is applied in section 5.2, with KL and RuLSIF used as change-detection metric and CMFAC metric respectively. There the performance is compared to standard FAC using KL with KL and RuLSIF with RuLSIF.

Chapter 3

Methodology

This chapter will delineate the theoretical ground work. To begin with, a short introduction to the concept of metrics and how they relate to the actual sound, is bridged using a spectrogram and a graphical representation of the MFCCs. Once this is in place, the independent approaches to SCD are described in detail, these include; Vector Quantization, Gaussian based approaches and Relative Density Ratio Estimation.

This chapter will then describe the parameter optimisation technique used to train the change detection and the FAC thresholds. These methods are applied in section 4.1.2. Finally this chapter will conclude with some miscellaneous basic required theory.

3.1 Metric introduction

This section will try to bridge the gap from sound pressure up one abstraction level to the SCD F-measure score using the various methods.

The upper graph seen in figure 3.1 shows a spectrogram of some speech data, i.e. with frequency on the y-axis, time on the x-axis and magnitude encoded in the color scale going from blue, low, to red, high. The speech is recognisably producing patterns, but obviously distinguishing between speakers is difficult at best.

Using the feature extration methods described in section 2.2; The MFCCs are shown in the middle graph which to a large degree resembles noise, but most assuredly contains SCD cues. Slightly confusingly the frequency scale in the middle graph is an inverted version of the upper graph, this is a remnant of an alignment process which was more straightforward this way and given that the MFCCs are standardized the relevance is negligible.

The bottom graph displays the metrics of the various methods applied to the MFCCs and here the speaker changes are obviously very detectable. The horizontal dotted green lines represent actual changes from one speaker to another, the red ones denote changes between files containing the same speaker, these are obviously invalid and should not get detected. The metrics are designed to peak at speaker changes and are independent of the green and violet circles.

These green and violet circles represent the change-detection and FAC algorithms respectively applied to the metrics, with a green simply signifying that the algorithm thinks it encountered a speaker change, not denoting the success. The violet circles represent potential speaker change-points - formerly green circles - that the FAC algorithm has flagged as false alarms. Note that a potential change-point must be within 1 second of the actual change to count towards the F-measure score seen in the legend. The F-measure being the harmonic mean between hits-to-tries ratio and hits-to-possible-hits ratio.

3.2 Speaker dissimilarity metrics

To determine the dissimilarity of two subsequent analysis windows a suitable dissimilarity metric must be chosen. In the literature many such measures have been proposed, in this project some of the most common approaches are singled out and compared, this section will describe the theory behind these and the limitations of them. The metrics will then be trained, see section 4.1.2, and compared using test data, see section 4.1.3, to determine their performance in the context segment comparison and false alarm compensation. In must be noted that alternatives to the metric based approach, when comparing analysis windows, are used in the literature, but will not be tested in this thesis. As an example Henkel et al. [46] compares distributions in the framework of hypothesis testing.

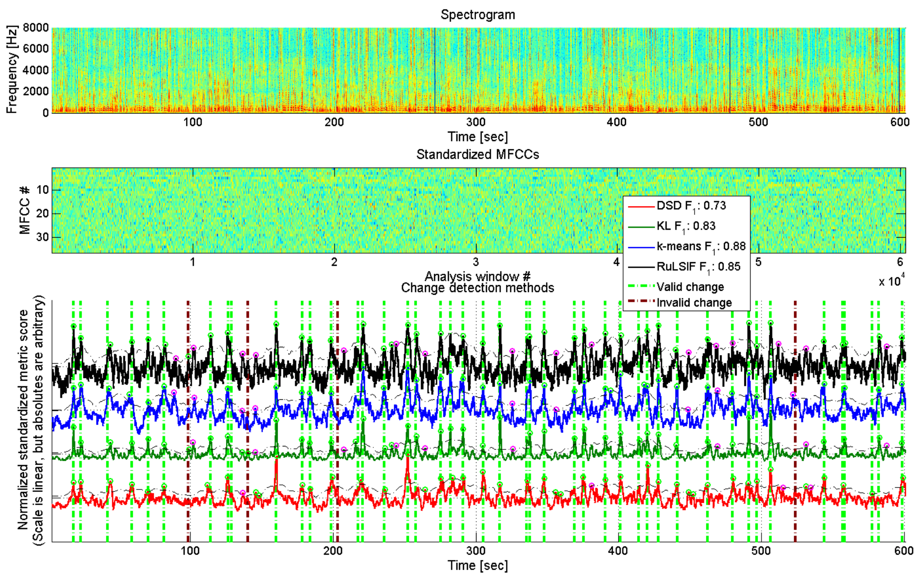

Figure 3.1: The upper graph shows a spectrogram of the data, i.e. with frequency on the y-axis, time on the x-axis and magnitude encoded in the color scale going from blue, low, to red, high. The standardized MFCCs are shown in the middle graph. Slightly confusingly the 'frequency scale' in the middle graph is an inverted version of the upper graph. The bottom graph displays the various metrics applied to the MFCCs. The horizontal dotted green lines represent actual changes from one speaker to another, the red ones denote changes between files containing the same speaker, these are obviously invalid and should not get detected. The metrics are designed to peak at speaker changes and are independent of the green and violet circles. These green and violet circles represent the change-detection and FAC algorithms respectively, with a green simply signifying that the algorithm thinks it encountered a speaker change, not denoting the success. The violet circles represent potential speaker change-points that the FAC algorithm has flagged as false alarms. As seen the FAC algorithm does fairly well, but sometimes marks correct change-points as false alarms, see upper right on bottom figure. Note that a potential change-point must lie within 1 second of the actual change to count towards the F-measure score seen in the legend. The F-measure being the harmonic mean between the hits-to-tries ratio and hits-to-possible-hits ratio.

3.2.1 Vector Quantization

Vector quantization, VQ, [39, 40] incorporates a variety of approaches to 'discover' the underlying structure of a dataset through iteratively improved guesses. More technically VQ is a central topic in signal processing, a sub-field of competitive learning and deals with the problem of representing a dataset by means of a smaller, finite size, K, codebook of representative data points or 'centroids' [50]. The size of K must be chosen to match the application and how the centroids are initially chosen as well as how they are updated depends on the choice of algorithm. An example of a simple VQ algorithm could be:

1. Pick a sample point at random.
2. Move the nearest centroid towards this sample point.
3. Go to step 1.

An input belongs to cluster i if i is the index of the closest centroid, here distance is measured using the Euclidean distance. This has the effect of dividing up the space into a Centroidal Voronoi tesselation, CVT, a high dimensional beehive-esque structure where the cells size is inversely proportional to the data density and the shape is determined by the proximity to other cells [26].

Many different algorithms for various purposes have been designed, within the field of speaker diarisation VQ has found its uses in many different aspects, such as word recognition [14], speaker recognition [108], and of cause speaker change detection [54].

3.2.1.1 Choice of algorithm

This section will describe the VQ algorithm choice, for a description of the choice see section 3.2.1.2.

Within the field of VQ there are two distinct groups of algorithms, the on-line algorithms and the batch algorithms. Batch algorithms recomputed all values at each iteration, whereas on-line algorithms compute only a single value each iteration. An example of this distinction is the standard K-means [8] vs. the so called Winner-Takes-All, WTA, version of K-means [45].

On-line does not necessarily refer to new data coming in, as VQ is sensitive to sequential data that is non-i.i.d. - independent and identically distributed - these centroids chosen for updates are either randomly or carefully selected [45].

On-line methods that update sequentially on incoming data are not directly applicable in the context of speech as the signal varies far too quickly to be of practical use, beyond this the common approach, when analysing temporal effects in the signal, is to use derivatives of the input along with the input, this is discussed in section 2.2.

Ahalt et al. [2] compares several algorithms in the context of speech data, but finds no significant difference in performance, signifying that the particular choice of training algorithm is to some degree irrelevant performance wise. The algorithms included the Self Organizing Maps [63], SOM, the Frequency-Sensitive Competitive Learning, FSCL, algorithm neural network [50, 108] and the Linde–Buzo–Gray algorithm, LBG, [73]. The LBG algorithm is an extension of Lloyd's algorithm [76] and similar to the more well-known K-means algorithm [8]. The result is similar in another study, this time by Kinnunen et al. [61], where five different clustering algorithms are compared for use in a speaker identification setup using vector quantization. The comparison shows only marginal difference in the results obtained with the five algorithms.

Beyond performance this still leaves attributes like efficiency, robustness and convergence. In lieu of other work that compares different algorithms, a robust common choice must be made. The previous work this project builds on Jørgensen et al. [55] used a version of the K-means algorithm and proposed some optimisations, see section 3.2.1.4, along with finding overall decent results using it. For this reason the K-means algorithm will be employed for use in VQ. The convergence properties, the limitations and the general principals of the K-means algorithm will be discussed in the following subsections.

3.2.1.2 K-means

Given a dataset $\{x1, \ldots, xn\}$ consisting of N observations of arbitrary dimensionality. The goal of K-means is to group the data into K pure clusters with the minimum within-cluster sum of squares, WCSS, distance. As such the objective function for K-means is given as [8]:

$$\text{WCSS} = \sum_{n=1}^{N}\sum_{k=1}^{K} r_{nk}||x_n - \mu_k||^2 \tag{3.1}$$

Where μ_k are the cluster centroids and r_{nk} is a vector signifying to which cluster x_n belongs:

$$r_{nk} = \begin{cases} 1 & \text{if } k = \arg\min_j ||x_n - \mu_k||^2 \\ 0 & \text{otherwise.} \end{cases}$$

This objective function is also referred to as a distortion measure by bishop [8] and is also used as the metric to quantify the change in the signal, see section 3.2.1.3.

The objective of the K-means algorithm is to solve the NP-hard problem [6] of finding the optimal combination of μ_k and r_{nk} for minimising WCSS heuristically.

This heuristic is a 2-step procedure similar to the more general EM-algorithm [8] used in the also very established Gaussian Mixture Model, GMM, [8] approach which has also found its uses in SCD [68, 81]. The GMM approach was originally included in this thesis, but was cut both for efficiency reasons and due to the difficulty in defining a suitable dissimilarity metric, which lead to the field of the Adapted Gaussian Mixture Models, AGGM, [81] or out of this paradigm altogether into the Hidden Markov Model, HMM, [68].

The basis of this heuristic is the fact that a closed form solution of the objective function is easily found looking at a specific cluster k, C_k. The objective function 3.1 is quadratic in μ_k and therefore convex, meaning that a local minimum is the global minimum. The minimum is found by minimising the derivative of WCSS with respect to μ_k:

$$\frac{d\text{WCSS}}{d\mu_k} = \sum_{n=1}^{N} \sum_{k=1}^{K} 2r_{nk}(x_n - \mu_k) = 0 \tag{3.2}$$

Where μ_k is just the arithmetic mean of x_n found by solving for μ_k:

$$\mu_k = \frac{\sum_{n=1}^{N} r_{nk} x_n}{\sum_{n=1}^{N} r_{nk}} \tag{3.3}$$

Since the denominator is the total number of points in cluster k, S_k, and the numerator is the sum of all points in cluster k, C_k:

$$\mu_k = \frac{\sum_{x_n \in C_k} x_n}{S_k} \tag{3.4}$$

This is also the reason why the method is called K-means.

Using this closed form solution a local minima to the NP-hard problem [6] is found using this algorithm [8]:

- **Initialisation step:** Initially Initialise centroids using Forgy's method [34], which simply uses k data points drawn at random. On subsequent analysis windows use previous centroids, see section 3.2.1.4.
- **Assignment step:** Assign each data point to the cluster whose centroid is closest to it. Where each data point x_n belongs to only one cluster, which is not a problem here as the distance measure used is Euclidean.

  If any cluster is empty, reinitialise it using the singleton method, i.e. move it to the data point furthest from all centroids.
- **Update step:** Calculate the new position for the centroids of each cluster using equation 3.4.

  If the stopping criterion is not met, return to the assignment step.
- **Stopping criteria:** The algorithm has converged when the no centroids were moved. This stopping criteria is shown in [11] to be stable, but as a fail-safe the algorithm will stop if a large maximum number of iterations have passed.

The MATLAB function **kmeans** from the statistics toolbox is used to implement K-means [82].

3.2.1.2.1 Limitations of K-means

K-means as a method for clustering has some stringent limitations that must to be addressed.

In particular K-means in the context of clustering has problems [8] when clusters have:

- Differing sizes
- Differing densities
- Non-globular shapes
- Outliers

Most of these concerns however are not as crucial here as K-means is used to approximate the underlying probability density function, PDF, not to identify meaningful clusters in the data.

As seen in figure 2.2 the data in the MFCC space is roughly globular, but does contain some outliers. What these outliers signify and by extension what individual clusters found by K-means signify is not investigated in this thesis.

These limitations do however limit K-means' ability to approximate nuances in the PDF and will therefore skew the results slightly as a difference between subsequent analysis windows might not be representative of the underlying PDF, but in how K-means approximates it. How this affects the results is not investigated in this thesis, however this issue is inverse proportional to the size of K, see section 4.1.1.3.2 for selection of codebook size.

Beyond this K-means have difficulties when unfortunate initialisation or iterations results in empty clusters. This thesis applies the singleton approach wherein empty clusters are assigned to the data point furthest from any centroid. This singleton approach obviously makes the K-means method even more outlier sensitive. The larger the codebook is the more likely it is that an outlier will be assigned as its own cluster, but the less likely it is that it will significantly impact that approximation. This issue however is mitigated due to how differences between codebooks are calculated, see section 3.2.1.3, since a cluster is merely compared to its closest neighbour in the previous codebook, not its previous position.

3.2.1.2.2 Codebook size

The choice of codebook size, K, and feature selection in the context of K-means is investigated in section 4.1.1.3.

3.2.1.3 Choice of metric

Given that the analysis windows are clustered using the K-means algorithm described in section 3.2.1.2, a crucial step remains, which is a choice of metric for comparing windows. Since the K-means algorithm is designed to minimise the WCSS, the obvious choice is to use the previous analysis window as training data and the subsequent analysis window as test data, thereby directly measuring the WCSS using equation 3.1.

This approach has been termed Vector Quantization Distortion, VQD, by Nakagawa et al. [90]. In summary VQD can be expressed as:

$$\mathrm{VQD}(C^A, B) = \frac{1}{N}\sum_{n=1}^{N} \underset{1\le k\le K}{\mathrm{argmin}}\{\mathrm{d}(c_k^A, x_n^B)\} \tag{3.5}$$

Where c_k^A denotes the k^{th} code-vector in C^A, with $1 \le k \le K$. $\mathrm{d}(\cdot,\cdot)$ is the distance between two feature vectors, in this context the well-known Euclidean distance is used:

$$d_{\mathrm{Euclidean}}(p,q) = d_{\mathrm{Euclidean}}(q,p) = \sqrt{\sum_{d=1}^{D}(q_d - p_d)^2} \tag{3.6}$$

Where p and q are J dimensional vectors, subscript d denoting dimension.

Thus, the VQD measure 3.5 is the mean distances from all the feature vectors x_n^B in the later analysis window B to the nearest code-vector in the prior analysis window c_k^A. The keen observer will notice that 3.5 is identical as a metric to 3.1 when using a constant analysis window size, since the normalization in this case is irrelevant.

This metric is slightly altered in order to optimise K-means, see section 3.2.1.4.

3.2.1.4 Optimisations

Two basic optimisations to the standard K-means / VQD approach of Nakagawa et al. [90] are given in Jørgensen et al. [55]. These have been included in the algorithm as they substantially improve run time, at almost no cost [55].

The first optimisation is in the initialisation approach, this is also mentioned in the algorithm described in section 3.2.1.2. For the very first analysis window the Forgy initialisation method [34] is used, wherein the centroids are just placed on random data points without overlap. This initialisation method is bias free, but is quite expensive computationally, as a large amount of steps are needed for convergence, see section 3.2.1.2. The proposed method is to initialise subsequent analysis window codebooks by the codebook of the previous analysis window.

The idea behind the proposed method is that subsequent analysis windows are either of the same speaker or not, and that the codebooks should be person specific. This means that if the previous codebook was not a good approximation of the next analysis window, then a large change in the codebook should occur, signifying a speaker change. This optimisation improves the run-time by a factor of about 2.5 using a codebook size of $K = 56$.

The second optimisation applies the assumption that the codebook of an analysis window is a fair approximation of said analysis window. Thereby replacing all data points, x_n^B, in equation 3.5, with the codebook for analysis window B, C_k^B:

$$\mathrm{VQD}_{\mathrm{Optimised}}(C^A, C^B) = \frac{1}{N} \sum_{n=1}^{N} \underset{1 \le k \le K}{\operatorname{argmin}} \{ d(c_k^A, c_k^B) \} \tag{3.7}$$

This assumption is the basic assumption of VQ in general, see section 3.2.1 and is therefore a logical necessity. This optimisation improves the run-time by a factor of about 7, using $K = 56$, see section 4.1.1.3.2 for choice of codebook size.

In total these optimisations improve the run-time of the K-means algorithm by a factor of about 15, enabling its use, for a deeper analysis into the computational efficiency of the various approaches compared to real time, see section 4.1.3.2.

3.2.2 Gaussian based approaches

This section will describe the second group of approaches tested in this project in the context of SCD, which is the single multivariate Gaussian approach, along with its individual sub constituents. This approach is common in the context of SCD, and has been shown to perform decently [54, 85].

This approach is basically modelling each analysis window as being drawn from its individual multivariate Gaussian and then comparing analysis windows by the distortion of the estimated underlying PDF. These distortion measures include the Kullback-Leibler distance, KL, and its constituents, the Divergence Shape Distance, DSD, and the Euclidean distance between the sample means.

The distance between sample means is implemented using VQ and a K-means algorithm with a codebook size of K = 1, in which case it degenerates into the Euclidean distance between sample means, see section 4.1.1.2. The sections below will describe the KL and DSD dissimilarity measures, 3.2.2.3 and 3.2.2.4 respectively.

3.2.2.1 Estimation of population parameters

A multivariate Gaussian distribution has two population parameters, the population mean, μ, and the population covariance matrix, Σ, and is given by [8]:

$$\mathcal{N}(x|\mu,\Sigma) = \frac{1}{(2\pi)^{\frac{D}{2}}}\frac{1}{\sqrt{|\Sigma|}}\exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right) \tag{3.8}$$

Where D is the number of dimensions, x is a D-dimensional vector in the MFCC space, x^T and x^{-1} signifies the transpose and inverse of x respectively and finally the population parameters μ and Σ which must be estimated empirically.

The population mean is estimated using the sample mean [8], which is basically the arithmetic mean of every dimension individually:

$$\overline{x} = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{3.9}$$

Where N is the sample size.

The calculation of the sample covariance matrix is slightly more complicated, as it has the disadvantage of being very prone to outliers [51, 117]. Section 2.1.3.1 analysis how prone the MFCC data is to outlier behaviour, as seen, outliers are rare and the data follows a multivariate normal distribution to a good approximation. For this reason an unbiased estimator of the covariance matrix is used and is given by [8]:

$$q_{jk} = \frac{1}{N-1}\sum_{i=1}^{N}(x_{ij}-\overline{x}_j)(x_{ik}-\overline{x}_k) \tag{3.10}$$

Where q_{jk} are the entries of the sample covariance matrix, Q, Which is just the average covariance of every pair of dimensions, where $N-1$ is used rather than N, since a D.o.F was spent calculating the sample mean. This is implemented using the MATLAB function **cov**.

3.2.2.2 Limitations of the Gaussian approach

Since the data can, to a good approximation, be modelled as drawn from a multivariate Gaussian, see figure 2.2, and since it appears to be a simple model, this approach appears quite optimal. This supposedly simply approach does however have quite significant limitations. The main limitation [8] being that it fails to utilise the large amount of parameters effectively. The method is intrinsically uni-modal and as such, even if that is sufficient for modelling a speaker, it will fail if multiple speakers are present in the same analysis window, i.e. if the analysis window crosses a change-point. Even given this fairly large disadvantage the number of independent parameters needed, P, is quadratic in the dimensionality, D, more precisely the amount of parameters is given by [8]:

$$P = (\frac{D^2}{2} + \frac{D}{2}) + D \tag{3.11}$$

Where the first 2 terms are the parameters needed for the covariance matrix, divided by 2 since the covariance matrix is symmetric, with the second term adding the missing half of the diagonal and the last term are the parameters needed for the mean. In this case, given the 36 dimensional input space, see section 2.2, the amount of parameters used is $P = 702$, more than twice the 300 MFCCs in an analysis window, see section 2.2.2. One should think that the obvious over fitting issues would be a detriment to performance and that using the small input space of $D = 12$, $P = 90$, would circumvent these issues. This however appears not to be the case, see section 4.1.1.2 for performance depending on attributes and dimensionality of the input space. Irrespective of the limitations this approach is very computationally efficient as the models can be directly applied, as opposed to the VQ approach, see section 3.2.1.2, where no closed-form solution for the model parameters in known. The fact that the parameters can be directly calculated results in a very short runtime, see section 4.1.3.2.

3.2.2.3 Kullback Leibler Distance

Information theoretically the relative entropy, or the information gain, between two PDFs is called the Kullback-Leibler divergence [69, 105]:

$$\mathrm{KL}_{\mathrm{divergence}}(A; B) = \int_{-\infty}^{\infty} P_A \ln\left(\frac{P_A}{P_B}\right) \mathrm{d}x \tag{3.12}$$

This measure is not directly applicable as a metric, as it is a divergence and is obviously not symmetric. This is handled by introducing the notion of Relative Cross Entropy [105], or in common parlance the Kullback Leibler distance, often abbreviated KL2, here for practical reasons referred to simply as KL. The Kullback Leibler distance merely solves the issue by looking from both directions at once:

$$\begin{aligned}\mathrm{KL}_{\text{distance}}(A;B) &= \mathrm{KL}_{\text{divergence}}(A;B) + \mathrm{KL}_{\text{divergence}}(B;A) && (3.13)\\ &= \int_{-\infty}^{\infty} (P_A - P_B)\ln\left(\frac{P_A}{P_B}\right)\,\mathrm{d}x && (3.14)\end{aligned}$$

A full derivation of the Kullback Leibler distance can be found in the original work by Kullback and Leibler [69].

Inserting equation 3.8 into 3.13 and moving into the discrete domain, yields [55]:

$$\begin{aligned}\mathrm{KL}_{\text{distance}}(A;B) = &\frac{1}{2}\mathrm{Tr}\left[(\Sigma_A - \Sigma_B)(\Sigma_A^{-1} - \Sigma_B^{-1})\right] + \\ &\frac{1}{2}\mathrm{Tr}\left[(\Sigma_A^{-1} - \Sigma_B^{-1})(\mu_A - \mu_B)(\mu_A - \mu_B)^T\right] \qquad (3.15)\end{aligned}$$

This is the implemented version of the Kullback-Leibler distance, see appendix A.3.2. The Kullback-Leibler distance has been applied broadly, with decent performance, in the field of speaker diarisation [54, 85, 105].

3.2.2.4 Divergence Shape Distance

In Lu et al. [78] it was found that environment to a large degree is encoded in the position of the Gaussian in space, or its mean. To circumvent this the Kullback-Leibler distance 3.15 was applied without the second term involving the means, a measure called the Divergence Shape Distance, DSD:

$$\mathrm{DSD}(A;B) = \frac{1}{2}\mathrm{Tr}\left[(\Sigma_A - \Sigma_B)(\Sigma_A^{-1} - \Sigma_B^{-1})\right] \qquad (3.16)$$

As such the DSD is used as a metric for determining the divergence of the covariance matrix. All data in this project, see section 2.2.3, is as environment-free as possible, given that it is not recorded in an anechoic chamber. To avoid any effects the environment could have on the performance, code was developed for applying various noise types, but due to lack of resources this has been relegated to further work in section 5. The DSD measure is however included to determine whether excluding the means improves the performance even in an environment free setting. This appears to not be the case, see section 4.1.1.2.

3.2.3 Relative Density Ratio Estimation

The field of change-detection in general is vibrant with new methods and approaches. This section will describe how and why a novel approach development by Liu et al. [75] is applied. Let it be noted that for unreferenced statements in this section [75] is implied.

The other applied approaches, see section 3.2.1 and 3.2.2, suffer from a core deficiency that this new approach circumvents. They estimate individual models for the preceding and the succeeding analysis windows, then subsequently estimate the difference between these, discarding the individual models, see figure 1.2. This section will describe a novel approach in the field of change-detection that directly estimates the relative density-ratio. The applied method is called Relative unconstrained Least-Squares Importance Fitting, RuLSIF, and is an extension of the more comprehensible Kullback-Leibler Importance Estimation Procedure [112], KLIEP, by Sugiyama et al.

RuLSIF was introduced to overcome certain deficiencies of KLIEP. These deficiencies are mainly extreme sensitivity to outliers, see section 3.2.3.1.1, and the lack of a closed-form solution [112]. Along with the avoidance of these deficiencies RuLSIF provides a set of advantages, these include; an improved non-parametric convergence rate [57, 125], numerical stability [58] and robustness [111].

This section will first outline KLIEP and its relation to the KL method described in section 3.2.2.3, as well as proceed to extent KLIEP to the full RuLSIF method.

3.2.3.1 From KLIEP to RuLSIF

The KLIEP method is related to the KL method described in section 3.2.2.3, in that it is based on the KL-divergence. KLIEP however takes a non-parametric approach and directly approximates a measure similar to the KL distance.

The essence of the density-ratio estimation is the notion of a function, $g(x|\theta)$, called the density-ratio that when multiplied by the PDF of the succeeding analysis window, P_B yields the PDF of the preceding analysis window, P_A:

$$P_A = g(x|\theta)P_B \tag{3.17}$$

$$\Leftrightarrow$$

$$g(x|\theta) = \frac{P_A}{P_B} \tag{3.18}$$

In other words estimating the density-ratio, $g(x|\theta)$, can be expressed as an optimisation problem, whereby $g(x|\theta)$ is estimated through a minimisation of the KL-divergence 3.12:

$$\mathrm{KL}_{\mathtt{divergence}}(A;B) = \int P_A \log_{10}\left(\frac{P_A}{P_B g(x|\theta)}\right) \mathrm{d}x \tag{3.19}$$

Where the PDFs, P_A and P_B are functions of x. To elaborate; in practice $g(x|\theta)$ is approximated as seen below, 3.22, if approximated ideally the $\mathrm{KL}_{\mathtt{divergence}}(A;B)$ is naturally:

$$\mathrm{KL}_{\mathtt{divergence}}(A;B) = \int P_A \log_{10}\left(1\right) \mathrm{d}x = 0 \tag{3.20}$$

Indicating no information gain from P_A to $g(x|\theta)P_B$.

The reader will have noticed that the logarithm base has changed from 3.12 to 3.19. To dispel confusion it must be noted that the KL divergence holds irrespective of log base [69]. Logarithms in these formulae are taken to base 2 if information is measured in units of bits and to base e if information is measured in nats. Too simplify the implementation the version used here uses base 10 as chosen by Liu et al. in [75] on which the work here is based.

The trick then is that by the quotient logarithm rule [1] the unknown PDFs P_A and P_B can be ignored:

$$\mathrm{KL}_{\mathtt{divergence}}(A;B) = \int P_A \log_{10}\left(\frac{P_A}{P_B}\right) \mathrm{d}x - \int P_A \log_{10}(g(x|\theta))\, \mathrm{d}x \tag{3.21}$$

As they are irrelevant to $g(x|\theta)$ and as the second term for obvious reasons cannot exceed the first term, since that would result in the approximation resembling the function more than the function itself. Since the first term must be positive and has no parameters, the only way to minimise 3.21 is to minimise the second term.

In KLIEP as well as RuLSIF the density-ratio, $g(x|\theta)$, is modelled using a kernel model:

$$g(x^A|\theta) = \sum_{i=1}^{N} \theta_i K(x^A, x_i^B) \tag{3.22}$$

Where N is the number of samples in an analysis window and where $\theta = (\theta_1, \ldots, \theta_N)^T$, are the parameters to be learned via the mentioned optimisation problem 3.21. For the kernel basis function the Gaussian kernel is applied:

$$K(x^A, x^B) = \exp\left(-\frac{||x^A - x^B||^2}{2\sigma^2}\right) \tag{3.23}$$

Where σ is the kernel width trained for each density-ratio individually, see section 3.2.3.2 for more details.

Inserting 3.23 into 3.21 yields a convex optimisation problem which in the case of KLIEP lacks a closed form solution and therefore requires an iterative approach [75]. This optimisation yields the estimate of the density-ratio, $\hat{g}(x|\hat{\theta})$.

Finally given this estimate of the density ratio, the estimate of the KL-divergence can be approximated as the arithmetic mean of the log density ratio [59]:

$$\hat{\mathrm{KL}} = \frac{1}{n} \sum_{i=1}^{n} \log_{10} \hat{g}(x|\hat{\theta}) \tag{3.24}$$

Where n is the number of samples in an analysis window.

3.2.3.1.1 Outlier sensitivity of KLIEP

The main reason KLIEP is not applied is its extreme sensitivity to outliers which for lack of a referenceable source will be fully deduced in this section.

Given two PDFs, $p(x)$ and $q(x)$, the KL-divergence is given as 3.12:

$$\begin{aligned} \mathrm{KL}_{\mathrm{dvg}}(p;q) &= \int p(x)\log_{10}\left(\frac{p(x)}{q(x)}\right)\mathrm{d}x && (3.25) \\ &= \int p(x)(log_{10}p(x) - log_{10}q(x))\,\mathrm{d}x && (3.26) \end{aligned}$$

Now setting $q(x)$ equal to $p(x)$ plus an error term $\delta(x)$:

$$q(x) = p(x) + \delta(x) \tag{3.27}$$

Where the error term $\delta(x) \ll p(x)$, yields:

$$\mathrm{KL}_{\mathrm{dvg}}(p;q) = \int p(x)(log_{10}p(x) - log_{10}(q(x) + \delta(x)))\,\mathrm{d}x \tag{3.28}$$

Performing a Taylor series expansion [1] of the term containing the error:

$$\begin{aligned} log_{10}(q(x)+\delta(x)) &= f(y) && (3.29) \\ &= f(a+\delta(x)) && (3.30) \\ &= \sum_{n=0}^{\infty}\frac{f^{(n)}(a)}{n!}(y-a)^n && (3.31) \\ &= \sum_{n=0}^{\infty}\frac{f^{(n)}(a)}{n!}(\delta(x))^n && (3.32) \\ &= \sum_{n=0}^{\infty}\frac{(log_{10}p(x))^{(n)}}{n!}\delta(x)^n && (3.33) \\ &= \frac{log_{10}p(x)}{1}\delta(x)^0 + \frac{\frac{1}{p(x)}}{1}\delta(x)^1 + \frac{-\frac{1}{p(x)^2}}{2}\delta(x)^2 + \dots && (3.34) \\ &= log_{10}p(x) + \frac{\delta(x)}{p(x)} + \left(-\frac{1}{2}\frac{\delta(x)^2}{p(x)^2}\right) + \dots && (3.35) \end{aligned}$$

Reveals an interesting fact:

$$\mathrm{KL}_{\mathrm{dvg}}(p;q) = \int p(x)\left(log_{10}p(x) - log_{10}p(x) + \frac{\delta(x)}{p(x)} + \left(-\frac{1}{2}\frac{\delta(x)^2}{p(x)^2}\right) + \dots\right) \mathrm{d}x \tag{3.36}$$

$$= \int \left(-\delta(x) - \left(-\frac{1}{2}\frac{\delta(x)^2}{p(x)}\right) - \dots\right) \mathrm{d}x \tag{3.37}$$

Now since $p(x)$ and $q(x)$ are PDFs they are constrained to an integral of 1, which means that 3.27 integrated over all x reduces to:

$$\int q(x)\,\mathrm{d}x = \int p(x)\,\mathrm{d}x + \int \delta(x)\,\mathrm{d}x \tag{3.38}$$

$$1 = 1 + \int \delta(x)\,\mathrm{d}x \tag{3.39}$$

$$0 = \int \delta(x)\,\mathrm{d}x \tag{3.40}$$

$$\tag{3.41}$$

Thereby the integral further reduces to:

$$\mathrm{KL}_{\mathrm{dvg}}(p;q) = \int -\left(-\frac{1}{2}\frac{\delta(x)^2}{p(x)}\right) - \dots\,\mathrm{d}x \tag{3.42}$$

Now ignoring all additional terms in the taylor expansion as the integral will be dominated by the second term, due to the $\frac{1}{n!}$ factor on the n^{th} term:

$$\mathrm{KL}_{\mathrm{dvg}}(p;q) \approx \frac{1}{2}\int \frac{\delta(x)^2}{p(x)}\,\mathrm{d}x \tag{3.43}$$

This result is similar to a squared loss, but as is evident, errors are not weighted uniformly, they are weighted by the probability density at the position of the error. This means, that as $p(x)$ goes towards zero, which as a PDF is it required to do since it must integrate to 1, the entropy required to compensate for the error term $\delta(x)$ grows towards infinity.

Q.E.D.

In essence this result means that a single outlier, like the ones discussed in section 2.1.3.2, can significantly skew the result, rendering KLIEP impractical.

3.2.3.1.2 RuLSIF

The RuLSIF method solves both the major problems of KLEIP in one stroke by using the Pearson-divergence, PE, instead of the KL-divergence. These problems being the outlier sensitivity, see section 3.2.3.1.1, and the lack of a closed form solution to its optimisation problem, see section 3.2.3.1. The PE-divergence is given by [95]:

$$\mathrm{PE}(A;B) = \frac{1}{2}\int P_B \left(\frac{P_A}{P_B} - 1\right)^2 \mathrm{d}x \tag{3.44}$$

Where A and B are the preceding and succeeding analysis windows, where P_A and P_B are the corresponding populations.

Both KL and PE -divergence are merely different versions of the more general f-divergence [5, 21]. The PE-divergence avoids the issue of outlier sensitivity, see section 3.2.3.1.1, by not being based on a log scale. More precisely the PE-divergence is not a measure of the entropy, or the information gain, the density-ratio model is instead fitted to the true density-ratio under the squared loss, i.e. the divergence is independent of $p(x)$.

Through a similar deduction to the one described in section 3.2.3.1, finding the optimal parameters for the kernel approximation is a matter of minimising the PE-divergence, using the regularization parameter mentioned in section 3.2.3.2, to avoid overfitting. This, as opposed, to the KLIEP case does yield a closed-form solution, this calculation is beyond the scope of this section, for more detail see [75].

Finally the 'R' in RuLSIF denotes a smoothing used to avoid the issue that the density-ratio can be unbounded, i.e. it can be infinite. This poses problems as the convergence rate is governed by the supremum norm or sup-norm of the true density-ratio function [75].

According to [102] the sup-norm does not allow unbounded functions, since it will fail to yield a norm or metric in a strict sense, it will only allow one to define a topology on the function space in question.

This smoothing is governed by the smoothing parameter, α_R, which simply defines a mixture-density, termed the alpha-relative density-ratio, r_α:

$$r_\alpha(x) = \frac{P_A}{\alpha_R P_A + (1 - \alpha_R)P_B} \tag{3.45}$$

Which is obviously bounded above by $\frac{1}{\alpha_R}$ when the density-ratio is unbounded, the unbounded case is now a subset of $r_\alpha|_{\alpha=0}$. It must be noted that symmetry is achieved in an identical fashion to the method used in section 3.2.2.3.

3.2.3.2 Limitations

The main limitation of RuLSIF is the need for parameter training using cross-validation for every individual density ratio between analysis windows. These parameters are the kernel width, σ, in the kernel model 3.22 and the regularization parameter, λ. This parameter training is very computationally expensive, see section 3.2.3.3, for the implications.

It must be noted that usually cross-validation is performed to decrease overfitting to a training data set and thereby to maintain generalisability. However in this case the training set is the entire data set consisting of the full analysis window. This means that the bias of the dataset is merely lessened.

3.2.3.3 RuLSIF Implementation

The implemented version of RuLSIF was adapted from the change-detection version of the RuLSIF distribution by Song Liu, lead author of [75]. The original implementation can be found on Prof. Sugiyama's webpage [74]. Several alterations to the code were implemented, mainly an extension from 1-dimensional change-detection to an D-dimensional paradigm using matrix notation. Source code can be found in appendix A.3.8.

It is left to further work, see section 5, to test RuLSIF performance depending on the smoothing parameter α_R. In this thesis α_R is left unaltered from the default setting of 0.1 as the performance gains of RuLSIF compared to plain uLSIF is evident [125]. The alpha parameter could not be added as a third cross-validation parameter as that would stretch already limited computing resources and might ramify the 1D peak detection algorithm described in section 2.3, as a constant width on the moving average threshold might not be sufficient. It should be noted that a heuristic in-exhaustive trial using different values of α_R

was naturally performed to verify its impact and to check if small alterations causes large performance variations, this was not the case.

The range of values for the kernel width, σ, and for the regularization parameter, λ, to be chosen during training had to be set. This was done empirically by feeding the algorithm a large range of values and subsequently zooming in on the range of chosen values. These values fell roughly in bell curves around the values of 1 times the median for σ and 10^{-4} on a log_{10} scale for λ respectively. The resolutions of the ranges was kept to the default setting, which is 5 and the ranges were truncated as close to the point where the values beyond were chosen in less than 10% of the trials. 10% was chosen arbitrarily to conserve the limited range of values inside the region of maximum likelihood. The range is scaled linearly for σ and logarithmically with base 10 for λ. In MATLAB notation:

$$\sigma : \text{median} \cdot (0.2 : 0.4 : 1.8) \tag{3.46}$$

$$\lambda : 10^{-6:1:-2} \tag{3.47}$$

As per default, the algorithm does a 5-fold cross validation of every combination. It then selects the optimal combination for a final run.

Circumventing this entire training procedure does reduce runtime, but also by about a factor of 4, which is the major drawback of this approach and is the reason it was discarded as the sole method, see section 4.1.3.2. However, the reduction in runtime was not deemed significant to pursue this line of inquiry and as such the ramifications on performance and the underlying theory is left to further work, see section 5.

3.3 Parameter optimisation techniques

This section will describe the method used for the parameter selection of the two threshold gains associated with change-detection, α_{cd} and false alarm compensation, α_{FAC}, respectively, see section 2.3 and 2.4 for further description of these parameters. In addition, the rationale behind the design, the theory behind the individual components and the repeatability of said method is also discussed.

The reason these two parameters in particular require special attention is the fact that they are very interdependent, i.e. either can compensate for the other.

For this reason an optimisation of one parameter followed by an optimisation of the other would undoubtedly result is a suboptimal solution where one parameter tries to do things the other would be more proficient at. This point is demonstrated experimentally in section 4.1.2.

It must be noted that only the F-measure, see section 3.4.1, is optimised. The algorithm does therefore not optimise for computational efficiency, which could be done given that a higher α_{cd}, see section 2.3 reduces the runtime of the FAC as fewer potential change-points makes it through the change detection to be screened for false alarms. This would however increase the average data used in the FAC step. However, since the complexity of the metrics do not scale linearly with the data size the picture is not clear.

Many advanced methods for parameter optimisation have been developed. The method used here will employ a non-linear optimisation technique, with steps taken to avoid local minima. This is achieved by first locating a rough estimate using a multi-step grid search.

3.3.1 Basic grid search approach

The de facto standard for performing parameter optimisation is the grid search paradigm. Given some boundaries and a mesh, this method is simply an exhaustive search of the parameter space for an optimal solution. Unfortunately, maintaining a sufficient resolution of the mesh along with a sufficient amount of change-points to keep the Signal-to-Noise Ratio SNR high enough stretches the available computing power beyond their feasible limits. The simple brute force approach for parameter training of even a single metric takes on the order of a day or of weeks of computing time on a high end gaming system, depending on the metric used.

The obvious solution to this issue was an implementation on the IMM-Cluster, this would seem ideal as a grid search is perfectly parallelizable, this should reduce the complexity by up to a factor of 16, the number of grid terminals available. This however required about a weeks worth of redesigning the scripts involved. This approach was discontinued when another project with higher priority kept shutting down threads. The approach was simply too unreliable, too impractical and required too much data recombination.

The approach and the associated code was therefore scrapped and a novel approach was designed. The trade-off remains between a sufficient resolution of the mesh and a sufficient amount of data to achieve a reliable SNR.

3.3.2 Novel method design

Instead of the basic brute force approach to grid search, an iteration based approach is implemented, see appendix A.2 for the source code. This iteration based approach is used as an initialisation step and is followed by a non-linear optimisation technique called the 'Nelder-Mead method', that part is described in section 3.3.4.

The concept is based on the observation that the underlying cost function has a uni-modal envelope, possibly with smaller peaks on the border regions where one of the parameters is ignored, and a multi-modal fine structure. This is also evident in the results, see section 4.1.2.

The algorithm therefore starts out with a relatively fine grid and a low amount of change-points per parameter combination in order to do a grid search on the envelope of the cost function. This process is then repeated using a new grid centred on the location found in the previous iteration. The new grid's size is reduced by half in both dimensions and the resolution is lowered, whereas the amount of change-points per parameter combination is increased in order to progressively increase the SNR of the fine structure. In this fashion only few precise calculations have to be made. The parameters of each iteration are designed to results in an equal runtime per iteration by maintaining a constant total amount of change-points.

In practice 3 iteration of this algorithm are run, thereby reducing the area of the search space by a factor of 16 before the most precise measurements are made.

This iteration based grid search, is followed by the Nelder-Mead method, see section 3.3.4.

3.3.3 Location of grid boundaries

The use of grid search obviously requires the design of a grid. For simplicity the grid used in each iteration is regular, uniform and is rectangular in proportion to the size of the original grid.

During each iteration the range of values in each dimension is halved. This only leaves the original boundaries as free parameters to be chosen. Fortunately this is a simple task and is performed by heuristically locating upper and lower values that are definitely outside the suitable range. The reader is reminded that these values refer to the gain on the acceptance threshold, i.e. the values for which each threshold accepts and rejects everything respectively.

It must be noted that only the original boundaries are respected when drawing a new grid.

3.3.4 The Nelder-Mead method

Once the grid search method finds a suitable combination of initial values for the threshold gains the non-linear optimisation technique called the Nelder-Mead method, or downhill simplex method, is applied to locate the nearby local minima of the cost function. Note that the cost function is minimised as minimisation in optimisation techniques is the standard procedure; the cost-function is merely one minus the F-measure.

This section will not go into great detail on the background theory of this method. It will however discuss how and why it applies.

The method is part of the class of optimisation methods known as "Direct Search" [72]. Unlike more traditional optimisation methods that use information about the gradient or higher derivatives to search for an optimal point, a direct search method merely searches for points around the current point, looking for values lower than the value at the current point. For this reason a direct search method is ideal as the notion of a smooth cost function might be a fantasy, direct search methods do not require derivatives and are robust for problems with discontinuities or where the function values are affected by noise [15]. In this case noise merely refers to the variability of the F-measure given a random data sample according to the method described in section 2.1.5.

The Nelder-Mead method is also termed the 'downhill simplex method' as that is exactly what it does. A simplex is a polytope of D+1 vertices in D dimensions. Or in other words a 1-simplex is line segment on a line, a 2-simplex is a triangle on a plane, a 3-simplex is a tetrahedron in 3-dimensions and so on. What it does is continually try to fit smaller versions of this simplex down the local minima of the cost function, through a series of deformations of the simplex.

The stopping criterion is arbitrarily set to 0.01 as the diameter of the simplex, i.e. the longest of the shortest paths between any pair of vertices. The rationale being that, according to [91], if the curvature is slight then the sampling variance of the estimates will be large and so there is no sense in finding the coordinates of the minimum very accurately. Whereas if the curvature is marked, then there is justification for pinning down the minimum more exactly. In other words, if the diameter is sufficiently small then it can be assumed that a local minima is reached and given that these are merely parameters trained on training data, then moving to the exact local minima is bound to equate to over-fitting.

The implemented version uses the MATLAB function **fminsearch**, which is based on the work by Lagarias et al. [70].

3.3.5 Repeatability

In any science having a result is useless without a notion of the error involved in the measurement of said result. This however presents a problem as the designed procedure is optimised to be barely feasible given the computing resources available. Since the resolution to which a result is gathered is set by the time required to calculate the result. This 'resolution' being a trade-off between the amount of data and the size of the mesh in the space of possibilities.

Fortunately the F-measure error is obviously inversely proportional to the amount of change-points each combination of parameters is evaluated on. Furthermore the most efficient algorithm is about two orders of magnitude quicker than the least efficient approach, see section 4.1.3.2. For this reason the smallest amount of change-points and the sparsest mesh which is applied to RuLSIF, see section 4.1.2, is applied here to the most efficient approach, DSD, repeatedly. In order to, in lieu of a standard error on results, at least, get an upper bound on the variability of chosen threshold gains associated with this procedure.

The procedure is repeated 4 times and the results are seen in figure 3.2, 3.3, 3.4 and 3.5.

As seen the procedure is fairly repeatable finding roughly the same local maxima in all 4 cases, albeit by quite different paths, as is expected the positions of the intermediate local maxima vary. The result is promising and shows a robust and efficient parameter optimisation technique, even using low amounts of data.

The full procedure has a repeatability given by a standard deviation of 0.023 on the chosen α_{cd} and 0.0433 on the chosen α_{FAC} , or $\approx 5\%$ and $\approx 10\%$ of the valid ranges respectively. As a worst case scenario this variability is deemed very acceptable. The repeatability is expected to be much greater on the most precise runs as the used data is increased fivefold, see section 4.1.2.

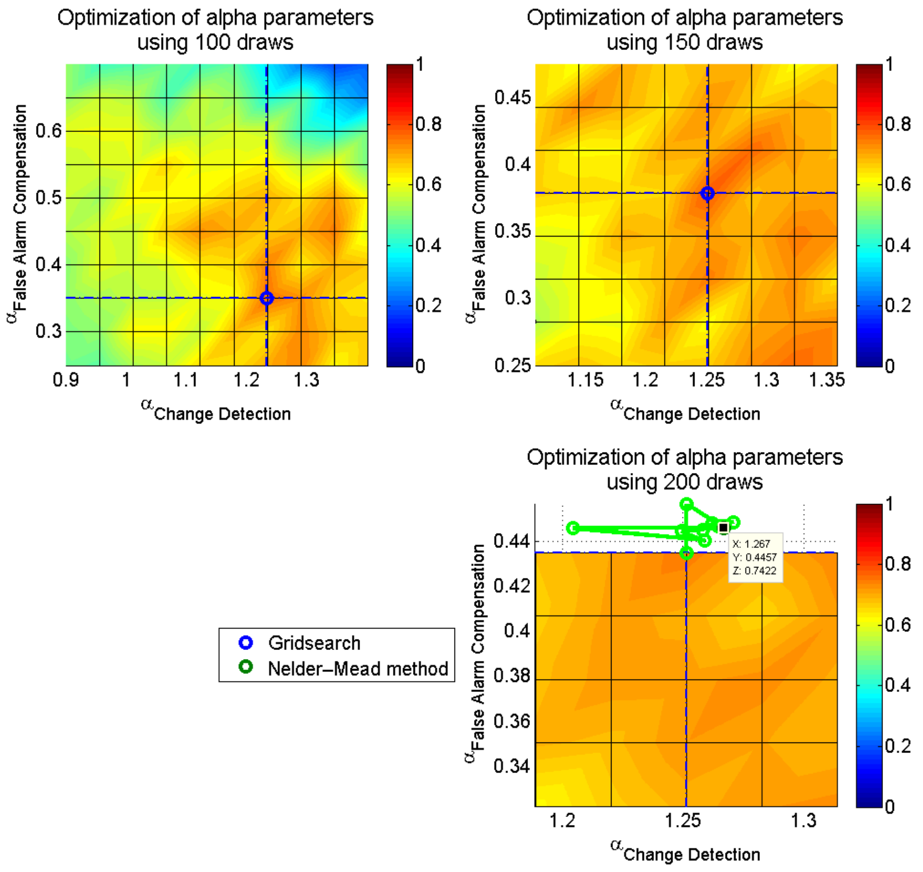

Figure 3.2: The colours - which are a linear interpolation of the values of the grid - represent the F-measure for the corresponding threshold gain combination. Note that the grid is independent of the axes. This figure displays the first run of the repeatability test for the custom parameter optimisation process described in section 3.3. The repeatability test and its results are discussed in section 3.3.5. Computing resources were dedicated to run this test 4 times in total, the results are seen in this figure, along with figures 3.3, 3.4 and 3.5. The amount of data at each grid point and for each iteration of the direct search method is displayed below the title of the figure. The number of draws refers to the data generation method described in section 2.1.5, is 1 higher than the number of change-points ad corresponds to about 10 seconds of speech per draw.
In this particular run the algorithm chose to favour FAC slightly above the norm and change detection slightly below the norm; this yielded a training result slightly below the norm.

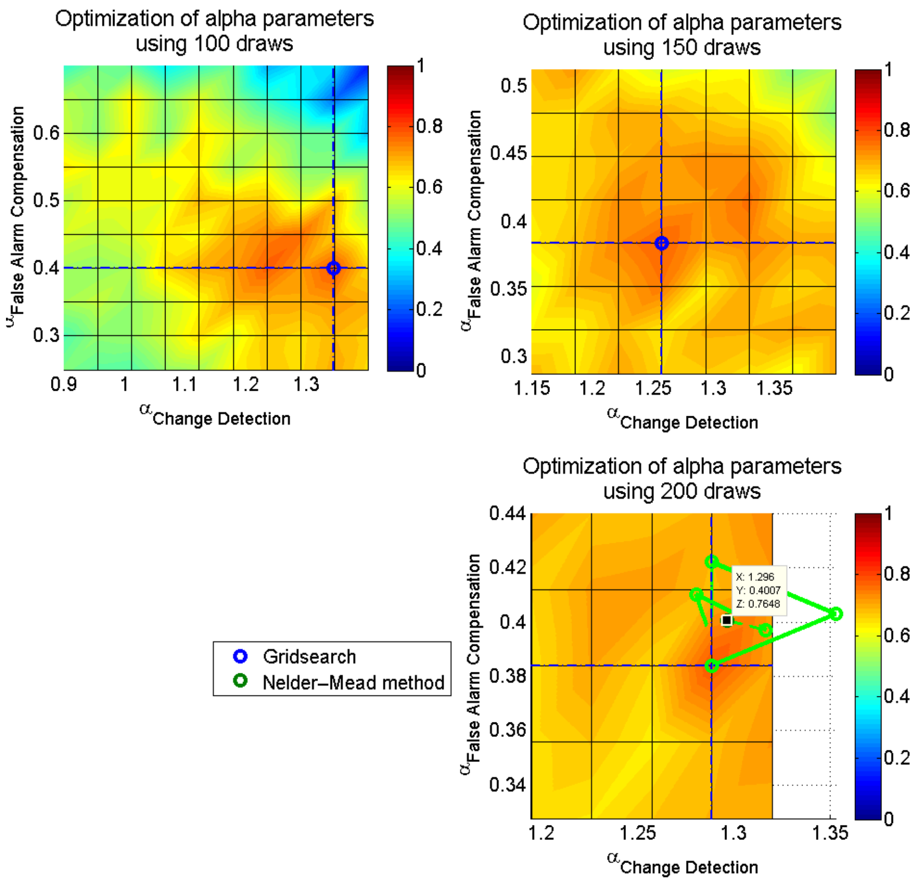

Figure 3.3: The colours - which are a linear interpolation of the values of the grid - represent the F-measure for the corresponding threshold gain combination. Note that the grid is independent of the axes. This figure displays the first run of the repeatability test for the custom parameter optimisation process described in section 3.3. The repeatability test and its results are discussed in section 3.3.5. Computing resources were dedicated to run this test 4 times in total, the results are seen in this figure, along with figures 3.2, 3.4 and 3.5. The amount of data at each grid point and for each iteration of the direct search method is displayed below the title of the figure. The number of draws refers to the data generation method described in section 2.1.5, is 1 higher than the number of change-points ad corresponds to about 10 seconds of speech per draw.
This particular run falls very close to the norm on all scores; FAC threshold, change detection threshold and the F-measure at the local peak.

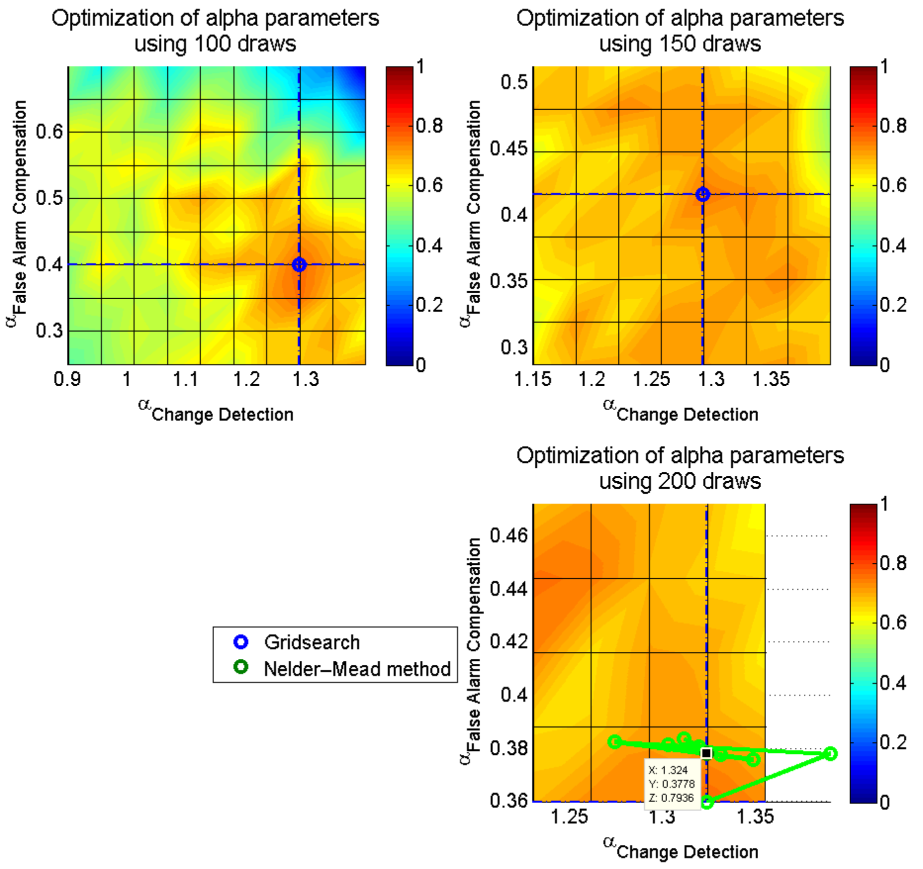

Figure 3.4: The colours - which are a linear interpolation of the values of the grid - represent the F-measure for the corresponding threshold gain combination. Note that the grid is independent of the axes. This figure displays the first run of the repeatability test for the custom parameter optimisation process described in section 3.3. The repeatability test and its results are discussed in section 3.3.5. Computing resources were dedicated to run this test 4 times in total, the results are seen in this figure, along with figures 3.2, 3.3 and 3.5. The amount of data at each grid point and for each iteration of the direct search method is displayed below the title of the figure. The number of draws refers to the data generation method described in section 2.1.5, is 1 higher than the number of change-points ad corresponds to about 10 seconds of speech per draw.
This particular run found the largest F-measure of the batch. The choice fell on a fairly high change-detection threshold and a correspondingly lower FAC threshold.

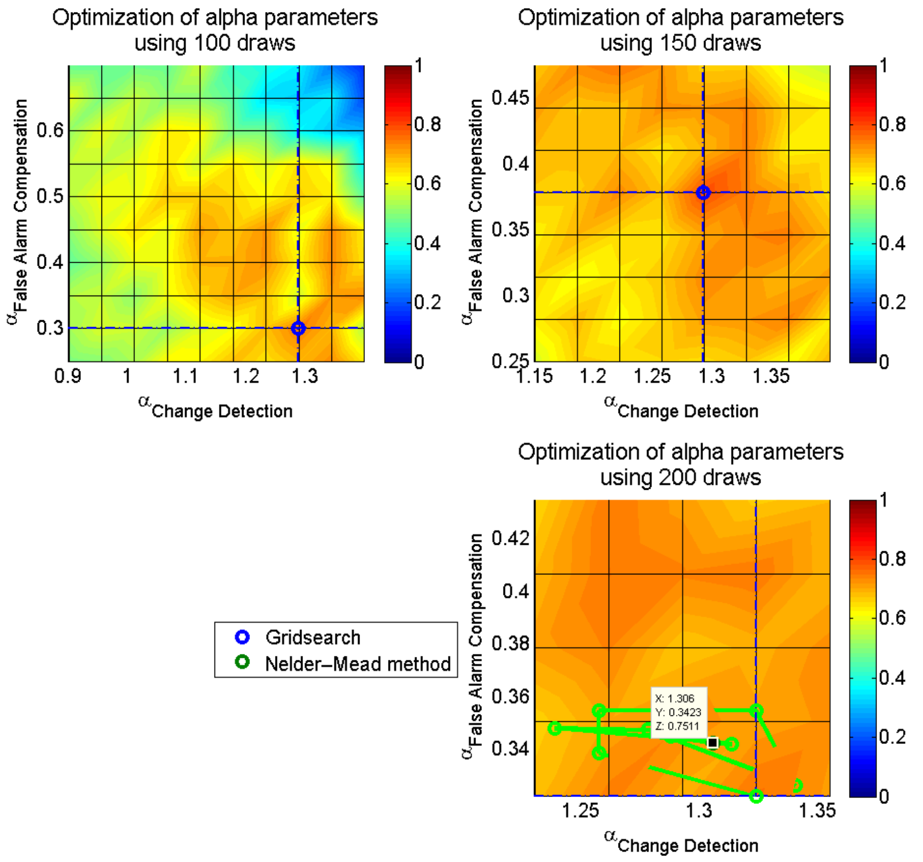

Figure 3.5: The colours - which are a linear interpolation of the values of the grid - represent the F-measure for the corresponding threshold gain combination. Note that the grid is independent of the axes. This figure displays the first run of the repeatability test for the custom parameter optimisation process described in section 3.3. The repeatability test and its results are discussed in section 3.3.5. Computing resources were dedicated to run this test 4 times in total, the results are seen in this figure, along with figures 3.2, 3.3 and 3.4. The amount of data at each grid point and for each iteration of the direct search method is displayed below the title of the figure. The number of draws refers to the data generation method described in section 2.1.5, is 1 higher than the number of change-points ad corresponds to about 10 seconds of speech per draw.
This run of the algorithm found an F-measure close to the norm, using a change detection threshold close to the norm, but surprisingly using a low FAC threshold.

3.4 Miscellaneous

Essential concepts not directly connected to other sections are relegated to this section. These include the optimisation parameter from section 3.3, the F-measure, which this section will start off by explaining. Subsequently the notion of the Standard Error of the Means, SEM, which is seen as the errorbars in every graph will be delineated.

3.4.1 F-measure and the confusion matrix

Most results in this thesis will primarily be presented in the form of an F-measure, also know as the F_1 score. In addition the optimisation from section 3.3 used in section 4.1.2 is designed to maximise this F-measure. This section will explain this choice and describe the F-measure.

Two types of error are possible in this thesis' approach to SCD. A missed change-point and inferring a false alarm. The evaluation measures used to measure these are recall, RCL, 3.48 and precision, PRC, 3.49. The F-measure is the harmonic mean between RCL and PRC, see equation 3.50. The RCL and the PRC are condensations of the confusion matrix, see figure 3.6. By this process the confusion matrix is expressed by a single number between 0 and 1.

$$\mathrm{RCL} = \frac{\text{no. of hits}}{\text{no. of targets}} = \frac{\mathrm{tp}}{\mathrm{tp}+\mathrm{fn}} \tag{3.48}$$

$$\mathrm{PRC} = \frac{\text{no. of hits}}{\text{no. of tries}} = \frac{\mathrm{tp}}{\mathrm{tp}+\mathrm{fp}} \tag{3.49}$$

Are combined into:

$$F = \frac{\mathrm{RCL} \times \mathrm{PRC}}{\alpha_F \times \mathrm{RCL} + (1 - \alpha_F \times \mathrm{PRC})} \tag{3.50}$$

Where α_F is a parameter used to tweak the cost function between the error types. In this thesis α_F is kept at an equal weighting:

$$\alpha_F = \frac{1}{2} \tag{3.51}$$

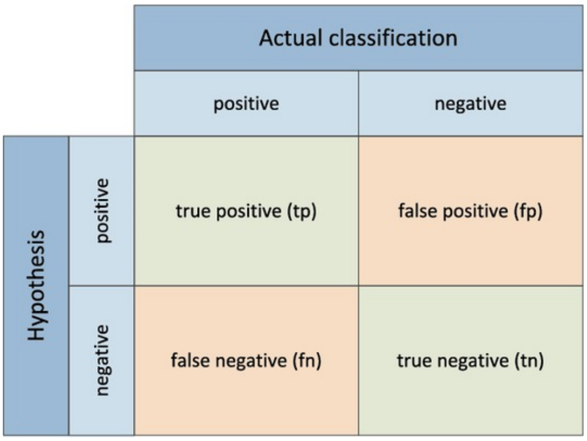

Figure 3.6: The confusion matrix. A graphical representation of the possible outcomes of any binary choice. For practical reasons the confusion matrix, in the context of this thesis, is distilled into the precision 3.49, the recall or sensitivity 3.48 and the specificity 3.56. The precision and recall are then further distilled into the F-measure. While the sensitivity and the specificity are used to generate ROC curves.

This is done in order to maximise the F-measure, the option of tweaking the cost between errors is included in the code as some practical system may have a higher intolerance for either error type.

The primary advantage of the F-measure for this thesis, beyond the convenient simplification of the confusion matrix, is that it circumvents the need for a known number of true negatives, tn, which in other words is the correct absence of check. In the context of this thesis' approach to SCD, correct absence of check is not applicable, see figure 1.1.

A change-point proposed by the algorithm may not be precisely aligned with the actual change-point. To take this into account, a found change is counted as correct if it is within 1 second of the correct change-point as defined by the T_i parameter, see section 2.3.

The implementation of the F-measure is found in appendix A.6.1 and in the context of FAC in appendix A.4.3.

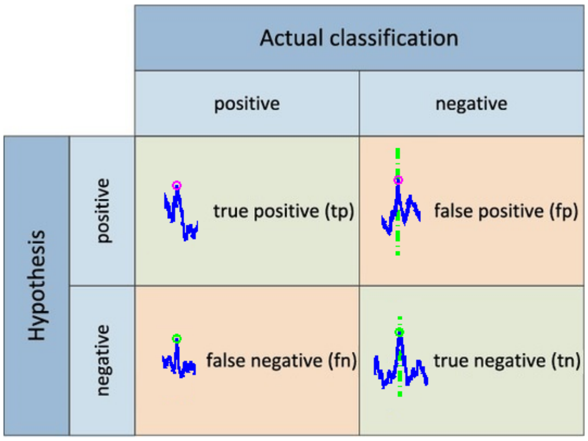

Figure 3.7: This figure elaborates on how the confusion matrix is applied in the context of FAC and in extension how PRC, RCL and F-measure fit. The different cases show how the concepts of tp, fp, fn and tn apply to FAC choices. As seen from the color the examples are taken from the K-means method, the blue curve therefore represents the VQD metric. The green dotted line represents a true change-point, with the absence denoting a non-change-point position in time. The magenta circles represent change-points marked as false alarms (a positive hypothesis) and the green circles denoting that the FAC algorithm rejected it as a false alarm (a negative hypothesis).

3.4.1.1 Interpretation of FAC performance

Both proofreaders found that the calculation of FAC performance was obscure, this section will try to shed light on this issue. Figure 3.7 shows the confusion matrix and how the corresponding FAC cases fit into it.

Furthermore, when the various result sections conclude that FAC is used aggressively what is meant is that if:

$$\mathrm{RCL} > \mathrm{PRC} \tag{3.52}$$

$$\frac{\mathrm{tp}}{\mathrm{tp}+\mathrm{fn}} > \frac{\mathrm{tp}}{\mathrm{tp}+\mathrm{fp}} \tag{3.53}$$

$$\mathrm{fn} < \mathrm{fp} \tag{3.54}$$

Then the algorithm favours positive hypotheses over negative hypotheses, an aggressive strategy.

It should be noted that upon reflection a better measure for FAC performance would have been the accuracy given by:

$$\text{Accuracy} = \frac{\text{tp} + \text{tn}}{\text{tp} + \text{tn} + \text{fp} + \text{fn}} \tag{3.55}$$

Since it would take the true negatives into consideration. This however, was discovered a matter of hours before the deadline and would require a major revision of chapter 4, it therefore did not make it into the thesis. This is not a huge concern as the optimisation algorithm, see section 3.3, optimises for general performance not FAC performance. Using the accuracy could have revealed some interesting design choices made by the optimization algorithm though.

3.4.1.2 Receiver Operator Characteristics

In section 4.1.1.1 some related measures are applied. In the context of ROC curves the false negative score is unobtainable, for this reason the PRC is invalid. There the sensitivity is another name for Recall 3.48 and its counterpart called specificity 3.56 is the true negative rate.

$$\text{specificity} = \frac{\text{tn}}{\text{tn} + \text{fp}} \tag{3.56}$$

3.4.2 Standard Error of the Mean

The repeatability of results are commonly measured using the Standard Error of the Mean , SEM, in lieu of a more rigorous approach using maximum likelihood or confidence intervals. Most results in this thesis use the SEM, while confidence intervals are estimated for a few.

Most results in this thesis are based on parameters that affect the precision of said results, mostly in the form of the amount of data averaged over, using the method described in section 2.1.5. Unintuitive the tests cannot simply be run on as much data as the computation resources allow, as this would preclude any information on the precision of the result, beyond the knowledge that it is as precise as possible.

To get a measure of the precision of a results; the result which itself is the average over results from data will have to be repeatedly gathered.

This is where the notion of SEM enters the picture. To avoid running several trials of the series of experiments averaged to get a result, the average of a series of experiments is assumed to be normally distributed. Using this assumption, the series of experiments only has to be run once and the accuracy of the average can be approximated [27]:

$$\mathrm{SEM} = \frac{s}{\sqrt{N}} \tag{3.57}$$

Where s is the standard deviation of results calculated using equation 3.10 and N is the number of results obtained.

The SEM of results are provided in the form of error bars in every graph.

Chapter 4

Application

This chapter will describe and discuss how the various methods are compared, contrasted, combined and improved. This is done by first performing individual feature selection for the various methods using ROC curves based on the metrics, see figure 3.1. ROC curves are used to enable pre-training feature selection of the methods, as it circumvents the entire change-point detection system, see section 2.3.

Subsequently the remaining methods, along with a hybrid method, are trained using the novel optimisation technique, see section 3.3, the results of which are compared based on the F-measure, the thoroughness and the efficiency as applied on test data. Evaluation of the involved trade-offs are made and the KL method is shown to outperform the other methods. The KL method subsequently undergoes a final backward feature selection based on a model of the human speech production system, see section 2.2.4. This backwards feature selection is shown to result in a marked increase in SCD performance.

4.1 Feature selection and method comparison

This section starts with a pre-training comparison based on receiver operating characteristic, ROC, curves which are used to perform forward feature selection on the raw metrics without any FAC. In addition to this, the ROC curves are used to rule out DSD as a SCD metric due to its poor performance and they are used to determine a suitable codebook size for K-means, which is subsequently validated during training.

4.1.1 Pre-training method comparison

This section will describe how ROCs are generated and used as a precursor to model training for model choice elimination and feature selection.

The problem of comparing metrics is tricky as individual metrics may outperform each other depending on the choice of basic parameters such as the various time constants, described in section 2.3, but may also outperform each other depending on the feature set used. This section will compare the use of first and second order derivatives of the MFCCs for the various metrics, moving forward with separate feature sets onto the post-training method comparison, see section 4.1.3, as necessary. Once a final method is selected a backward feature selection will be performed, see section 4.3.1. As this project builds on the work of Jørgensen et al. [55], this section will use the basic parameters found to be optimal previously, see section 2.2.

In order to compare the different metrics a method is needed that can work without the entire change point selection mechanism, described in section 2.3, the reason being that this will eliminate a significant amount of the parameters simplifying the problem immensely.

Using this method it is unfortunately not possible to apply the FAC mechanism described in section 2.4, the reason being that the iterative approach is incompatible with the ROC method used here. Nonetheless the FAC mechanism only improves the results slightly, see section 4.1.2.2.3, and the rough estimate of model performance achieved using ROCs should suffice.

4.1.1.1 ROC generation

The method used to generate the ROCs was borrowed from the supervisor L. K. Hansen, see appendix A.6.4.

The method has been adapted slightly to improve the performance in the context of this thesis, mainly the application of a non-constant threshold, this however was achieved through manipulations of the input. The method in essence takes a metric and a list of equal length specifying when change-points occur as input. The change-points are defined as true if within T_i seconds, as described in section 2.3.

The algorithm subsequently varies the gain on the moving average of the metric in order to scan over the ROC. The moving average defines the threshold, similar to what is described in section 2.3.

Ideally this method would simply scan the threshold over the metric calculating the precision and the specificity for each iteration, enabling the use of FAC, see section 2.4, however this would pose some key issues. The primary issue being that the change-point detection, the 1-dimensional peak detection algorithm, borrowed from [55] has the T_i constraint on the minimum distance between proposed change-points, in other words it is biased towards not predicting an excess of change-points. This in essence would result in a ROC that does not converge towards a ratio of one, instead the specificity would converge to one whereas the precision would converge to some value between 0 and 1 depending on the size of T_i. This would not necessarily facilitate the comparison of different metrics.

To avoid this complication, at each iteration, the algorithm simply calculates the precision as the percentage of metric above the threshold within T_i seconds around each change-point and calculates the specificity in an identical manner for the parts of the metric not within T_i seconds of a change-point.

This method arguably is biased towards metrics with slow slopes around change-points, but is far more robust, does converge to a ratio of one and the steepness of the slopes on the proposed metrics as roughly similar. This method will naturally only evaluate the metrics abilities to model speaker change dynamics at the used analysis window length of 3 seconds, borrowed from [55], see section 2.3.

The more complex methods, K-means and RuLSIF, may be able to more closely follow the speaker change dynamics when given longer analysis windows, i.e. they would provide better performance in the context of FAC, see section 2.4.

In actuality this method is performed through sorting lists of indexes as an iterative based approach would be time consuming.

4.1.1.2 Results from ROC analysis

Figure 4.2 shows an example of the metric scores with the actual change-points mark as the region contained by the vertical lines. Using the method described in section 4.1.1.1 the ROC curves in figure 4.2 are generated, using no MFCC derivatives, using only 1^{st} and using both 1^{st} and 2^{nd} derivatives respectively. As all curves are smooth arcs the concept of Area Under the Curve, AUC, applies directly and is used to marginalise each curve to a single performance score, see Figure 4.1, here the different methods are compared to each other as a function of used derivatives.

As seen in figure 4.1, DSD consistently under-performs all other methods. In addition the only theoretical difference between KL and DSD is whether or not the mean of the normal distribution fitted to the data is discarded, which in the case of DSD it is, see section 3.2.2.4. For this reason DSD as a change-detection method is discarded.

Another interesting conclusion that can be drawn from figure 4.1, is that DSD and KL improve performance with the addition of derivatives as features, whereas K-means significantly decreases in performance. This K-means effect might be caused by the 'hub effect', discussed in section 4.1.1.3.3, a feature of the not fully understood curse of dimensionality [52]. How the size of K, number of centroids, affects the AUC is investigated in section 4.1.1.3.2, suffice to say that a suitable value for K is found to be 25 in section 4.1.2.1. The use of derivatives does not appear to affect RuLSIF performance greatly, for convinience and to facilitate the novel method described in section 2.4.1, the full feature set is used for RuLSIF.

Going forward with alpha training, see section 4.1.2, RuLSIF and KL will be trained using all features, that is 0^{th}, 1^{st} and 2^{nd} derivatives, whereas K-means is trained and designed to work on the smallest subset of the features to avoid the derivatives issue, see section 4.1.1.3.3.

It should be mentioned that the use of 3^{rd} derivatives and upwards are not tested for two reasons. First, the ISP toolbox [53] does not support this and second, no mentioning of the use of higher derivatives has been found in the literature. This however might be interesting to investigate as the slope of KL seems to indicate that it will outperform the other methods using higher derivatives or possibly just more features.

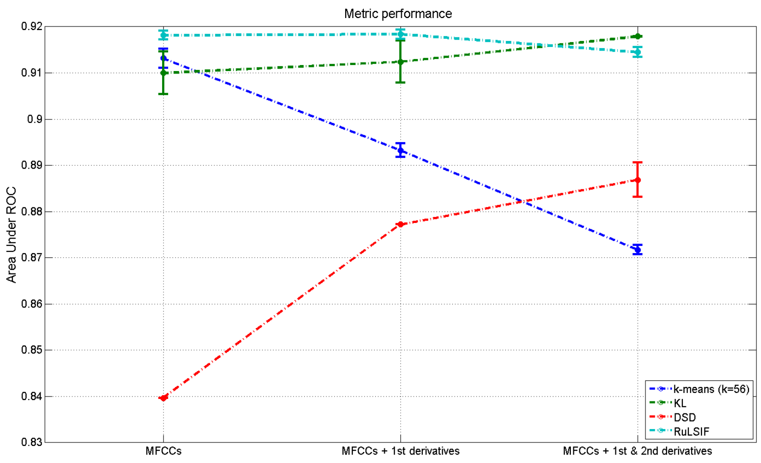

Figure 4.1: This figure displays the feature selection results from section 4.1.1.2. All four methods were run using 1500 change-points, corresponding to roughly 4 hours of speech, and every feature/method combination were run 2 times to verify the approximate repeatability. A ROC curve was then calculated for each point, see figures in 4.2, and marginalised to their AUC. The errorbars represent the standard error of the results.
As seen the RuLSIF method in terms of AUC barely changes depending on the feature set applied. For convenience and to enable use of the hybrid method between KL and RuLSIF, see section 2.4.1, RuLSIF will use the full feature set.
DSD and KL both show marked improvement using the larger dataset in spite of the dimensionality issue mention in section 3.2.2.2. Since DSD shows significantly worse results it is discarded from here on in.
The K-means method on the other hand shows a marked decrease in performance with the larger feature sets, this is investigated in detail in section 4.1.1.3.

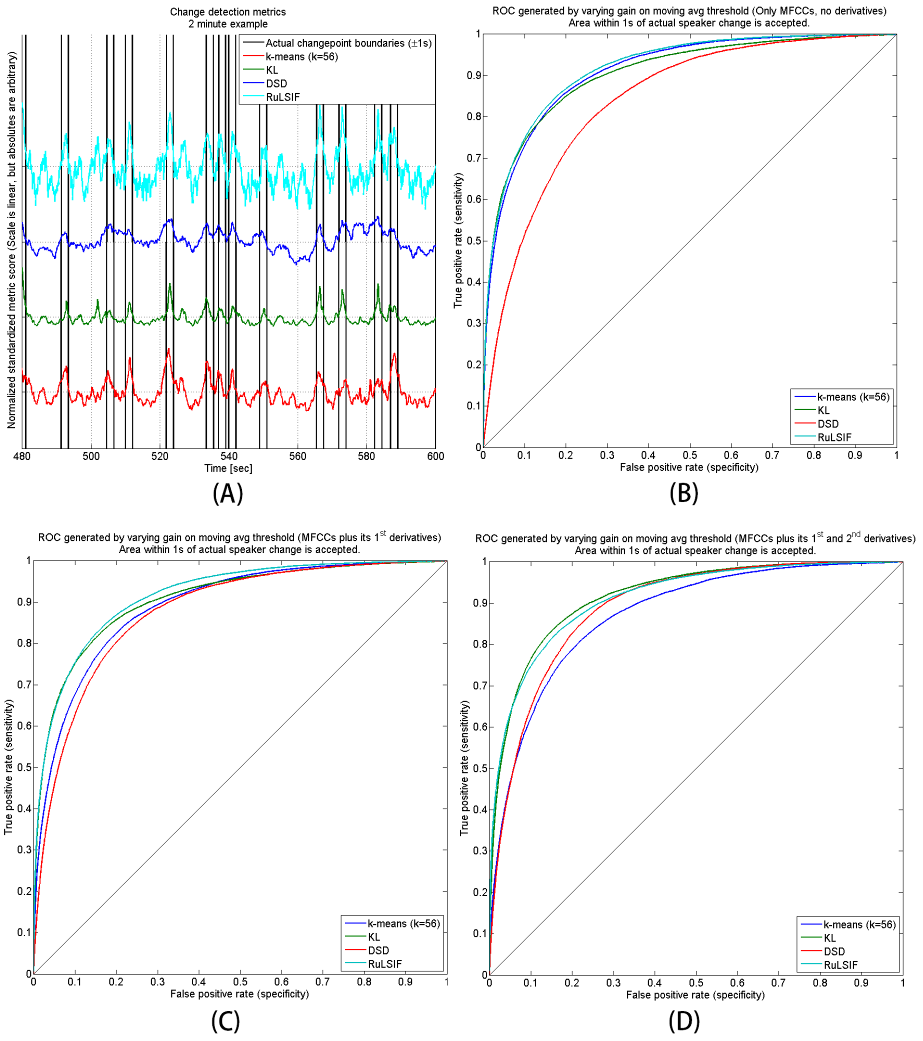

Figure 4.2: This figure serves as the example for the abstraction levels below the results seen in figure 4.1.
In (A) a 2 minute example of the about 4 hours of data for each metric in each trial is displayed for reference. Figures (B-D) are calculated by varying a moving threshold of each metric seen in (A) for the feature set displayed in the titles of (B-D), as described in section 4.1.1.1.

It should also be mentioned that the use of more features in the form of derivatives does increase the overall likelihood of over-fitting, see section 3.2.2.2.

4.1.1.3 K-means - Feature and codebook size selection

As mentioned in section 4.1.1.2, a trade-off between the dimensionality of the feature vector and the amount of centroids, K, must be explored for K-means to be an effective method. This section will explore this trade-off and make a suitable selection for use in alpha training, see section 4.1.2.

This section will scan over a range of K, using the ROC method described in section 4.1.1.1, for three choices of feature sets, these being without derivatives, with 1^{st} derivatives and with 1^{st} & 2^{nd} derivatives. Initially the values of K are spaces logarithmically in the suitable range, from 1 centroid to about a centroid for every 2 data points. Since a singleton approach is taken to empty clusters in the version of K-means applied here, see section 3.2.1.2, as K approaches the number of data points in an analysis window the K-means method begins to resemble some version of a nearest neighbour metric [8], such an approach could be taken, but is not investigated in this thesis.

Going into this section the hypothesis was that the deficiency of K-means with increasing derivatives, found in section 4.1.1.2, was attributed to some version of the 'hub problem', see section 4.1.1.3.3. In essence this would mean that a suitable choice of K should merely differ depending on the input feature space. However as seen in figure 4.3, the choice of K is not sufficient to account for the decrease in AUC. In effect this might still be caused by some dimensionality issue, however no means to compensate for it remains as the value of K is the only free parameter in the Euclidean distance based K-means algorithm, see section 3.2.1.2.

The deficiency could be caused by the optimisations described in section 3.2.1.4, this however has not been tested since the simulation would take significantly longer without these optimisations.

Another interesting point is that given a single centroid the K-means algorithm reduces to a geometric mean distance metric. This approach in combination with the DSD measure would resemble the KL distance, what is interesting to note is that DSD and K-means with K=1 both have an AUC of approximately 0.89. The fact that the KL measure has an AUC of about 0.92, means that they encode complimentary information, giving yet another reason for the invalidation of the DSD metric as a method for SCD. It should be noted that this work does not explore which parts of the auditory information is carried by the covariance and

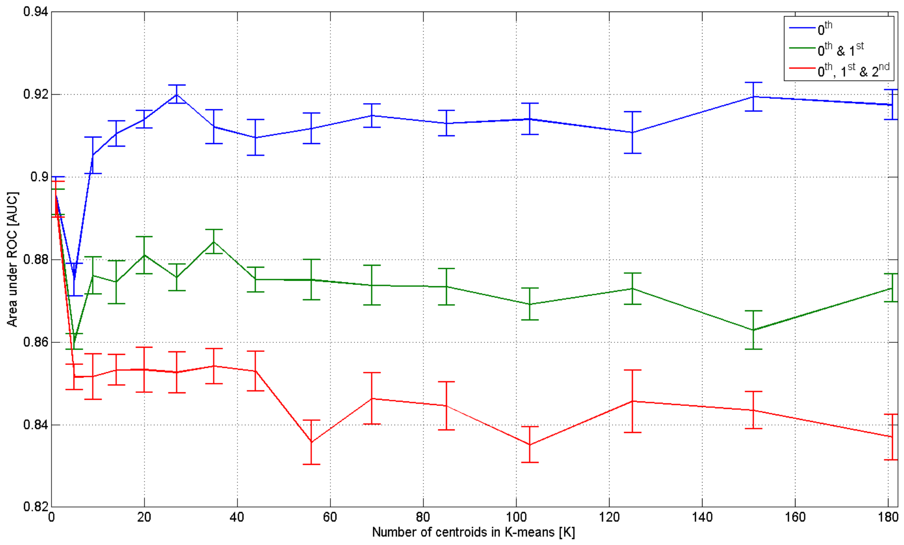

Figure 4.3: Performance of K-means as a function of codebook size for different sets of MFCCs. Each set is comprised of derivatives of the 12 MFCCs as described in section 2.2. Each combination is repeated 10 times using a random set of 100 change-points drawn as described in section 2.1.5. The error bars indicate the standard error, described in section 3.4.2, under the assumption that the mean is drawn from an underlying normal distribution. The performance is measured using the AUC measure calculated using the ROC method described in section 4.1.1.2. As seen the performance deteriorates with the addition of more derivatives irrespective of codebook size, this may merely be caused by the increasing dimensionality, as discussed in section 4.1.1.3.3. The peak at K=27 is shown in greater detail in figure 4.4.

which is carried by the mean, it might be of interest to discard one of them given that the other might base the SCD decision on more reliable information, see section 3.2.2.4.

4.1.1.3.1 Feature selection

Going forward with alpha training, see section 4.1.2, the K-means method will be trained without the use of the derivatives of the input features. The reason being that figure 4.3 shows a significant advantage irrespective of the number of centroids. Whether K-means is merely under-performing for higher dimensional spaces irrespective of the number of centroids would need further analysis. This further analysis could be in the form of a comparison between the performances, using various combinations of features, while keeping the absolute amount of features constant. This further analysis into an optimal feature set for this particular method, K-means, would have been examined using backwards feature selection if K-means had proved to be the most viable method. As seen in section 4.1.3, KL proves to be a superior method in terms of a performance-efficiency ratio, for this reason the backwards feature selection is performed on KL, see section 4.3.

4.1.1.3.2 Centroid amount selection

The work by Kinnunen et al. [61] concludes that identification rate increases with increasing codebook size. Figure 4.3 however seems to indicate a peak around K=27 using only the MFCCs. Using MFCC and 1^{st} derivatives does also indicate this peak. In order to investigate this further the simulation is rerun at higher precision between the values K=20 to K=35 using only the MFCCs, see figure 4.4.

Figure 4.4 obviously does not indicate any statistically significant advantage of one value of K over another K. To further illustrate this, figure 4.5 shows the results from the test run a second time.

The results here do not provide any reason to choose a specific value of K, beyond the need for it to be above about 10. Theoretical speculations in section 3.2.1.2.1 indicate that a higher value of K may be preferable. However efficiency results from section 4.1.3.2 indicate that K-means must be fairly optimised to be of practical use. For these reasons K-means will be trained using the fairly high value of K=56 chosen in [55] and the value of K=25, as figure 4.4 does indicate a peak. Whether the amount of centroids influence the performance can be seen in section 4.1.2.1, where K=25 is compared to K=56.

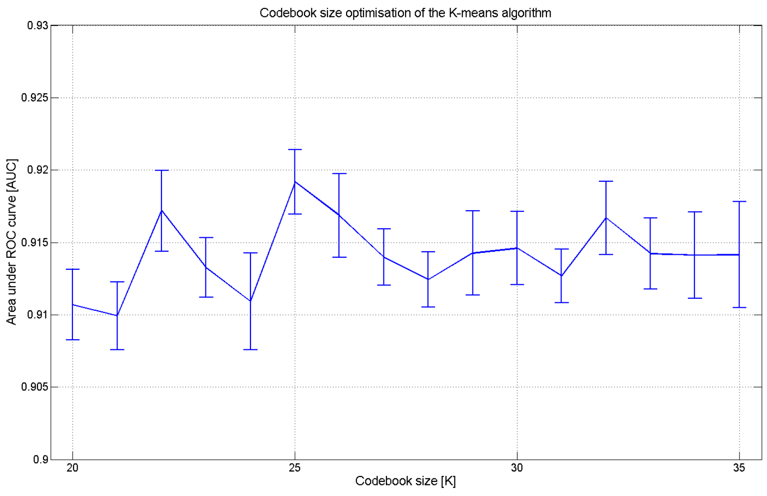

Figure 4.4: Figure 4.3 promted a more precise test around a codebook size of K=27, the results of which are shown here. The results here show that the peak around 27 was statistical noise. To further illustrate this point, this test is rerun in figure 4.5, showing a completely different results. The test here is run 10 times and the errorbars indicate the standard error.

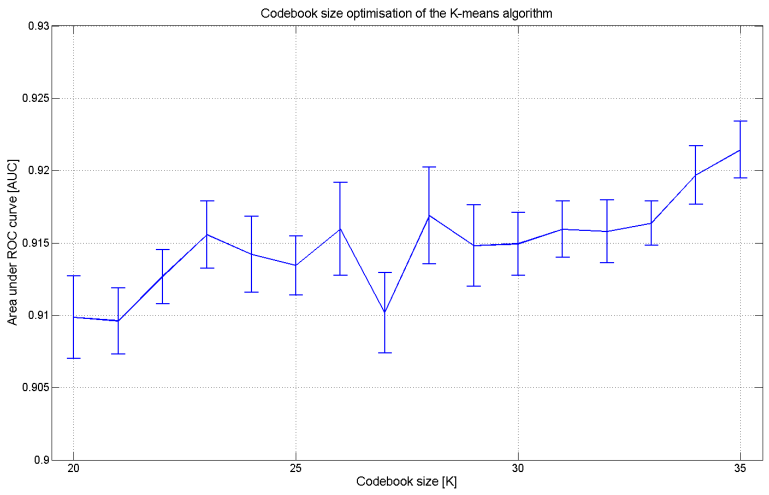

Figure 4.5: Figure 4.3 promted a more precise test around a codebook size of K=27. To test this the test was rerun around K=27, 10 times in 2 trials. This figure shows the result from the second trial, the first trial is seen in figure 4.4. The standard error is displayed using errorbars. As seen the results in the 2 trials do not resemble eachother and do not indicate any peak around K=27. The variations seen are probably just statistical noise. An upwards trend, as shown in [61], is still possible in the original and both re-trials. Going forward the effect of the codebook size is investigated using the full change-point detection and FAC paradigm to reduce the statistical noise, the results of which are seen in section 4.1.2.1.

4.1.1.3.3 K-means and high dimensional data

Two hypotheses can be drawn from K-means failure when using derivatives, seen in section 4.1.1.2, either the derivatives hold very little information, which the results from the other metrics refute, or K-means has issues that do not affect the other methods. The reason that K-means fails using high dimensional data might be linked to the higher dimensionality of the input space using derivatives, this is also discussed in section 4.1.1.2. If this is the case then one of the reasons posted in a survey by Zimek et al. [126] where they identified the following problems when searching for anomalies in high-dimensional data could be the cause:

1. **Concentration of scores and distances:** derived values such as distances become numerically similar.

2. **Irrelevant attributes:** in high dimensional data, a significant amount of attributes may be irrelevant.

3. **Definition of reference sets:** for local methods, reference sets are often nearest-neighbour based.

4. **Incomparable scores for different dimensionalities:** different subspaces produce incomparable scores.

5. **Interpretability of scores:** the scores often no longer convey a semantic. meaning

6. **Exponential search space:** the search space can no longer be systematically scanned.

7. **Data snooping bias:** given the large search space, for every desired significance an hypothesis can be found.

8. **Hubness:** certain objects occur more frequent in neighbour lists than others.

In particular the Hubness might be an issue with respect to K-means. Particularly as centroids would gravitate towards these meaningless - in the context of SCD - Hubs. In addition to this, point number 1 might pose problems for K-means as it is heavily based on distance measures. In this case an input space of higher dimensionality might render K-means more susceptible to noise, since making distances more similar would decrease the signal to noise ratio.

4.1.2 Method training results

Given that the ROC method - used to provide a rough comparison of the metrics, see section 4.1.1.2 - only succeeded in eliminating a single candidate. The full method is trained using K-means, KL and RuLSIF as contenders. In addition the results in section 4.1.1.2, prompted an idea for a novel FAC method, here termed CMFAC, the concept behind this novel approach is described in section 2.4.1. In this section CMFAC will be used to apply RuLSIF as FAC metric to KL, the result of which is compared to the pure methods that made it into threshold training.

The choice of codebook size for K-means proved ambiguous in section 4.1.1.3.2; therefore this section will start off by comparing two choices of codebook size, before moving on to compare K-means with KL, RuLSIF and the KL/RuLSIF hybrid. This codebook size comparison is on the basis of the F-measure at the local maxima of the cost-function that the individual codebook choices locate in the threshold parameter space. As well as on the relative magnitude between chosen thresholds for each method.

4.1.2.1 Compare K-means codebook sizes

The ROC method applied in section 4.1.1.3.2 was unable to select a codebook size for K-means, as it appeared to reach a maximum around K = 20 with further increase having no statistically significant impact.

Previous work found that larger codebook size should increase performance [55, 61] and the ROC method was deemed too coarse to discount the literature. For this reason the values of K=25 and K=56 were selected, K=25 was selected as a probably statistically insignificant peak was located there, see figure 4.4, and K=56 was chosen as that was the value chosen in [55].

This section will apply the full training method, described in section 3.3, to these two values and compare the results. The results of this process can be seen in figures 4.6 and 4.7 for K=25 and K=56 respectively.

As seen in figures 4.6 and 4.7, the end results is identical to 3 significant digits, both choosing an α_{cd} around 1.1 and an α_{FAC} around 0.91. The more expensive method using K=56, figure 4.7, chose to favor FAC slightly more, but not to a significant degree. The F-measures found for the different approaches is also effectively identical meaning that the approaches probably found the same local maxima, again lending credence to the hypothesis that the codebook size is irrelevant.

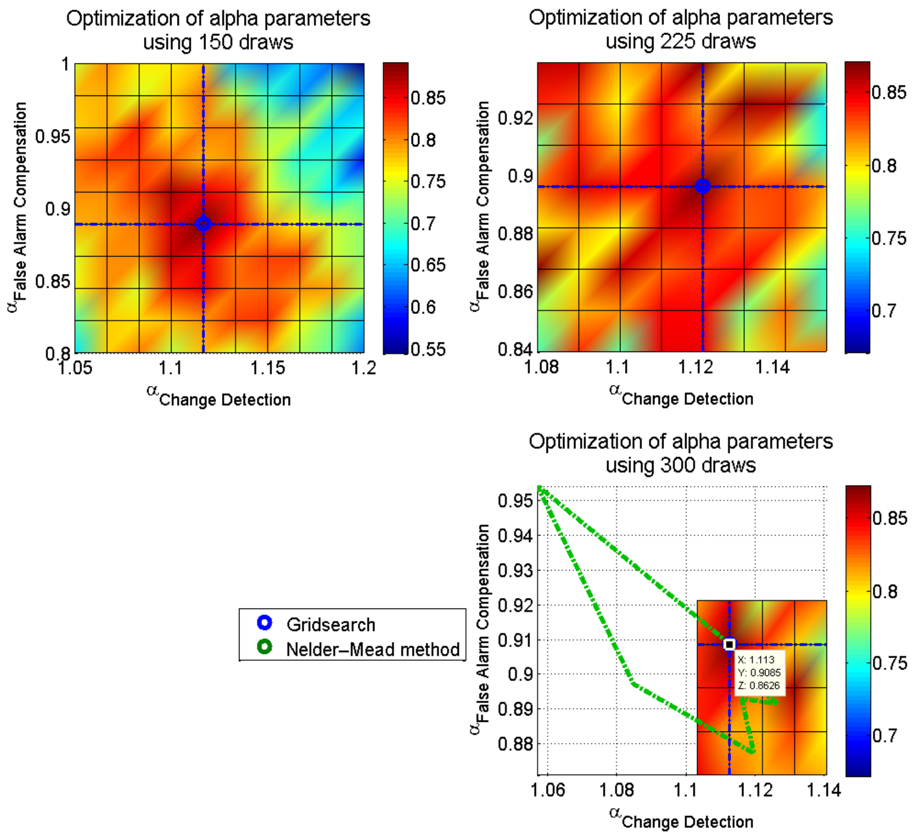

Figure 4.6: This figure shows the K-means algorithm trained using a codebook size of K=25, with the colours representing the F-measure for the corresponding threshold gain combination, the contours in between is a linear interpolation.

The ROC based method used in section 4.1.1.3 was unable to provide an unambiguous answer to the question on whether a larger codebook size in the K-means algorithm would yield better results. The results seemed to indicate that an increase beyond K=25 had no effect. This section tests this definitively by training the full K-means algorithm on both K=25 and K=56, see figure 4.7, using the optimisation method from section 3.3. As seen the results from the 2 trials using K=25 and K=56 show almost identical results. For this reason K=25 is used as it is more efficient.

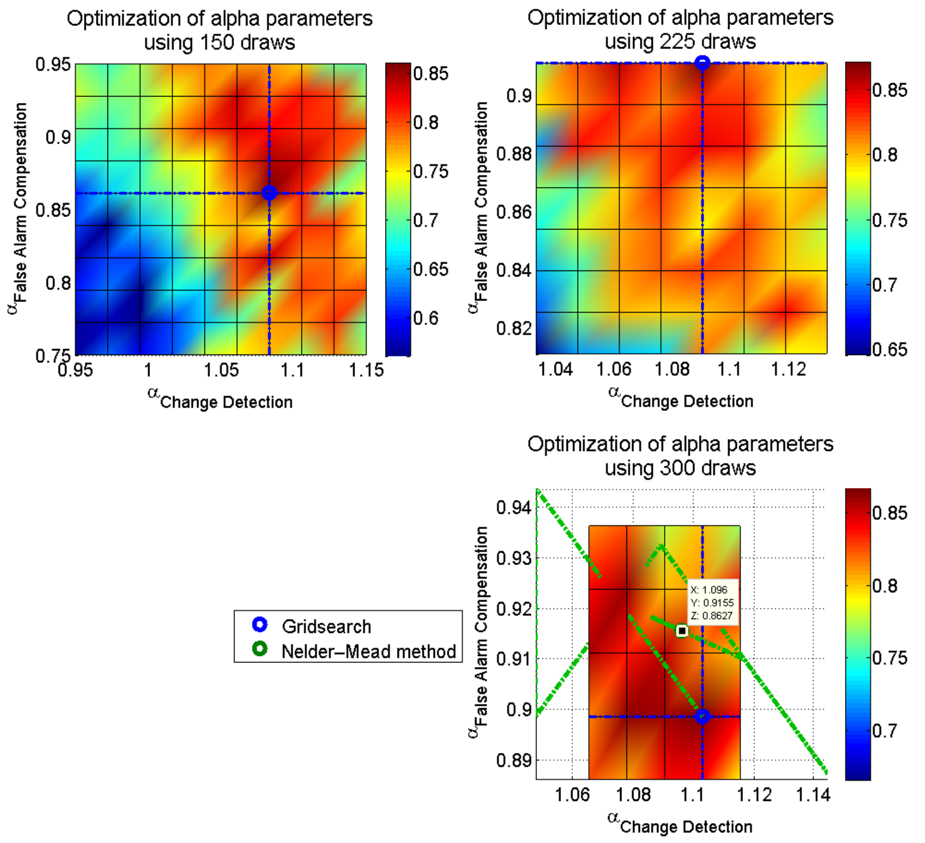

Figure 4.7: This figure shows the K-means algorithm trained using a codebook size of K=56, with the colours representing the F-measure for the corresponding threshold gain combination, the contours in between is a linear interpolation.
This figure is the second part in the K-means codebook selection test. See figure 4.6 or section 4.1.2.1 for further details.

Obviously to get an unbiased result the found parameters should be used to run a series of test on the test data. Effectively a hypothesis test is clearly needed to find any significant difference between the results using different codebook sizes. However, as the results are almost identical any difference would be insignificant compared to the additional computational cost of the larger codebook.

For this reason the series of tests on test data are not performed as this would be a waste of resources. The parameters found using the smaller codebook (K=25) are selected and compared to the other contenders, KL and RuLSIF in section 4.1.2.

4.1.2.2 Pre-test comparison

As the alpha training, see section 3.3, of the K-means algorithm was run in section 4.1.2.1, only the alpha training of the KL and RuLSIF metrics remain. The reader is reminded that DSD was discarded in section 4.1.1.2. This section will therefore begin by providing the results for the remaining metrics, then subsequently compare the 3 metrics on the basis of performance and efficiency using as of yet unused test data. In addition to this the combination of KL and RuLSIF using CMFAC, see section 2.4.1, is trained and compared to the pure methods using standard FAC, see section 2.4.

4.1.2.2.1 Threshold training

The alpha training results of the KL and the RuLSIF metrics are displayed in figure 4.8 and figure 4.9 respectively. The result for the KL/RuLSIF combination using CMFAC, see section 2.4.1, is shown in figure 4.10.

In figure 4.8 the result of the threshold training for the KL metric is displayed. As the KL metric is quite light computationally, see section 4.1.3.2, a large resolution along with a large amount of data per threshold combination is possible. What is interesting to note is that the strength of the grid search is very evident here. In figure 4.8 top left the overall envelope of the cost function appears to have two minima (or maxima in F-measure). In the second iteration using more data, top right, this however turns out to be random fluctuations as one simply turns out to be an upwards slope.

As seen in figure 4.9 the process did not completely follow the procedure described in section 3.3. As the RuLSIF method proved too computationally heavy to fully realise the process. Only relatively few change-points are used for each combination and the resolution of each iteration is set to a bare minimum. As

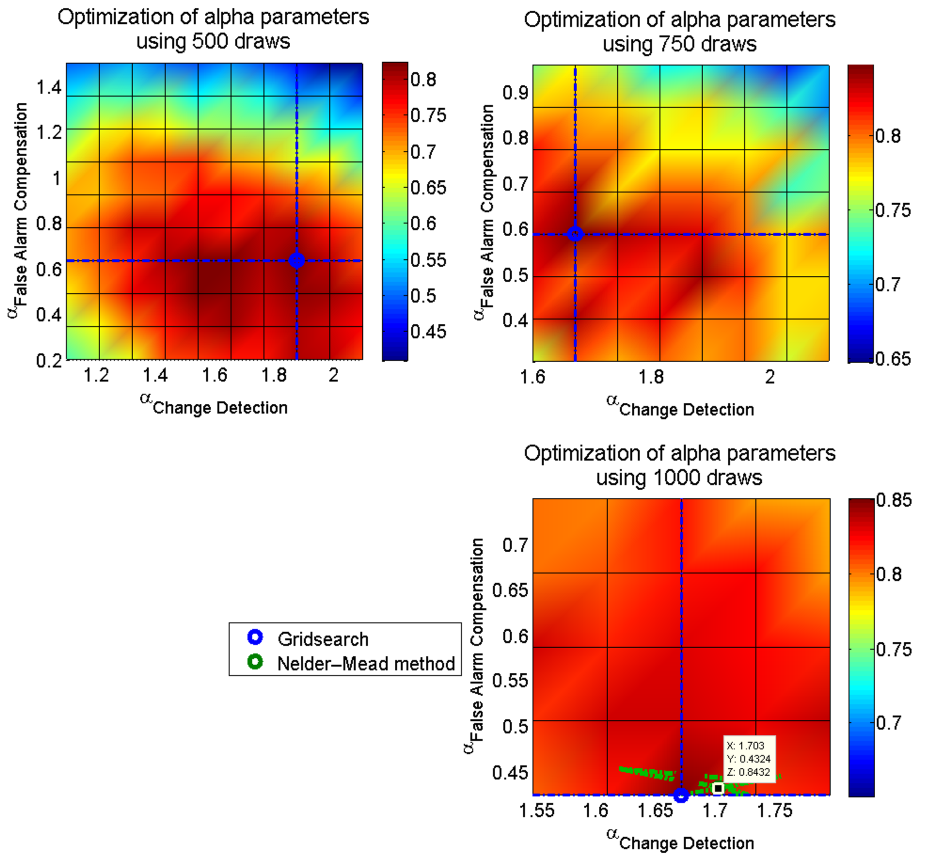

Figure 4.8: The colours - which are a linear interpolation of the values of the grid - represent the F-measure for the corresponding threshold gain combination. Note that the grid is independent of the axes. This figure displays the training steps and training result of the full KL method; using the optimisation method described in section 3.3, the results here are compared to the other methods on training data in table 4.1. The various methods are compared using test data in section 4.1.3.
As seen in this figure a large amount of data is drawn for each grid point, and the resulting objective function is very smooth. In the first iteration the grid method finds two peaks and selects one of them, in the second iteration using more data it realises that there really was only a single peak.

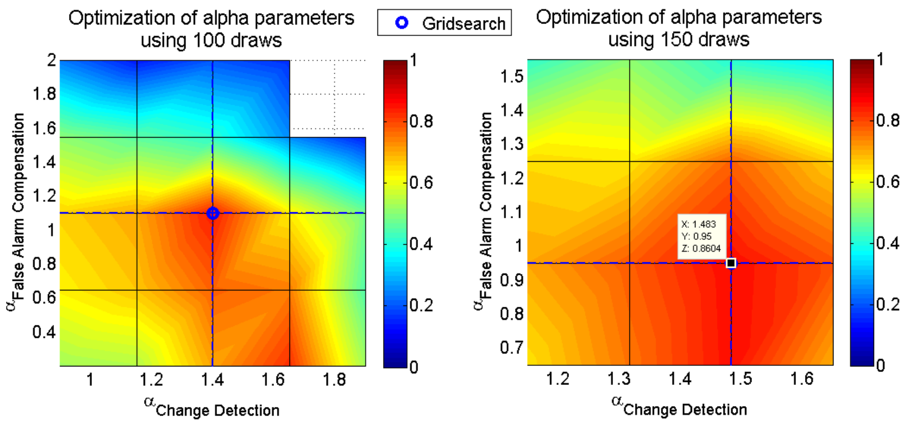

Figure 4.9: The colours - which are a linear interpolation of the values of the grid - represent the F-measure for the corresponding threshold gain combination. Note that the grid is independent of the axes. This figure displays the training steps and training result of the full RuLSIF method; using a slightly scaled down version of the optimisation method described in section 3.3. As seen the resolution is kept at a bare minimum and the data amount is quite low, yet still running the 3^{rd} iteration and the final direct search method proved too computationally expensive. The results here are compared to the other methods on training data in table 4.1. The various methods are compared using test data in section 4.1.3. As seen the envelope of the cost function appears relatively smooth and the maxima found in the first iteration is also located in the second iteration indicating that the algorithm is probably centering on a maxima and not just on noise.

such only the first step is fulfilled, namely locating the approximate maxima of the cost function envelope. The second step of locating a local maxima of the fine structure and the final step using a non-linear optimisation algorithm was skipped as this process would take on the order of weeks to finish. As seen the envelope of the cost function appears relatively smooth and the maxima found in the first iteration is also located in the second iteration indicating that the algorithm is probably centring on a maxima and not just on noise. The threshold gains could clearly be optimised further probably yielding an increase in F-measure of no more than about 0.01 – 0.02. RuLSIF might have benefited from a training session without FAC altogether, see section 2.4, thereby locating the precise peak of the local maxima seen in the figure on the left on the south border, as the computing power could then be concentrated on a 1-dimensional grid-search rather than a 2-dimensional one. This test was not performed as it would require unavailable computing resources. It should be mentioned that a FAC'less approach is provided for reference using KL in section 4.1.2.2.3, showing a fairly marked lose in performance.

It should be mentioned that the missing square, in the top right corner of figure 4.9, merely indicates that no change-points at all were found given that parameter combination.

The results for the CMFAC metric combination using KL and RuLSIF is shown in figure 4.10. As seen in section 4.1.3.2, the cost of applying the computationally heavy RuLSIF even as a FAC step significantly decreases the computational efficiency, for this reason the data applied for each threshold combination is set quite low to keep the resolution high. This trade-off was selected to make sure the global maxima was not simply too narrow to be missed and since the process yields fairly consistent results, as seen in the smoothness of the cost-function.

What is shown in this training session is quite interesting; the south border with a threshold gain of 2 on the CMFAC RuLSIF metric was found heuristically to be the point at which the CMFAC process had no effect at all, however as seen the grid search method iteratively approaches this boundary even selecting a value on it.

The obvious conclusion is that the CMFAC process failed completely, that the change-detection is clearly better off without it. Upon closer inspection this however is not the case, as the final local minima of the cost function yields an F-measure of comparable magnitude, if not larger, than the pure KL method in figure 4.8. Further, in section 4.1.2.2.3, the KL metric is trained without any FAC at all, this is shown to yield significantly poorer results confirming the necessity for the FAC step.

The next natural conclusion is that this lower border of 2 is too high, that given

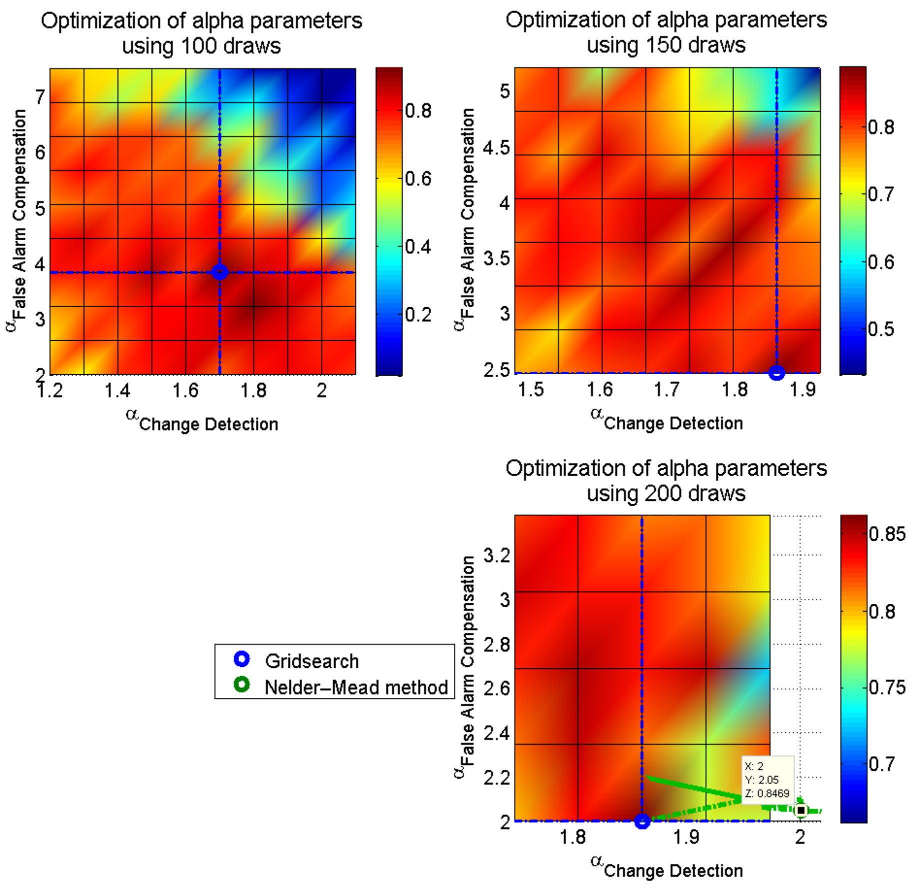

Figure 4.10: The colours - which are a linear interpolation of the values of the grid - represent the F-measure for the corresponding threshold gain combination. Note that the grid is independent of the axes. This figure displays the training steps and training result of the KL/RuLSIF hybrid method; see section 2.4.1, using the optimisation method described in section 3.3, the results here are compared to the other methods on training data in table 4.1. The various methods are compared using test data in section 4.1.3. As seen the added RuLSIF FAC step to the very efficient KL change-point detection took its toll and the training was completed using a fairly low amount of data per threshold gain combination.
Interestingly this figure shows that the hybrid method finds an optimal solution using a very high thresholds on both change-point detection and a very low threshold on FAC. Yet still manages to find a solution that seems as good, if not better than the pure KL method. This indicates that the CMFAC method using RuLSIF is very proficient at locating very difficult false alarms, but fails at locating false alarms that the pure method excels at.

	α_{cd}	α_{FAC}	F-measure
K-means	1.096	0.9155	0.8627
KL	1.703	0.4324	0.8432
RuLSIF	1.483	0.9500	0.8604
KL/RuLSIF	2.000	2.050	0.8469

Table 4.1: This table shows the final threshold parameters for the various methods along with their results as applied on training data. The intermediate stages, of the optimisation method, for the various methods can be seen in figures, 4.8 (KL), 4.9 (RuLSIF), 4.6 (K-means) and 4.10 (the KL/RuLSIF hybrid). These results seem to indicate that as expected the more expensive methods pull slightly ahead of the more efficient methods. The generalisability of these results are investigated in section 4.1.3.

the opportunity the algorithm would choose an even lower threshold gain on CMFAC. This step was rerun and a border of 2 was found to be very slightly too constricting, but was confirmed to be suitable. The suitability is also seen in the path of the Nelder-Mead method, which is not designed to be constrained by the boundaries of the grid search algorithm, yet decides to move slightly northward of the border.

4.1.2.2.2 Threshold training results

As seen in figures 4.6, 4.8, 4.9 and 4.10 the threshold gains and the F-measures found in the training session are as seen in table 4.1.

The training results, seen in table 4.1, indicates a fairly close run between the different approaches. The results indicate that the most expensive method, RuLSIF, is probably also the most precise given adequate computational resources for the training process, which were unavailable.

As it stands K-means and RuLSIF show fairly identical results edging slightly ahead of the computationally cheapest and simplest solution, KL, and the experimental KL/RuLSIF combination, which also yield similar results.

The results also show that the metrics differ in SNR, as is also evident in figure 3.1. This difference in SNR is seen in the ratio between α_{cd} and α_{FAC}. It is seen that K-means looks similar on the short time scale that the change-detection operates on to the long time scale of FAC. As is also seen in figure 3.1, KL shows a tendency to have a low noise floor and a large difference in the magnitude of the peaks, this is seen to affect its performance as the change-detection threshold favors a large value and the smoothing long time scale of FAC requires a much lower threshold. RuLSIF is somewhere in between these two extremes, but suffers from another unique disadvantage. For RuLSIF the noise floor is not independent of the speaker. This effect is seen visually in figure 3.1 and had impacts in section 4.1.1.1, where a moving average threshold had to be employed, since otherwise the RuLSIF method falsely indicated a poor performance.

The experimental KL/RuLSIF combination is oddly enough seen to favor a very restrictive change-detection step, apparently signifying that the CMFAC RuLSIF step can correctly filter out quite certain potential change-points as false alarms. It must be mentioned that K-means has an advantage given that all the time-constants, see section 2.3, used in this thesis were borrowed from the previous project [55], which found the K-means method superior to the KL and DSD methods. In spite of this, section 4.1.3.2 will show that KL has its advantages. An optimisation of these time-constants could, as mentioned in section 4.1.1, not be performed until a method is selected and due to time constraints on the thesis are relegated to further work, see section 5. The RuLSIF method was not applied in [54] as it was invented several years later [75].

These results are obviously not conclusive since they were the product of the training process, applied on training data. Section 4.1.3 will apply the metrics with the optimised parameters on test-data, thus enabling a comparison in precision, recall, thoroughness and efficiency.

4.1.2.2.3 No FAC reference

In this section the KL metric is trained without the use of false alarm compensation, see section 2.4. This is not done as a viable method, but is meant as a reference to verify the necessity for the FAC step.

As the exclusion of the FAC step reduced the 2-dimensional optimisation to a 1-dimensional problem, the intricate multi-step optimisation procedure described in section 3.3, is unnecessary. In 1-dimension the standard brute force approach is still computationally viable. The relevant range is simply split into a sufficiently fine resolution and the algorithm is trained with a large amount of data for each individual potential threshold 10 times.

The result can be seen in figure 4.11, displayed through the mean of the 10 runs along with the corresponding SEM, see section 3.4.2.

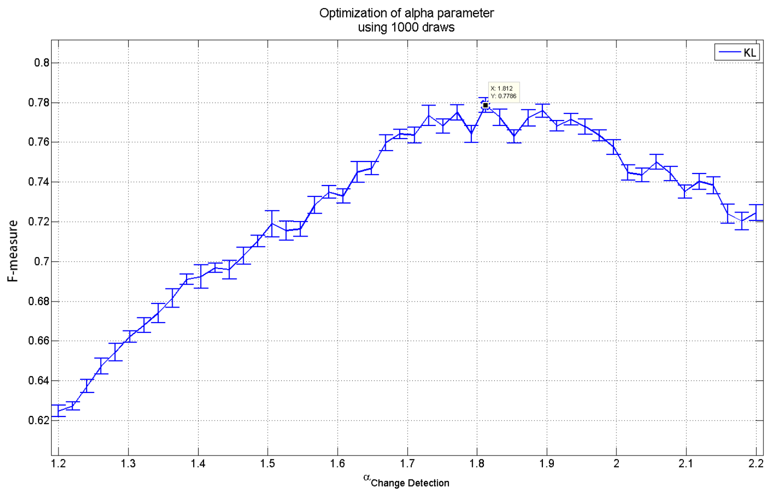

Figure 4.11: This figure is presented as a reference to show the performance of a method without a FAC step. KL is used as it shows decent results and is very efficient, thereby reducing the complexity of this test. Since this is a 1-dimensional optimisation problem the advanced custom optimisation technique used in all other tests is excessive. In this case the KL method is simple tested using a large amount of data - as seen below the title - and a finely meshed basic grid search. This grid search is run 10 times to provide a better estimate and to show the standard error.
As seen the optimisation is convex and a global good approximation of the global maxima is clearly reached. The reader is reminded that the borders represent the approximate levels at which the threshold either accepts or rejects everything. Even though the method is optimised almost perfectly it is seen that the corresponding F-measure reached is far below the methods that apply FAC, see table 4.1.

As seen in figure 4.11, the optimisation problem is quite simple. Clear slopes leading to a plateau with the algorithm selecting a threshold gain roughly in the center of this plateau. The plateau is assumed to be a range of threshold magnitudes where a change merely shifts the precision-recall trade-off. This assumption however remains untested as it is irrelevant to the actual optimisations used.

Obviously the most important information this test yields is that the KL metric without the FAC step is significant worse than with the FAC step. With an F-measure of 0.7786 compared to 0.8432 using FAC, see table 4.2. The other methods have not been trained in a similarly rigorous fashion, but very imprecise trial runs indicated that this conclusion holds for the other methods as well.

4.1.3 Post-test method comparison

This section will apply the remaining methods using the optimised threshold parameters from section 4.1.2. The remaining methods are K-means, KL, and RuLSIF, since DSD was eliminated in section 4.1.1.2, along with the KL/RuLSIF hybrid method.

This comparison will be on the basis of precision, recall, thoroughness and efficiency as applied on test data that was set aside for this purpose, see section 2. Finally this section will select a method based on this comparison.

In section 4.3.1, this chosen method will then be used to test the hypothesis that high MFCCs encode person specific information. In addition to this, the further work section will apply various approaches to optimise this method further, this will however not be in a rigorous fashion and is meant to inspire further research, see section 5.

It should be mentioned that it would be interesting to compare the methods on the basis on how close the average potential change-point is to the true change-point, as was done by Jørgensen et al. [55]. This however was found to be very tricky as it required guessing to which true change-point a specific potential change-point belonged. This was probably not a problem in [55] as a preprocessing step similar to the one described in section 2.1.4, was not performed. This left their data biased, but enabled model comparison on the basis on change-point accuracy. For this reason, this particular comparison is not performed and will not influence the choice of method.

	K-means	KL	RuLSIF	KL/RuLSIF
F-measure	0.840	0.835	0.820	0.793
Precision	0.934	0.889	0.910	0.928
Recall	0.763	0.788	0.747	0.692
F-measure pre-test diff.	-0.0227	-0.00820	-0.0404	-0.0539

Table 4.2: The precise results of the method performance comparison. For a more detailed description of the results and the implications see figure 4.12 or section 4.1.3.1.

4.1.3.1 Performance results

Each method is run on a sample size of 500 change-points, or the equivalent of about 2 hours of speech, created using the method described in section 2.1.5. This is then repeated in a loop for each method individually with a runtime of approximately 10-12 hours per method. Depending on the runtimes, see section 4.1.3.2, this resulted in somewhere between 8 and 68 reruns. The results are presented in figure 4.12 and table 4.2, with the error bars naturally representing the SEM of the measurement. The corresponding FAC performance is seen in figure 4.13 and table 4.3.

The reader is reminded that precision is the percentage of correctly placed change-points, recall is the percentage of change-points found and the F-measure is the harmonic mean of precision and recall, see section 3.4.1.

From figure 4.12 and table 4.2 it is seen that the performance of the methods have shifted compared to each other when compared with training results, see table 4.1. This is caused in part by the generalisation error, but is thought to mainly be caused by the selection of positive noise during training. This can be seen in the fact that the generalisation error is inversely correlated with the amount of data used during the training procedure. Even though this correlation would also cause a larger generalisation error, it is believed that this should not be at this large a magnitude.

As seen the hybrid method suffers the worst, whereas standard KL barely changes at all. With the greatly reduced efficiency of applying CMFAC, the only advantage is in how well CMFAC does at identifying the tough false alarms, as mentioned in section 4.1.2. Interestingly this might merit a hybrid between the standard KL and the Hybrid method using CMFAC, whereby both standard FAC and CMFAC is applied to KL in a similar fashion to the ensemble method mentioned in section 2.4. The reason why this could increase performance is seen in table 4.1, where the CMFAC method chooses a very high change-detection threshold compared to standard FAC. In essence this means that the CMFAC

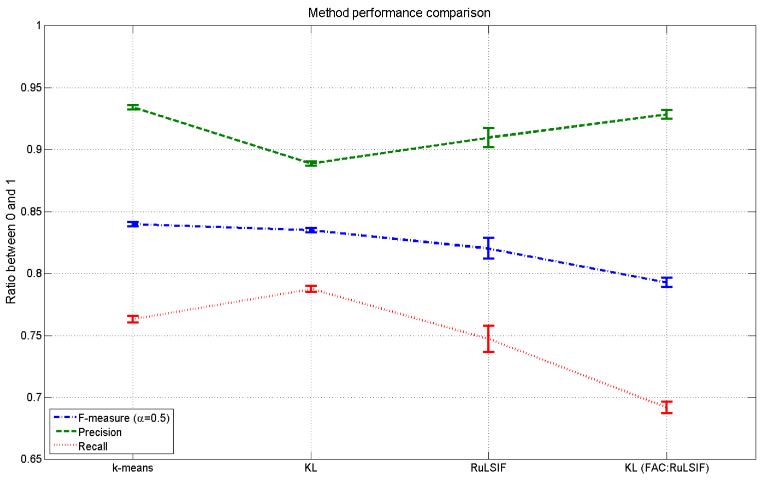

Figure 4.12: The results from the method performance comparison using test data in the form of an F-measure and is constituents. Each method is run on a sample size of 500 change-points, created using the method described in section 2.1.5. This is then repeated in a loop for approximately 10-12 hours per method. Depending on the runtimes, see section 4.1.3.2, this resulted in somewhere between 8 and 68 reruns. For the precise numbers see table 4.2. Oddly enough the failures of the expensive metrics seem to outweigh their benefits and the simple KL method seems to outperform them using the current methodology. Not strictly in terms of F-measure - where K-means is slightly higher - but in terms of a performance versus efficiency ratio, see section 4.1.3.2. In addition the KL method fields the best recall score of all the methods, indicating that it has the highest potential to see further improvements, performed in section 4.3 and 5.2. This result is counter-intuitive as the expectation was that the more advanced methods would also have the highest potential. Their inefficiency during optimisation shows through clearly.

is optimised to only remove the tough false alarms, the ones standard KL will obviously tend to miss. This would also barely decrease efficiency of the CM-FAC method as the standard KL FAC is very efficient using less than 1% of the computing resources, see section 4.1.3.2. The task of implementing this ensemble base FAC is unfortunately beyond the scope of this thesis and is relegated to further work, see section 5.

RuLSIF generalises poorly, probably due the poor training, this could undoubtedly be improved by running the training algorithm of RuLSIF at a higher resolution. However as mentioned in section 4.1.2 and seen in section 4.1.3.2, this is not computationally feasible as such a simulation would take weeks if not months. It could be done using IMMs Cluster, however as mentioned in section 3.3.1, that prospect has been exhausted. Therefore, due to the poor efficiency of RuLSIF it is not feasible in real world applications for years to come and is discarded as a change-detection method.

K-means fared slightly better than RuLSIF in the generalisation and actually showed better results than all other approaches. The margin is slight as compared to standard KL and at the additional factor 2.5 increase in computational runtime, see section 4.1.3.2, the better F-measure performance is clearly not enough. Another point where K-means excels is in its precision where it only competes with the hybrid KL/RuLSIF CMFAC method. Unfortunately this precision comes at the cost of a lower recall score, this is thought to be caused by the smaller feature space used by K-means, see section 4.1.1.3.1. The conclusion is that K-means is a decent choice, but is upper bounded by its dimensionality issues, see section 4.1.1.3.3.

Oddly enough the failures of the expensive metrics seem to outweigh their benefits and the simple KL method seems to outperform them using the current methodology. In addition the KL method fields the best recall score of all the methods, indicating that it has the highest potential to see further improvements, performed in section 4.3 and 5.2. This result is counter-intuitive as the expectation was that the more advanced methods would also have the highest potential.

In figure 4.13 and table 4.3, the FAC performance for the various methods is seen. It is immediately obvious that the optimisation step tend to use FAC aggressively, since it favors recall over precision, or in other words it has found it optimal to remove potential change-point if they might be false alarms. Another interesting note is that, as seen in figure 3.1, RuLSIF as applied to speech detects many 'changes' that do not correspond to speaker changes. What these correspond to is untested, but this has shaped a method that relies heavily on FAC for SCD. However even though RuLSIF relies heavily on FAC, see table 4.3 and figure 4.14, it is simultaneously seen to spend very few computational

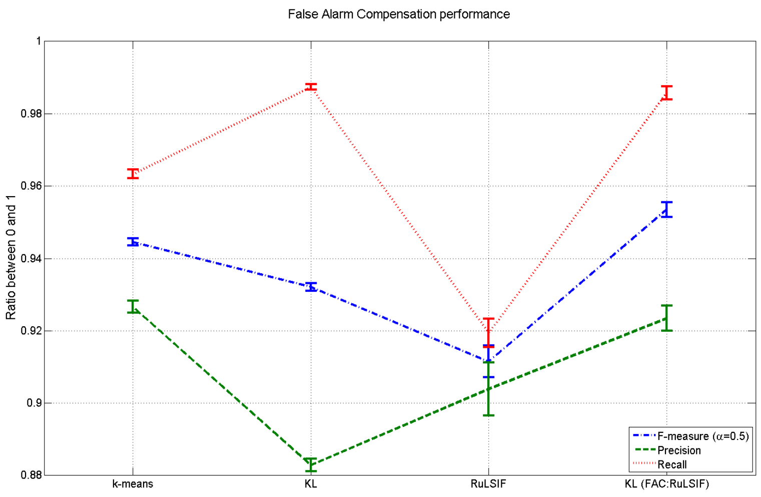

Figure 4.13: The FAC performance during the method performance comparison, see figure 4.12. As seen the optimisation step tends to use FAC aggressively, since it favours recall over precision. In other words it has found it optimal to remove potential change-point if they might be false alarms. Interestingly it is seen that standard FAC for KL has almost the same recall as the CM-FAC method, but as expected the more expensive CMFAC is substantially more precise. RuLSIF is seen to have the worst FAC performance, this however is misleading. As seen in figure 4.14 RuLSIF applied FAC to a far greater degree than the other methods, it is actively using it, rather than merely polishing its results. Unsurprisingly the K-means method has a solid FAC performance in line with its great performance in general.

	K-means	KL	RuLSIF	KL/RuLSIF
F-measure	0.945	0.932	0.911	0.953
Precision	0.927	0.882	0.904	0.923
Recall	0.963	0.987	0.919	0.986
False alarms [%]	7.79	6.56	17.8	4.01

Table 4.3: The precise results of the FAC performance, see figure 4.13, and the false alarm percentage, see figure 4.14. For an explanation of the results and their implications see the caption of the corresponding figure or section 4.1.3.1.

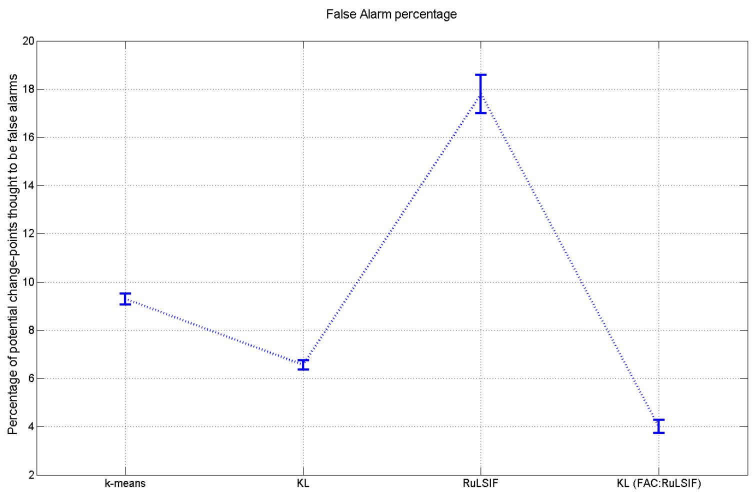

Figure 4.14: The false alarm percentage used by the methods during the method performance comparison, see figure 4.12. As seen RuLSIF relies heavily on FAC, further decreasing its efficiency, it however is simultaneously seen to spend very few computational resources on the FAC step, see section 4.1.3.2. A discussion of this strange phenomena is found in section 4.1.3.1. The hybrid method is seen to use FAC aggressively, which is fortunate as that increases its efficiency. It does however suggest that the hybrid method fails to use FAC proficiently. K-means and standard KL are seen to use FAC in a more well-rounded manner. For the precise percentages see table 4.3.

	K-means	KL	RuLSIF	KL/RuLSIF
Total runtime [Sec./CP]	0.51	0.20	16	1.4
FAC runtime [% of total]	5.9	0.90	0.013	86

Table 4.4: The precise results for the runtime test, included for reference. For a description and interpretation of these results see figure 4.15 or section 4.1.3.2.

resources on the FAC step, see in section 4.1.3.2. The hypothesis is that this is caused by a combination of factors; RuLSIF is optimised to produce a lot of potential change-points and then to remove these by applying FAC, this is seen in the relatively close threshold gains, see table 4.1. However with the large number of change-points the average distance between change-points decreases. During FAC the amount of data used to test for false alarms is maximised, but is limited to not use data within 1 second of other potential change-points, see section 2.4. Since RuLSIF's runtime scales super-linearly with the data amount the large number of change points might actually reduce the runtime of the FAC step.

It must be stressed though that this is an untested hypothesis and that its testing is left to further work, see section 5.

If this is true, it would mean that RuLSIF might fare better on data with long speaker turns, with a recursive FAC paradigm or without FAC entirely. However as RuLSIF is very computationally expensive none of these options are explored, the slight improvement in performance is far out weighted by the further decreases in efficiency they would cost. This could potentially also be the reason for the peak on the south border of left figure in figure 4.9, which is the part of the function-space that corresponds to a FAC'less method.

It should be noted that RFAC is found to improve KL performance, see section 5.2, and KL is found to decrease in performance without the FAC step, see section 4.1.2.2.3. How/if this applies to RuLSIF is untested.

4.1.3.2 Efficiency results

Runtime measurements for the individual methods are displayed in figure 4.15.

The time measure is a sum of the time it takes to calculate the models, the time

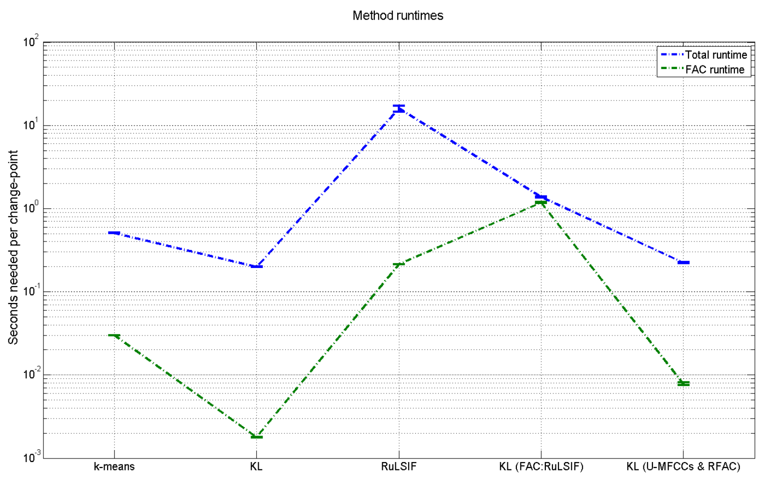

Figure 4.15: The runtimes for the various method, displayed as the time needed per actual change-point, meaning that thorough methods automatically score slightly higher. The data is presented on a logarithmic scale to fit all results onto the figure, the precise numbers are seen in table 4.4.
As seen RuLSIF is very computationally heavy at about 2 orders of magnitude slower than the most efficient method which as expected is KL. This is also seen in the hybrid method, where the FAC step takes the majority of the computational effort. K-means has undergone several optimisations, see section 3.2.1.4, and is therefore faster than the hybrid, but still clocks in at 2.5 times slower than KL.

it takes to calculate the dissimilarity measures and finally the time it takes to run the FAC. This sum is then normalized by the amount of change-points in the measurement. It must be noted that the result here does not include the overhead, e.g. the feature extraction among others. The result is presented as the mean and the SEM. As seen in figure 4.15, RuLSIF is very computationally heavy at about 2 orders of magnitude, or more precisely 80 times, slower than the most efficient method which as expected is KL.

K-means has undergone several optimisations, see section 3.2.1.4, and clocks in at 2.5 times slower than KL. The precise numbers are seen in table 4.4.

Disclaimer: It must be noted that the run-times are merely guidelines and the exact numbers depend on the implementation, particularly on how parallelizable it is and how well the MATLAB's Just-In Time, JIT, accelerator can optimise it. The run times at measured using MATLAB's tic/toc functions.

To maintain a clean measure of the run-times the methods were naturally run individually except for RuLSIF, which was run in parallel with itself to maximise CPU usage. This slightly exaggerates is efficiency, but was necessary to reduce the required total runtime.

For reference; the hardware used was a single Intel® Core^{TM} i5-2400k processor running four cores at 3.4GHz, 32GB RAM and running MATLAB 7.11.0 (R2010b) 64-bit.

As such these run-time ratios are meant as guidelines, especially in the context of implementation outside the MATLAB environment, they are however assumed to be similar in such a case.

4.2 Method comparison conclusion

In summary the choice of applying K-means in the previous project by Jørgensen et al. [55] was a decent choice given the implementation. However it is found in section 4.1.2.1, that a smaller codebook will increase efficiency with no effect on the performance. The K-means method has a severe limitation though, its performance is upper bounded by its dimensionality issues, see section 4.1.1.3.3.

It is found here that the KL method is fairly unambiguously the superior method due to its close second in F-measure, its low computational cost and its high recall indicating hidden potential. Furthermore the KL training procedure is efficient enough that retraining given new data in commercial software is still slow, but not excessively so.

The backwards feature selection mentioned in section 2.2.4, will be performed on KL in section 4.3.1. Once an optimal feature set is found the KL method will undergo yet another training session, in section 5.2, using the multi-step FAC method, mentioned in section 2.4.

4.3 Method refinement

This section will test the hypothesis from section 2.2.4, that using only the upper MFCCs will improve performance. The theory is that the deconvolution of the time signal via the ceptrum shifts the envelope of the spectrum to the low MFCCs and the fine structure to the high MFCCs. The hypothesis is that the human speech system when talking can be modelled as an Linear Time-Invariant, LTI, system with the excitation signal generated by the voice box undergoing a convolution by the vocal tract acting as a linear filter. The vocal tract filter in this model shapes the excitation signal, thereby forming the literal semantics of the speech signal, whereas the excitation signal is arbitrary, person specific and should thereby work as sort of an acoustic fingerprint.

4.3.1 Backwards feature selection

If the hypothesis from introduction to this section holds then the upper MFCCs should yield better results than the lower MFCCs in the context of SCD. This section will use the designed optimisation tools from section 3.3 on the method selected in section 4.1.3 to test this hypothesis. As mentioned in section 2.2.2, up till this point 12 MFCCs have been used along with their 1^{st} and 2^{nd} order derivatives, in this section this will still be the case. What is changed is the Mel filter bank size described in section 2.2.4.2.

As mentioned in section 2.2.1, 20 Mel filters will be used and KL is trained on the outputs of the 12 upper and on the 12 lower ones. The very first component is obviously still left out, see section 2.2.2 for the explanation.

It must be noted that the ISP toolbox [53] - used to generate the MFCCs - does not support the selection of specific MFCCs. The ISP toolbox automatically selects the lower MFCCs, which probably means that it was designed with speech transcription as the main focus. So far the outputs from all filters have been used; this has therefore not had an impact up till this point. At least that was the assumption, however - as described in the next paragraph - this was not the case.

To enable feature selection the ISP toolbox [53] underwent more custom modifications, see section 1.2. During this overhaul of the ISP toolbox, some leftover debugging code from its creation was discovered, wherein several inputs were overridden by custom, non-default values, in a fashion easily overlooked under a casual inspection and invisible from the outside. This included the amount of filters used in the Mel filter bank. This debugging code was obviously removed and all previous tests were scrapped and rerun. The creators of the ISP toolbox were notified and the entire toolbox was thoroughly examined. The relevant modified scripts can be seen in appendix A.5.4.

4.3.2 Training results

The training results of the KL method using the upper and lower MFCCs are shown in figure 4.16 and 4.17, respectively. The training or optimisation method used is described in section 3.3.

At a casual glance the results in figures 4.16 and 4.17 seem similar; in both cases the optimisation of KL is roughly a convex optimisation problem which the optimisation algorithm handles well. The results seem similar with the main difference being slightly steeper slopes in the upper MFCCs case. However when studying the F-measures that the color contours correspond to and the F-measure at the found local minima the difference becomes readily apparent.

An interesting note is that the full KL method training results, see figure 4.8, did show two peaks. While the upper and lower KL methods find one peak each, see figures 4.16 and 4.17. These original peaks might correspond to paired down versions of the full method that resemble the upper and lower methods. Testing this hypothesis would however require an extensive feature selection process which will not be performed.

In table 4.5 the training result using the upper and lower MFCCs are compared to the previous training results using all MFCCs from section 4.1.2.

As seen in table 4.5, interestingly the FAC is apparently much more reliable in both the lower and upper case compared to the full case, as the optimisation chooses to use significantly more of it, especially the lower case which removes one out of four potential change-points, see table 4.3.

The fact that the algorithm is more FAC reliant means that it is better able

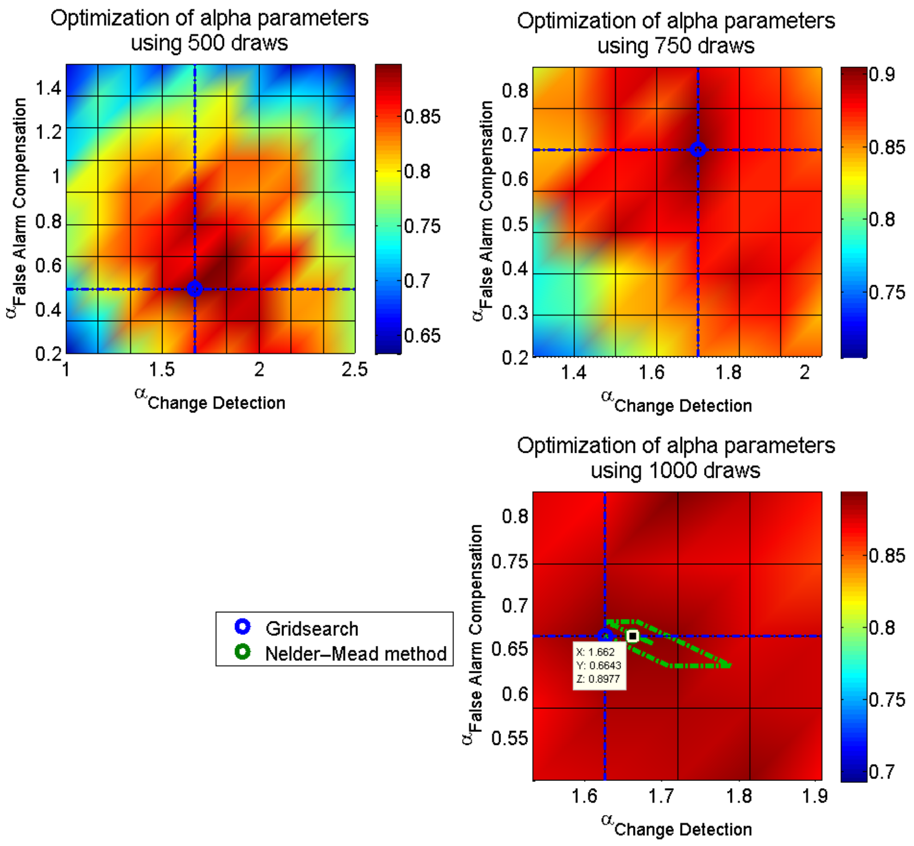

Figure 4.16: The colours - which are a linear interpolation of the values of the grid - represent the F-measure for the corresponding threshold gain combination. Note that the grid is independent of the axes. This figure displays the training steps and training result of the KL method using only upper MFFCs; using the optimisation method described in section 3.3. The results here are compared to the other methods, all MFCCs and lower MFCCs, on training data in table 4.5. The various methods are compared using test data in section 4.3.3.
As seen in this figure a large amount of data is drawn for each grid point, and the resulting objective function is very smooth. In the first iteration the grid method finds the global maxima roughly, the subsequent iterations and the direct search method merely locate the peak with greater precision.

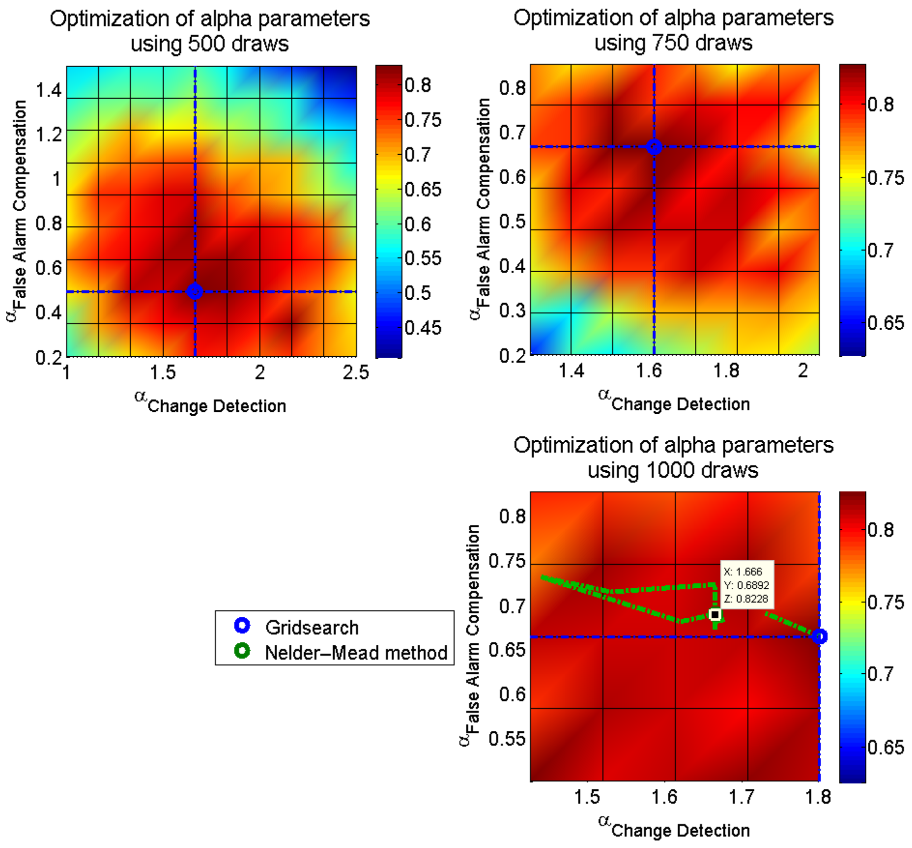

Figure 4.17: The colours - which are a linear interpolation of the values of the grid - represent the F-measure for the corresponding threshold gain combination. Note that the grid is independent of the axes. This figure displays the training steps and training result of the KL method using only lower MFFCs; using the optimisation method described in section 3.3. The results here are compared to the other methods, all MFCCs and upper MFCCs, on training data in table 4.5. The various methods are compared using test data in section 4.3.3.
As seen in this figure a large amount of data is drawn for each grid point, and the resulting objective function is very smooth. In the first iteration the grid method finds the global maxima roughly, the subsequent iterations and the direct search method merely locate the peak with greater precision.

(MFCC range)	α_{cd}	α_{FAC}	F-measure
KL (Full)	1.703	0.4324	0.8432
KL (Lower)	1.666	0.6892	0.8228
KL (Upper)	1.662	0.6643	0.8977

Table 4.5: This table shows the final threshold parameters for KL using the 3 feature sets, lower MFCCs, upper MFCCs and all MFCCs. The intermediate stages, of the optimisation method, for the various methods can be seen in figures, 4.8 (Full), 4.16 (upper) and 4.17 (lower).
Firstly it is seen that using the upper MFCCs, as expected, seems to result in a marked improvement. Beyond this it is seen that the FAC is interestingly much more reliable in both the lower and upper case compared to the full case, as the optimisation chooses to use significantly more of it, especially the lower case which removes one out of four potential change-points, see table 4.3. It is believed that this is a reflection of what a uni-modal model is able to interpret, see section 4.3.2.

to distinguish speakers given large amounts of data. As both K-means, but especially RuLSIF applied more FAC than KL, see section 4.1.3, indicates that the lower and upper MFCCs distinguish speakers based on different aspects and that they were better able to model this bi-modal underlying PDF. This is as expected since RuLSIF and K-means do not model the speaker using a uni-modal model like KL.

Interestingly this bodes well for KL as it should do well only looking at the lower or only the upper MFCCs. This is also indicated by the magnitude of the F-measure at the local minima found during training, see table 4.1; the training result indicates that KL using lower MFCCs is almost as good as using the full range, however as the hypothesis this section set out to test states, KL using only the upper MFCCs shows great promise.

These results are naturally not conclusive as they are the product of the training process, applied on training data. Section 4.3.3 will apply KL with the relevant feature set and the optimised parameters on test-data, thus testing the generalisability of the results found here.

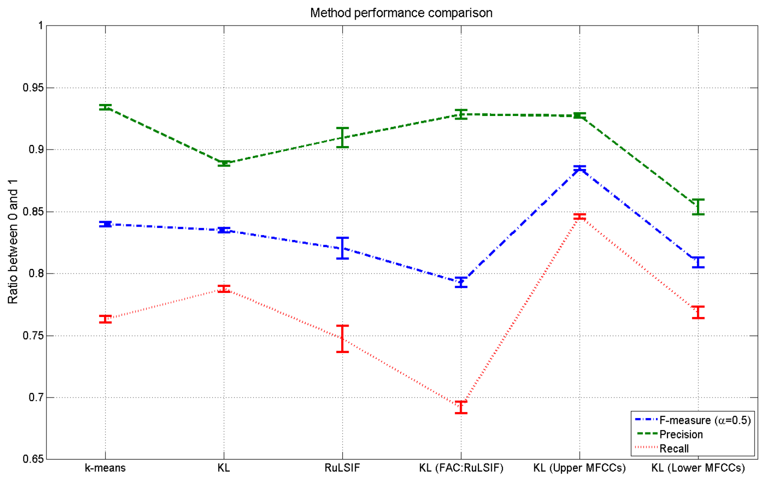

Figure 4.18: This figure shows the performance of the upper and lower MFCC KL methods compared to the methods tested in section 4.1.3.1. The result is seen to be conclusive; compared to the previous full range KL method, using only the upper MFCCs significantly improves the performance, whereas using only the lower MFCCs significantly deteriorates the performance. Where the full range KL lies in between, which further points to the hypothesis; that high and low MFCCs distinguish speakers differently and that full range KL being uni-model fails to capture each attribute independently. Confirming the validity of the speech production system model from section 2.2.4.

4.3.3 Test results

In this section the test results for KL using lower MFCCs and using upper MFCCs are compared to KL using the full range, see section 4.1.3.

In figure 4.18 and table 4.6 it is again confirmed, as was seen in section 4.1.3, that the training result found using the optimisation method generalises very well using the resolution and data amount applied here. The reader is reminded that the reason why all methods tend to favor precision over recall is the direct consequence of the FAC paradigm. Furthermore the result is seen to be conclusive; using only the upper MFCCs significantly improves the performance, whereas using only the lower MFCCs significantly deteriorates the performance. Where the full range KL lies in between which further points to the hypothesis;

(MFCC range)	KL (Full)	KL (lower)	KL (upper)
F-measure	0.835	0.809	0.885
Precision	0.889	0.854	0.927
Recall	0.788	0.769	0.846
F-measure pre-test diff.	-0.00820	-0.0138	-0.0127

Table 4.6: The precise results of the method performance comparison between the different feature sets for the KL method. For a more detailed description of the results and the implications see figure 4.18 or section 4.3.3.

(MFCC range)	KL (Full)	KL (lower)	KL (upper)
F-measure	0.932	0.895	0.952
Precision	0.882	0.849	0.924
Recall	0.987	0.948	0.981
False alarms [%]	6.56	26.8	13.3

Table 4.7: The precise results of the FAC performance, see figure 4.19, and the false alarm percentage, see figure 4.20. For an explanation of the results and their implications see the caption of the corresponding figure or section 4.3.3.

that high and low MFCCs distinguish speakers differently and that full range KL being uni-model fails to capture each attribute independently.

In figures 4.19, 4.20 and table 4.7, KL using the different feature sets is compared on the basis of FAC.

In figure 4.19, 4.20 and table 4.7, the conclusion, from section 4.3.2, that KL when trained on lower or on upper MFCCs relies heavily on FAC is confirmed. Similarly to the results with other methods in section 4.1.3, the FAC is erring on the aggressive side. It is also found that lower KL is far more reliant on FAC than upper KL, this seems to indicate one of two things. Either the information relevant for SCD in the lower MFCCs is found on fairly long time scales, at least longer than the analysis window size of 3 seconds or that lower MFCC is very sensitive to multiple speakers in an analysis window, something FAC inherently counteracts with its variable size analysis window, see section 2.4. Regardless of the reason for this KL using lower MFCCs is clearly inferior to KL using upper MFCCs. KL using upper MFCCs is seen to maintain the high recall FAC of full range KL, even though it applies FAC on twice as many potential change-points and has a greatly increased precision. This is clearly one of the reasons why upper KL has superior performance compared to full range KL.

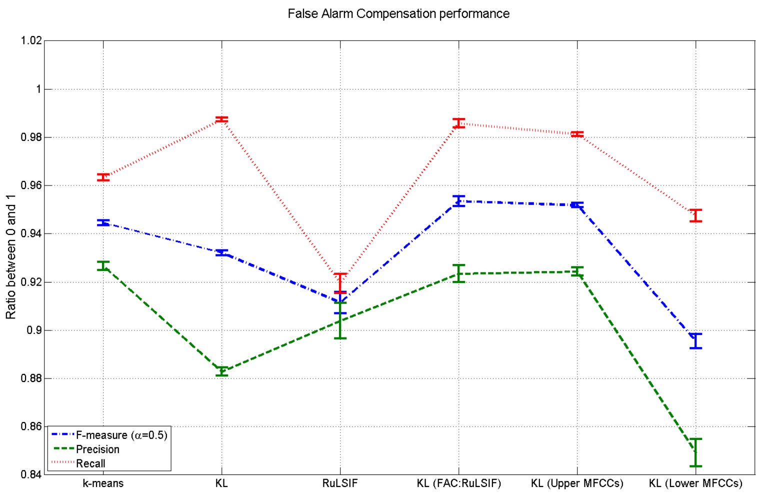

Figure 4.19: This figure displays the FAC performance during the performance test for different KL feature sets, see figure 4.18, here compared to the methods tested in section 4.1.3.1. Again the FAC method is found to adopt an aggressive strategy - liable to throw away potential change-points if unsure - seen in the fact that RCL is greater than PRC, see section 3.4.1.1. KL using upper MFCCs is seen to maintain the high recall of the full range KL, even though it applies FAC on twice as many potential change-points and has a greatly increased precision. This is clearly one of the reasons why upper KL has superior performance compared to full range KL.

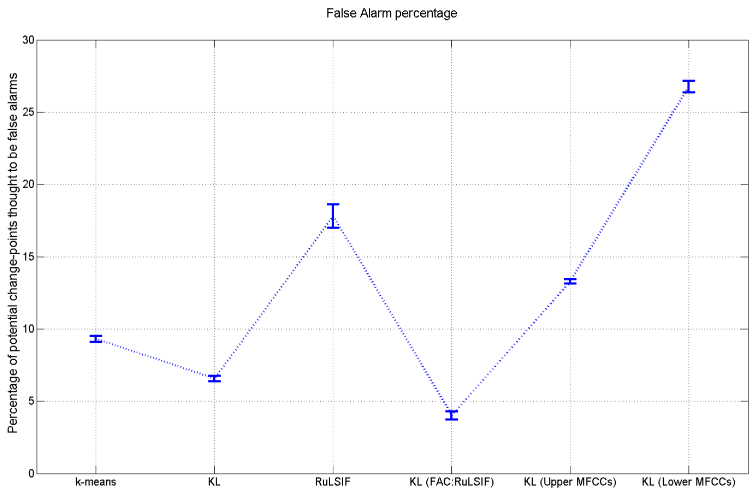

Figure 4.20: This figure displays the percentage of false alarms the KL method chooses depending on the feature sets compared to all methods from section 4.1.3.1. As seen the conclusion from section 4.3.2 is confirmed, the new KL methods rely choose to rely heavily on FAC. It is also found that lower KL is far more reliant on FAC than upper KL, this seems to indicate one of two things. Either the information relevant for SCD in the lower MFCCs is found on fairly long time scales, at least longer than the analysis window size of 3 seconds or that lower MFCC is very sensitive to multiple speakers in an analysis window, something FAC inherently counteracts with its variable size analysis window. This is not investigated as the lower MFCC method is clearly inferior to the upper MFCC one, see figure 4.18.

4.4 Method refinement conclusion

In conclusion the hypothesis this section set out to test is clearly validated. KL shows marked improvement using only upper MFCCs compared to using the full range, while maintaining the same feature vector size. Whereas using only the lower MFCCs forces the optimisation to choose a very heavy FAC strategy, however impressive this fails to reach even the performance of the full range KL found to outperform other methods in section 4.1.3.

The impact of feature selection on the runtime of the KL method is left untested. The runtime would be slightly longer as both the lower MFCC and upper MFCC versions rely more on the FAC step than the full range MFCC version. However as seen in section 4.1.3.2, this decrease in efficiency would be very slight as the FAC step of the KL method is very efficient, taking less than 1% of the full runtime for the full-range MFCC version.

Doing a full backwards feature selection, rather than a single step as is performed in this section, seems very promising. However this would likely tune the method to the format of the dataset and the generalisability could suffer; this however would require the method be applied to other data sets than ELSDSR, which is beyond the scope of this thesis. Beyond that a full backwards feature selection would be very computationally costly and though all the relevant software has been created it is beyond the scope of this thesis to run it. The work here unfortunately only has the resources to provide this proof of concept, the full backwards feature selection is left to further work, see section 5.

Chapter 5

Further work

Beyond the work described so far, a range of topics and fields were propped and tested at varying degrees of rigorousness. This section will quickly describe the results from the areas in which a sufficiently large amount of work was done. The methods and results presented in this section were not given adequate attention or computing resources to merit admission into the greater body of work. Beyond this, this section will conclude with a list of promising avenues that was left unexplored due to time constraints.

5.1 Speaker clustering

Once speaker change points have been identified and the audio stream has been segmented into a set of speaker turns, a crucial part of the original question of "who spoke when?" still remains, namely the who. Assuming some of the segments are spoken by the same person, the task of identifying these and annotating them with a label is an important part of speaker diarisation. As mentioned in section 1, this project does not attempt to classify these speech segments, the process of grouping the segments is purely addressed as a clustering problem.

This section assumes that the SCD process operates ideally separating speaker segments perfectly. The reason for this assumption is to avoid carrying over biases from the SCD process. It must be noted that this assumption beyond rendering the process of perfect clustering ideally possible, removes the possibility of investigating possible non-linear processes whereby biases in the SCD process interact with biases in the clustering process.

By cluster, what is meant is that all MFCCs from a speaker segment are grouped together with sequential information ignored. The speaker clustering process uses the 12 MFCC 1^{st} & 2^{nd} derivatives upper range features found to improve results in section 4.3.1.

5.1.1 Agglomerative Hierarchical Clustering

The clustering method implemented and compared in this thesis are various dissimilarity measures using an Agglomerative Hierarchical Clustering, AHC, [8] approach. The basic concept behind AHC is to define a dissimilarity metric to distinguish between clusters, e.g. the Euclidean distance between their closest members, usually in the form of linkage functions. Then to find the least dissimilar clusters and combine them, repeating this process until some desired criteria is met.

This process is slightly more complicated when clusters cannot be directly compared in a geometric sense as is the case in this thesis. The reason being that even using the process of MFC the clusters are occupying approximately the same region of MFCC space, see figure 2.2. This issue is dealt with by instead of directly comparing clusters, a model with a set of parameters is trained to describe the underlying cluster. This model is then compared to the models trained on every other cluster, forming the dissimilarity matrix, see an example in figure 5.1. This dissimilarity matrix must be computed fully at every step of the process, a range of method to optimise this was investigated, a good solution was not found.

This model comparison process brings with it an issue, as model training scales super-linearly with the data size. This issue is circumvented by simply building the model on a random sample of the data. The size of this sample is set to T_{max} which all the way through this thesis has defined the maximal amount of data to be applied.

This sampling however brings yet another issue into the picture, which is that sampled data might fail to represent the structure of the cluster. A solution to this issue is proposed by Han et al. [42] in the form of a normalized BIC based dissimilarity measure termed the Information Change Rate, ICR. It is called ICR as [42] shows it measures how much information (entropy) would be changed (increased) by merging the clusters under consideration. In this sense it functions slightly like a Kullback-Leibler dissimilarity metric with incorporated regularization and normalization of cluster sizes.

Using the non-geometric dissimilarity measures enable AHC, but losses the simple progression of ever more dissimilar cluster being the least dissimilar clusters, resulting in a slightly messy dendrogram. The issue of a non-simply progression makes the use of simple thresholding techniques as a stopping criteria fail.

The stopping criteria proposed here is to incorporate active learning techniques incorporating the user of the audio editing software as an oracle. Settles [104] provides a broad view into the field of active learning. This framework of ideas was not fully developed, but would require the user to provide an estimate of the amount of speakers in the pod-cast, the AHC algorithm would then look for a marked increase in dissimilarity around the required number of clusters.

5.1.2 AHC: Dissimilarity metrics

The AHC process is tested using all metrics described in section 3.2, along with the BIC and the ICR dissimilarity metrics as proposed in [42]. In this section only the results from the KL trial, along with the BIC and ICR trials are presented. The results are presented in the form of dendrograms, see figures 5.2, 5.3 and 5.4.

The generation of the dendrogram in figures 5.2, 5.4 and 5.3, are repeated 5 times and combined into the results seen in figure 5.5.

In the top figure in 5.5 it is seen that all metrics start mixing speakers into the same clusters before all segments of the same speaker are combined. Resulting in a smooth transition to a single cluster containing all speakers. Beyond this it is seen that BIC and KL do significantly better at maintaining cluster purity than ICR, as ICR start combing speaker clusters far in advanced of the target value of 5 speaker clusters.

In figures 5.2, 5.3 and 5.4, it is seen that the results from the KL and the BIC approach resemble each other and both work decently well. However a feed-back process is seen to deteriorates results. What happens is that two individually speaker-pure clusters, of several speaker segments each, can resemble each other more than the cluster's own outliers. When two of these pure clusters are combined this process is even more likely to repeat itself, rapidly deteriorating the results.

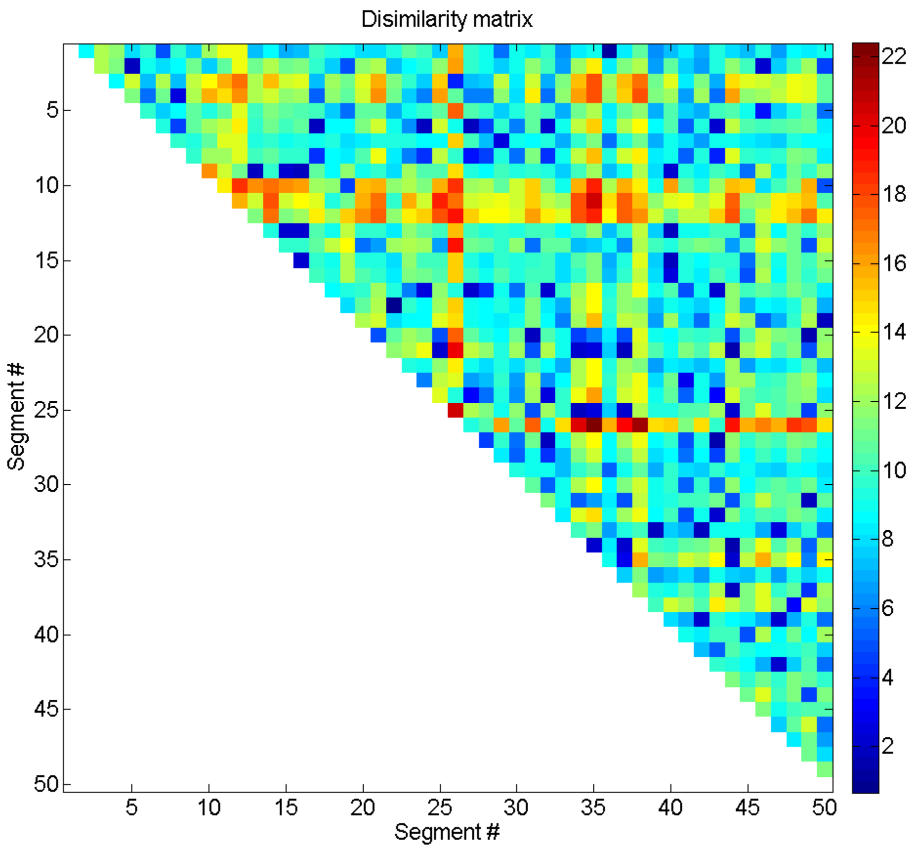

Figure 5.1: This figure shows the dissimilarity matrix at a single step in the AHC process. As seen every segment is compared to every other segment, this obviously forms a symmetric matrix, for efficiency only one triangle is calculated. Segment dissimilarity is displayed through color, with deep blue being identical and deep red being very dissimilar. The scale and slope of the dissimilarity is determined by the dissimilarity metric applied, in this case KL. As seen the dissimilarity matrix will form striped patterns, this reflects that some speakers are apparently very distinguishable.

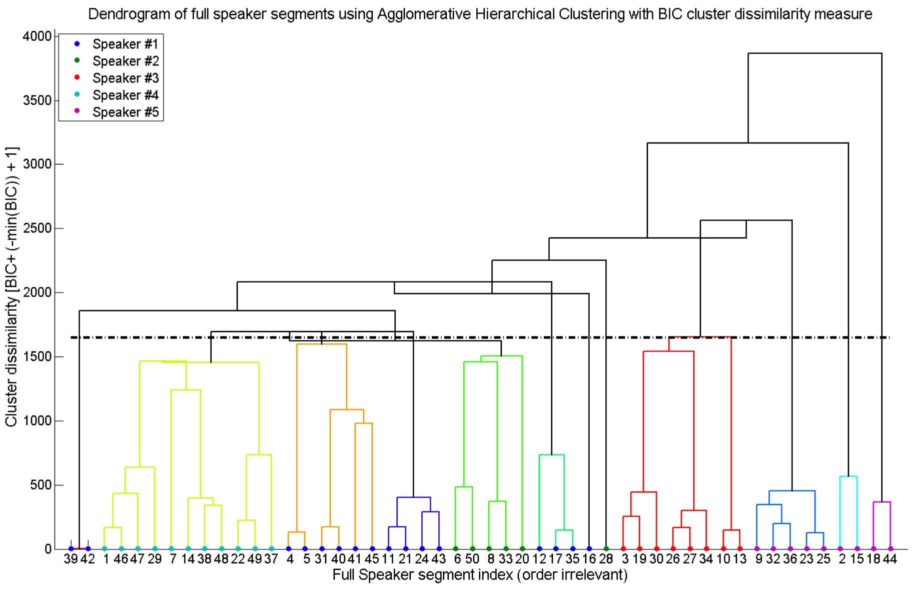

Figure 5.2: This figure shows the agglomerative hierarchical clustering process using the BIC dissimilarity metric in the form of a dendrogram. As seen the algorithm does fairly well to begin with at combining segments spoken by the same person. What is however also seen is that as soon as the process combines clusters containing different speakers a feed-back effect occurs, making more misclassification more likely. This feed-back effect makes the approach less appealing. This is also evident using KL, but the results from the ICR trials hints at a solution.

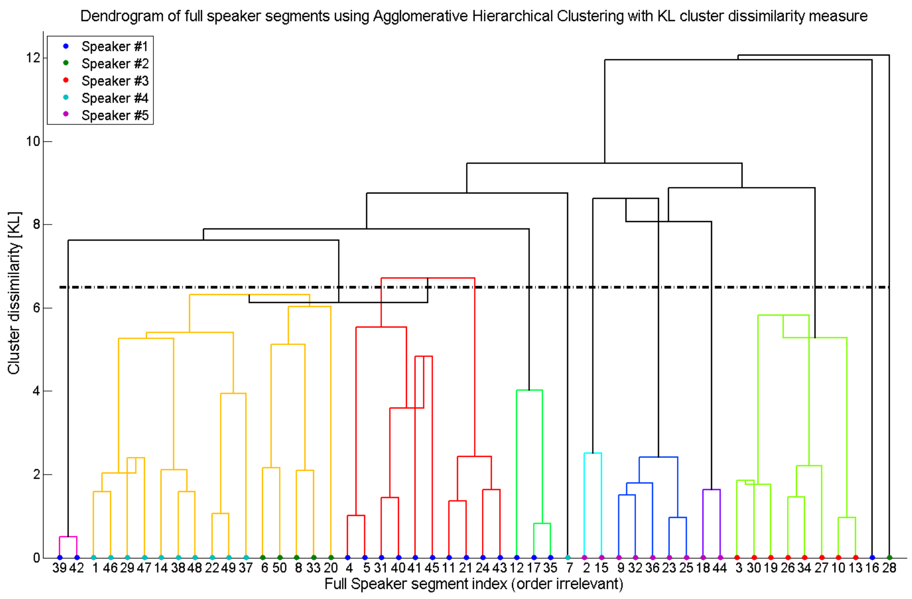

Figure 5.3: This figure shows the agglomerative hierarchical clustering process using the KL dissimilarity metric in the form of a dendrogram. As seen the algorithm does fairly well to begin with at combining segments spoken by the same person. What is however also seen is that as soon as the process combines clusters containing different speakers a feed-back effect occurs, making more misclassification more likely. This feed-back effect makes the approach less appealing. This is also evident using BIC, but the results from the ICR trials hints at a solution.

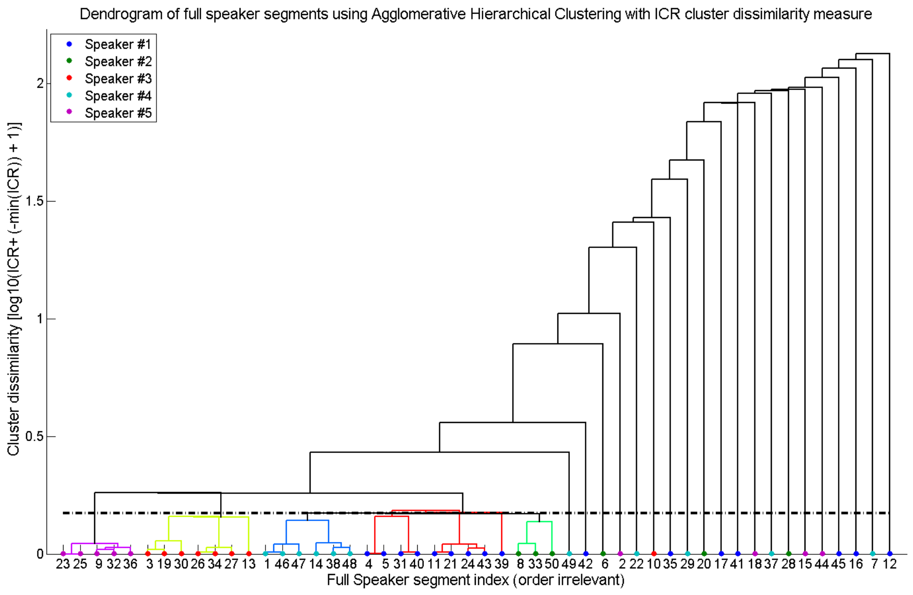

Figure 5.4: This figure shows the agglomerative hierarchical clustering process using the ICR dissimilarity metric in the form of a dendrogram. The reader is directed to notice that the cluster dissimilarity on the y-axis in this case of ICR is shown logarithmically. This makes it very easy to notice the point at which single speaker segments are added using a simple thresholding technique

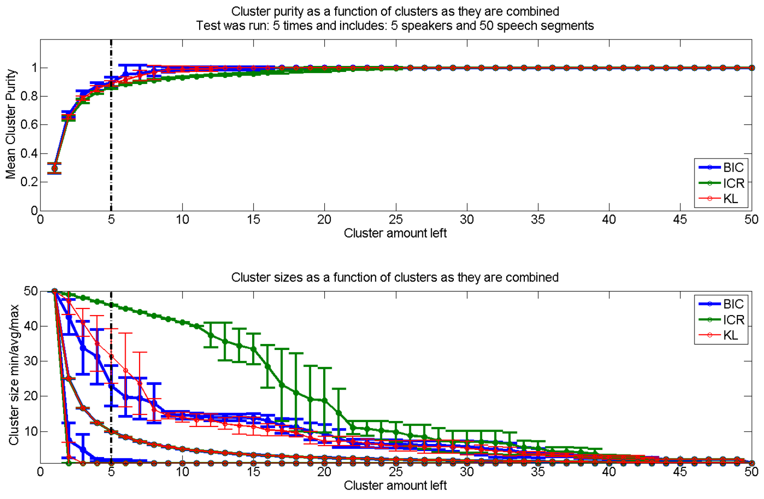

Figure 5.5: This figure presents the full results from a series of 5 AHC processes. It should be noted that these 2 figures must be read from right to left. Starting at 50 clusters on the right which are then combined down to a single cluster on the left. The upper figure shows the mean cluster purity, a measure describing the mean over all cluster of its ratio between its largest subclass and its total size. The lower figure shows the min., avg. and max. cluster sizes as a function of clusters left.
As seen all methods here fail to maintain perfect cluster purity down to the ideal of 5 speaker clusters. In the lower figure it is seen that BIC is slightly better at maintaining an equal size for its clusters, whereas ICR fails completely and quickly generates a single super cluster. KL is seen to be somewhere in between these extremes.

However, the ICR results suggest an interesting novel approach, rather than completing the full AHC process. As seen in figure 5.4 ICR starts out very robustly by finding representative samples of each speaker, and follows this by adding every other speaker one at a time. The interesting novel approach could be to simply stop the AHC process before the large clusters are combined. Then subsequently treat the clustering problem as a classification problem which has an "unknown" class. This approach remains untested, but looks promising as this, in similar fields, has been shown to result in a marked increase in performance [44].

Furthermore, in the context of SCD, the additional classification step is required. The reason being that speaker turns smaller than the size of an analysis window must be excluded from the clustering process and classified afterward to the cluster they resemble the most. Including these in the AHC process resulted in a marked decrease in performance.

The implementation of the speaker clustering using AHC can be found in appendix A.1.2.

5.2 Recursive False Alarm Compensation

This section will consider the notion of a multi-stage FAC procedure as mentioned in section 2.4, here termed Recursive False Alarm Compensation, RFAC. The idea behind standard FAC is to optimise the data usage around a potential change-point in order to validate it. Standard FAC is limited in that it does not use data within 1 sec of the nearby change-points, even if a nearby change-point is found to be a false alarm.

RFAC extends this by recursively calling FAC on change-points in which nearby data was freed by false alarms. This may cause a run-away effect, if the nearby false alarm was not actually a false alarm. The optimisation algorithm found the best use in standard FAC through a fairly aggressive strategy, this is seen in the recall to precision ratio of table 4.7. Besides improving the F-measure, the hope is that the optimisation algorithm will tune the method to use the RFAC in a more well rounded fashion, thereby optimising the KL approach to its fullest potential. Beyond this, it is seen from the efficiency results in section 4.1.3.2 that RFAC is very cheap computationally.

The training result for the RFAC KL method is seen in figure 5.6. When compared to the training result using standard FAC, see figure 4.16, it is very clear that the RFAC method opened up new design choices for the algorithm. The

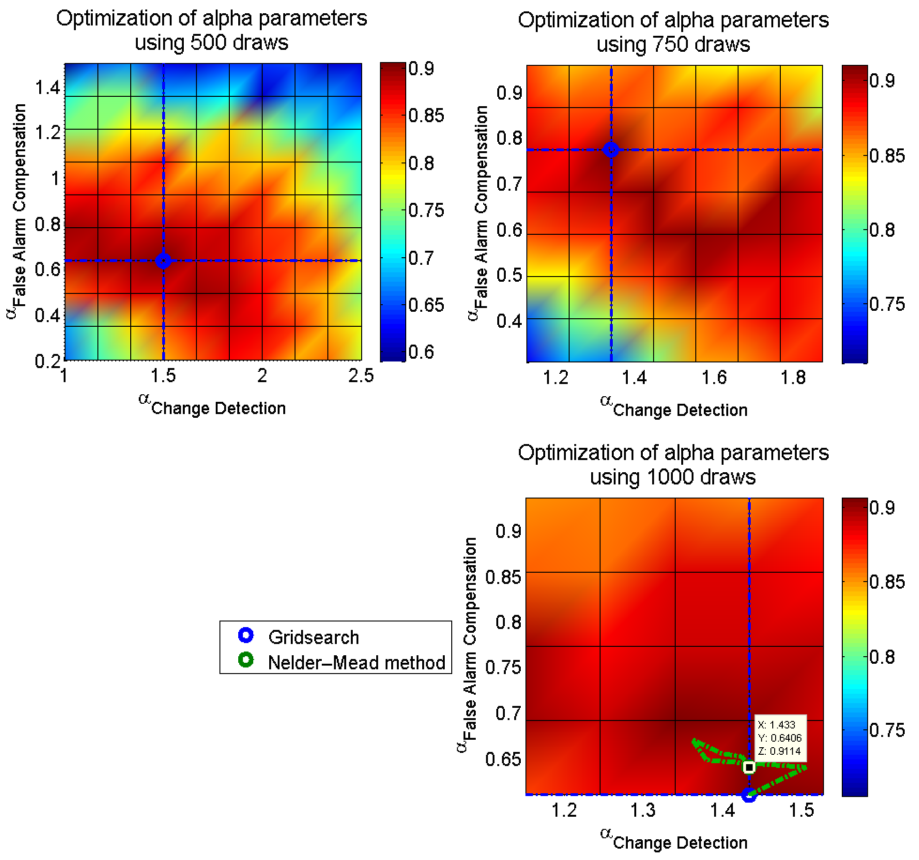

Figure 5.6: The colours - which are a linear interpolation of the values of the grid - represent the F-measure for the corresponding threshold gain combination. Note that the grid is independent of the axes. This figure displays the training steps and training result of the - upper MFCCs, RFAC - KL method using the optimisation method described in section 3.3. The result here shows that the RFAC step yields a slightly higher training F-measure compared to the standard FAC step in figure 4.16. It is also seen that the design leans heavily on RFAC as the change-point detection threshold is substantially lower and the FAC threshold is slightly lower. The reader is reminded that the borders represent thresholds that accept and reject everything respectively.

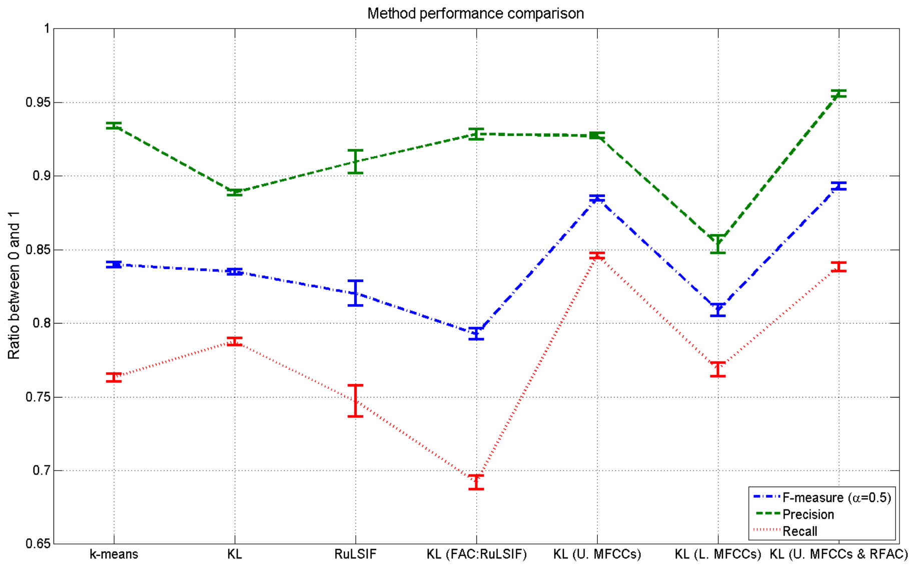

Figure 5.7: In this figure the performance of KL using RFAC is compared to all other trained methods. The result is generated using 500 draws of test 37 times, the data is represented as the mean and the standard error seen as errorbars.
As seen the RFAC method only slightly improves the performance, with an F-measure of 0.893 using RFAC and an 0.885 without, totalling an increase of 0.00828 slightly less than the training results indicated. This improvement comes from an increased precision up to 0.956 from 0.927, but a reduction in recall down to 0.838 from 0.846.

F-measure of the peak found using training data is 0.0137 higher, whether this generalises onto test data is seen i figure 5.7. As was expected it is seen that the algorithm also tests a paradigm where the change-point detection simply passing everything through and leaves the entire job for the RFAC step (a value on the left border). This design option was not selected, which is fortunate as the implications would be unclear. Suffice to say that a change-detection step that allows everything through still only selects a peak if it is the local maxima, where locality is dependent on the T_i parameter, see section 2.3, such an approach would be very data dependent. The algorithm's choice falls on a slightly less change-point detection based method, the FAC threshold however remains in roughly the same location. This obviously means that the newly designed approach leans heavily on the RFAC steps.

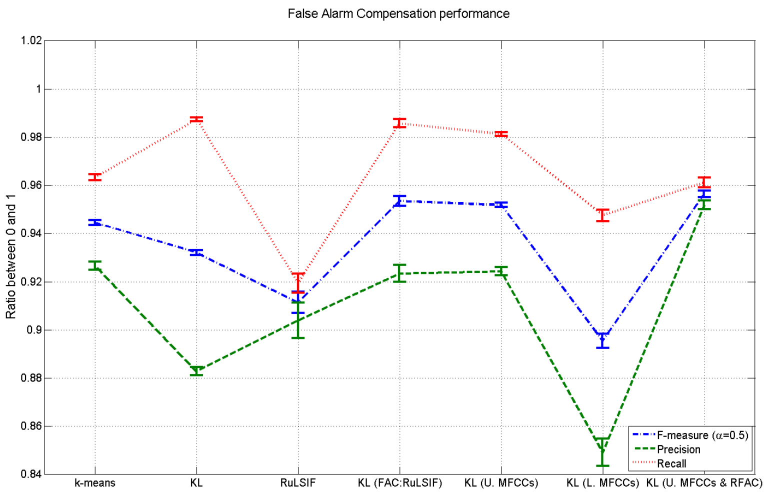

Figure 5.8: This figure shows the FAC performance of the RFAC KL method compared to all previous methods. The result is generated using 500 draws of test 37 times, the data is represented as the mean and the standard error seen as errorbars.
As expected the RFAC method produces a far more well-rounded FAC performace, but unfortunately the overall performance gain is very slight up to an FAC F-measure of 0.956 from 0.952. Then again this is as expected - since the FAC step almost perfectly captures all false alarms - further improvement through a higher F-measure is negligible. The performance gain must come from a more extensive - not a more precise - application of FAC.

In figure 5.7 it is seen that RFAC slightly improves the results, but it also seems to indicate that the KL method is reaching a barrier to its performance. This barrier could be posed by the temporal parameters. This is discussed further in section 5.4 and section 5.3 provides circumstantial evidence.

All in all the RFAC method does improve the KL method over the standard FAC method, but the improvement gain is marginal. Interestingly RFAC almost always runs 4 levels for FAC, with 4 runs in 78% of the tests and was not seen using less than 3 or more than 6. Even if the computational cost of the FAC step is increased slightly the RFAC takes advantage of the cheap FAC of the KL approach, which as seen in section 4.1.3.2 is very fast. Efficiency wise the RFAC method is definitely worth it as the runtime is only increased very slightly, whereas the performance gain is slight but statistically significant.

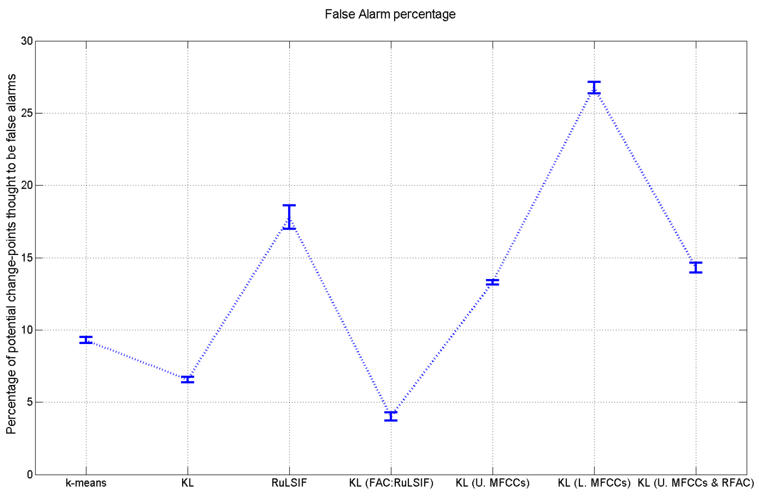

Figure 5.9: This figure shows the percentage of false alarms for the RFAC method. Unexpectedly the percentage of false alarms only went up very slightly, by about 1 percentage point, compared to the upper MFCC KL standard FAC method.

This section did not make it into the greater body of work since the effect of a set number of FAC runs, rather than a fully recursive one is left unexplored due to a lack of computational resources.

5.3 Mimicking news pod-cast data

Upon a closer inspection of the results using KL with RFAC and upper MFCCs, an interesting discovery was made. It was found that the method did surprisingly well, except in the case of very short speaker turns in rapid succession, here referring to speaker turns at a length of about 2 seconds. In the news pod-cast scenario this type of conversation is very rare and as seen in Jørgensen et al. [55] speaker segments below 5 seconds in general are uncommon.

As an experiment and to mediate better comparison with other work, the KL model already trained is simply fed data with speaker turns lengths always above 5 seconds. The result of this trial is seen in figure 5.10 and as seen the performance without short speaker turns is far greater than with. The F-measure increases to 0.940 with a precision of 94.3%, and a recall of 93.7%.

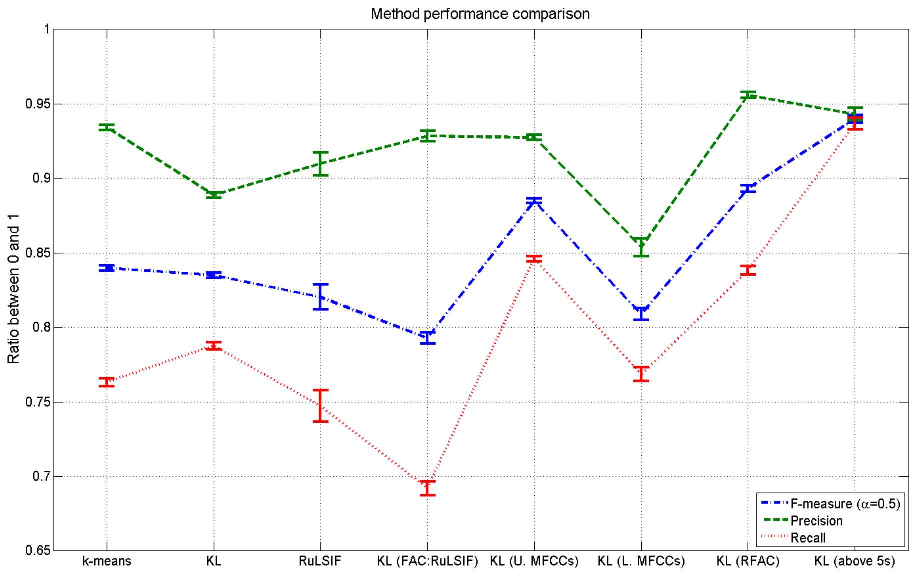

Figure 5.10: This figure shows that the performance of the KL method - using RFAC and upper MFCCs - is far greater if the speech data does not contain short speaker turns (<5s). With the F-measure increasing to 0.940 having a precision of 94.3%, and a recall of 93.7%.

This result is very promising as the optimisation method designed in this thesis could be exported onto commercial software. The commercial software could then train the algorithm to its particular speech data and performance could far exceed the general case performance.

This section is not admitted into the greater body of work, as mimicking the distribution of speaker turn lengths found in news pod-casts, goes far beyond the fairly arbitrary limit of min. 5 seconds of speech.

5.4 Promising avenues

This section will quickly enumerate promising unexplored avenues in bullet-point form.

- In section 2.2.3 several methods for reducing the effect of noise were investigated. Beyond this, code was developed for generating data immersed in varying SNR of several noise types, including babble noise, white noise and potentially noise caused by phone-line compression. As was mentioned MFCCs have been found to handle noise poorly and this might reveal interesting strengths and weaknesses in the various approaches. For instance section 3.2.2.2 mentions research where KL was found to be weak in a noisy environment.
- During the RuLSIF implementation, see section 3.2.3.3, the smoothing parameter α_R was left at its default setting. An optimisation of this parameter might increase the performance of RuLSIF, however as RuLSIF's weakness lies in its efficiency this avenue was barely explored.
- More promisingly it was found that RuLSIF's efficiency could be optimised significantly in section 3.2.3.3. This would require tampering with its cross-validation step of the kernel width and its regularization parameter. This cross-validation is performed in every single density-ratio estimation individually, which is probably excessive, as test revealed that the range of values selected was quite tight. If this cross-validation step could be skipped entirely its efficiency could be improved by up to a factor of 4.
- The change-point detection algorithm, the FAC algorithm and the MFCC calculation have a number of temporal parameters, in this thesis all these were borrowed from previous work. The majority of these were optimised by Jørgensen et al. [55] for a vector quantization based approach. As the selected method in this thesis is not based on vector quantization, a tuning of these parameters might increase performance.

- In section 2.4 alternatives to the FAC based SCD was mentioned. In section 4.1.3.1 results indicated that a combination of the ensemble based approach with with the FAC based approach might be promising.
- In section 4.1.3.1 it was hypothesized that RuLSIF might benefit significantly efficiency wise, if the optimisation algorithm from section 3.3 included efficiency in its optimisation goal.
- Finally in section 4.4 it is concluded that a full backwards feature selection of KL using the optimisation technique developed in section 3.3 would likely result in a marked improvement. This would however be a very costly process computationally.

Chapter 6

Final conclusions

This thesis has investigated, implemented, contrasted and combined a wide range of methods for speaker change detection using segmentation, and subsequently selected a method on which novel improvements have been implemented. These improvements revolve around a custom parameter optimisation technique for automatic model selection, a fully recursive speaker change detection paradigm and a model of the human speech production system. In addition the field of speaker clustering has been propped to fully resolve the problem of speaker diarisation, the question of "who spoke when?".

The investigated speaker change detection methods were drawn from the fields of vector quantization, Gaussian processes and relative density-ratio estimation, a novel approach from general change-detection theory not previously applied to speaker change detection. These methods mainly included an optimised K-means algorithm, the Kullback-Leibler distance, KL, and Relative unconstrained Least-Squares Importance Fitting, RuLSIF, respectively.

As the available speech corpora, ELSDSR & TIMIT, are designed for speech recognition, a novel method for generating speaker change detection data from ELSDSR was designed. This method solves the issues of file splicing, of non-uniform speech sample lengths and of the limited combinatorial nature of the corpora using a custom algorithm based on data bootstrapping. This method is designed to test the methods to their limit, capable of generating a broad range of change-point sequence types; anything from rapid-fire arguments to long monologues.

A broad search into the feature extraction techniques commonly applied to speaker change detection is conducted. Through a literary study a list of requirements for the feature set is established. It is found that short-term spectral features meet all requirements, yield decent performance and are easy to compute. Within short-term spectral features, particularly the Mel-Frequency Cepstral Coefficients are selected as they field a sound theoretical basis. The concept is to deconvolve the person specific excitation signal of the voicebox from the literal-semantic specific filtering of the vocal tract. The literary study found the amount of the Mel-Frequency Cepstral Coefficients, the use of derivatives and their placement in the cepstrum to be very diverse. This thesis employs several stages of feature selection to deal with this issue.

K-means, KL and RuLSIF, among others, were compared to various degrees of precision before these main contenders remained. Along with these a hybrid combination of KL and RuLSIF was designed to incorporate both the efficiency of KL and the thoroughness of RuLSIF. These 4 methods were fed through the custom optimisation algorithm using training data and tested on test data.

Out of the resulting optimised methods K-means fielded the highest F-measure of 0.84, with 93.4% of hypothesised change-point being true and finding 76.3% of all possible change-points. Whereas RuLSIF and the RuLSIF/KL hybrid failed to utilise the optimisation algorithm fully, due to efficiency issues. The KL method however came in as a close second, fielding an F-measure only 0.05 lower than K-means using a more well-rounded approach. KL only had a precision of 88.9%, but succeeded in locating 78.8% of the total change-points, meaning that KL in spite of being the simplest method displayed the greatest potential for improvement. Beyond this the KL approach has a closed form solution and therefore clocks in at 2.5 times faster than K-means despite the efficiency improvements reducing the runtime of K-means 15-fold, with KL running at 5 times real-time. Due to efficiency, the possible hidden potential and the close performance scores, KL was chosen.

KL subsequently underwent a series of improvements involving a backwards feature selection based on the model of the human speech production system and a fully recursive false alarm compensation paradigm. This process resulted in an F-measure of 0.893. This improvement comes from an increased precision up to 95.6%, with the method finding 83.8% of change-points, at almost no decrease in efficiency.

Upon close inspection almost all missed change-points were found in rapid-fire speech, with speaker-turns of about 2 seconds, in which case the algorithm often combined multiple speaker changes into a single speaker change. Using the same parameter set the F-measure increases to 0.940 using data with speaker turns always above 5 seconds, here with a precision of 94.3%, while locating 93.7% of all change-points. This result is very promising as the optimisation method designed in this thesis could be exported onto commercial software. The commercial software could then train the algorithm to its particular speech data automatically and performance could far exceed the general case performance.

Appendix A

MATLAB Code

For the sake of transparency, all essential functions for reproducing results are included here. Peripheral code such as toolboxes, plotting functions, file management, scripts, etc. are excluded.

A.1 Main Scripts

A.1.1 Main: Speaker Change Detection

```
%% enable wrapper function
if exist('wrapper_ON')
    wrapper_OFF = 0;
    enable_data_plots = 0;
else
    clc; clear all; close all;
    wrapper_OFF = 1;
    change_points = 50;
    if change_points > 100
        enable_data_plots = 0;
    else
```

```
        enable_data_plots = 1;
    end
    RandStream.setDefaultStream ...
        (RandStream('mt19937ar','seed',sum(100*clock)));
end

%% Change Point Detection parameters
% Parameters (values taken from Lasse section 4.4.4, law
   = 3s, Ti = 2s, Tmax = 8s, and ls = 0.1s.)
l_aw = 3;
l_s = 0.1;
Tmax = 8;
Ti = 2; % the width of a changepoint, another chanpoint
   must be outside this width. It is ofcause centered on
   the changepoint
buffer = Ti/2; %dont use data within x seconds of
   surrounding change points for false alarm compensation
if wrapper_OFF == 1
    alpha_cd.VQD          = 1.113;%56: 1.096; %25: 1.113;
    alpha_cd.KL           = 1.703;%lowermfcc 1.6656%
        uppermfcc 1.6621;%plain 1.703;%rulsif 2.0003;
    alpha_cd.DSD          = 1.3;
    alpha_cd.RuLSIF       = 1.483;

    alpha_fac.VQD         = 0.9085;%56: 0.9155; %25:
        0.9085
    alpha_fac.KL          = 0.4324;%lowermfcc 0.6892%
        uppermfcc 0.6643;%plain 0.4324;%rulsif 2.0500;
    alpha_fac.DSD         = 0.4;
    alpha_fac.RuLSIF      = 0.95;
end

%% samplerate
fs = 16000;

%% Change point detection methods
if wrapper_OFF == 1
    method_list = {'k-means','KL','DSD','RuLSIF'};
end

%% False alarm compensation methods
% CP method order: 'k-means' 'KL' 'DSD' 'RuLSIF'

fac_methods = {'k-means','KL','DSD','RuLSIF'};
```

```
if wrapper_OFF == 0
    fac_methods{strcmp(method_list,{'k-means','KL','DSD',
        'RuLSIF'})} = fac_method;
end
%% Read data
type = 'train';
shortest_length = 1;
combine_method = 'LHKs';

[data time_info] = combineelsdsr(change_points,
    wrapper_OFF,type,shortest_length,fs,combine_method);

%% Read changepoints
Read_actual_changepoints_section

%% Remove non-speech parts
%       Remove_nonspeech_parts_section %inactive

%% add noise
%       input_noise_file = 'babble';
%       SNRdB = 100;
%       add_noise

%% Feature extraction
Feature_extraction_section

%% Divide ceps into segments
ceps_segmentation_section

%% VQD using K-means
if sum(strcmp(method_list,'k-means'))~=0

    disp('WARNING: k_means hardcoded to only use first 12
        MFCCs, also in FAC!')
    ceps_temp = ceps;
    ceps = ceps(:,1:12);
    global K
    K = 25;

    Kmeans_VQD_section

    ceps = ceps_temp;
end
```

```

%% KL & DSD using single multivariant gaussian model
KL_DSD_section

%% RuLSIF
if sum(strcmp(method_list,'RuLSIF'))~=0
    alpha = .1; % Smoothing parameter
    sigma_list = 0.2:0.4:2; % Kernel Width
    lambda_list = -6:1:-2; % Reguralization parameter
    n = seg_length;
    step = seg_shift_length;
    k_folds = 5;
    RuLSIF_section
end

%% Calculate timestamps
x_axis = (seg_start_samples(1:end-seg_overlap)+(
    seg_length-1))./mfcc_opts.mfccprsec; %after the
    segment since the changepoint is measured inbetween 2
    segments

%% mark change points (Calculate VQD_n,avr & th_cd,n (
    Lasse's report page 62))
Mark_change_points_section

%% False alarm compensation (VQD)
any_FAC_updates=1;
any_FAC_updates_count = 1;
while(any_FAC_updates~=0) %Keeps running untill no
    changepoints are marked as false
    FAC_section
    any_FAC_updates_count = any_FAC_updates_count + 1;
end

%% f-measure
F_measure_section

%% FAC performance
FAC_performance_section

%% Plotting
if wrapper_OFF == 1
    main_fig = figure;

    if enable_data_plots == 1;
        % Compute and plot Spectrogram of data
        Spectrogram_section

        % plot MFCCs
        Feature_plotting_section
    end

    % plot Metrics and CPs
    Metric_plotting_section
end

%% ROC
if wrapper_OFF == 1
    ROC_section
end
```

A.1.2 Main: Speaker Clustering

```
clc; clear all; close all;
fs = 16000;
Tmax = 8;
runs = 5;

figure
for n_out = 1:runs
    %% Read data
    N_segment = 500;
    N_people = 5; %training max 23 %obs, simply wanting
        all the data is not possible (since i also use
        subsections of data as individual data) %obs,
        since the data is drawn randomly the dataset might
         not contain the full amount of speakers
    type = 'train';
    shortest_length = 3;
    fs_out = 16000;
    [input chosen_names] = elsdsr_clustering(N_segment,
        N_people,type,shortest_length,fs_out);

    %% Feature extraction
    for n = 1:size(input,1)
        data = input{n,1};
        Feature_extraction_section
```

```matlab
        Tmax_influence = Tmax*mfcc_opts.mfccprsec;
        input{n,3} = ceps;
    end

    input_orig = input(randperm(size(input,1)),:); % shuffle input

    %% clustering metric
    method_list = {'KL','DSD','k-means','RuLSIF','BIC','ICR'};

    for n_method = 1:numel(method_list)
        method = method_list{n_method};
        input = input_orig;

        figure(n_method)
        indexes = 1:N_segment;
        n = 1;
        while(1)

            %% compute dist matrix
            dist_matrix = 1e6.*ones(size(input,1)); % construct distance matrix
            %           wbar = waitbar(0,['Clustering using ' method]);
            lower_triangle_and_diag = 1;
            for row = 1:size(input,1)
                for col = (1+lower_triangle_and_diag):size(input,1)

                    A = input{row,3};
                    B = input{col,3};

                    if size(A,1)>Tmax_influence || size(B,1)>Tmax_influence
                        if size(A,1)>Tmax_influence
                            tmp = randperm(size(A,1));
                            A = A(tmp(1:Tmax_influence),:);
                        end
                        if size(B,1)>Tmax_influence
                            tmp = randperm(size(B,1));
                            B = B(tmp(1:Tmax_influence),:);
                    end

                    dist_matrix(row,col) =
                        compare_segments(A,B,method);
                else
                    dist_matrix(row,col) =
                        compare_segments(A,B,method);
                end

            end
            lower_triangle_and_diag =
                lower_triangle_and_diag+1;
        end

        %% find more common segments
        [mins(n),I] = min(dist_matrix(:));
        [row,col] = ind2sub(size(dist_matrix),I);

        %% combine likely condidates
        number_added(n) = sum(input{col,2});
        Z(n,:) = [indexes(row) indexes(col) mins(n)];

        input{row,1} = [input{row,1}; input{col,1}];
        input{row,2} = input{row,2} + input{col,2} ;
        input{row,3} = [input{row,3}; input{col,3}];
        indexes(row) = N_segment+n;

        input(col,:) = [];
        indexes(col) = [];

        %% purity
        purity = zeros(size(input,1),1);
        for n2 = 1:size(input,1)
            purity(n2) = max(input{n2,2})/sum(input{
                n2,2});
        end
        purity_avg(size(input,1),n_out,n_method) =
            mean(purity);
        clear purity

        %% cluster sizes
        sizes = zeros(size(input,1),1);
        for n2 = 1:size(input,1)
            sizes(n2) = sum(input{n2,2});
            end
            min_clust_size(size(input,1),n_out,n_method)
                = min(sizes);
            max_clust_size(size(input,1),n_out,n_method)
                = max(sizes);
            avg_clust_size(size(input,1),n_out,n_method)
                = mean(sizes);

            if sum(sizes > 1) == N_people
                save_sizes{n_method} = sizes;
                valid_clust(n) = 1;
            else
                valid_clust(n) = 0;
            end

            %% break condition
             if(size(input,1)==1)
                break;
            end

            %% plot
            dist_matrix(dist_matrix==1e6)=max(dist_matrix
                (dist_matrix~=1e6));
            imagesc(dist_matrix,[min(dist_matrix(:)) max(
                dist_matrix(:))])
            colorbar
            drawnow;

            %loop counter
            n=n+1;
        end
        close(n_method)
        %%
        if runs == 1

            if(min(Z(:,3)) < 0)
                Z(:,3) = Z(:,3) - min(Z(:,3)) + 1;
                neg = 1;
            else
                neg = 0;
            end

            if sum(valid_clust) > 0
                index = max(find(valid_clust));
```

```
    threshold = Z(index+1,3)*1.001;
else
    disp('Using default threshold')
    threshold= .7*(max(Z(:,3)));
end

if strcmp(method_list{n_method},'ICR')
    Z(:,3) = log10(Z(:,3));
    threshold = log10(threshold);
    logy = 1;
else
    logy = 0;
end

font = 13;
figure
[H,T,perm] = dendrogram(Z,0,'colorthreshold',threshold);%'default');
set(H,'LineWidth',2)
hold all
hleglines = [];
for n2 = 1:N_people
    dots = [];
    for n = 1:N_segment
        index = perm(n);
        if input_orig{index,2}(n2) == 1
            dots = [dots n];
        end
    end
    tmp = plot(dots,zeros(1,length(dots))+0.001*max(Z(:,3)),'.','markersize',20);
    legends{n2} = ['Speaker #' num2str(n2)];
    hleglines = [hleglines tmp];
end
plot(1:N_segment,threshold.*ones(1,N_segment),'k-.','linewidth',3)

ylabl = method_list{n_method};
if neg == 1
    ylabl = [ylabl '+ (-min(' ylabl ')) + 1'];
end
if logy == 1
```

```
                ylabl = ['log10(' ylabl ')'];
            end

            ylabel(['Cluster dissimilarity [' ylabl ']'],
                'fontsize',font)
            xlabel('Full Speaker segment index (order
                irrelevant)','fontsize',font)
            title(['Dendrogram of full speaker segments
                using Agglomerative Hierarchical
                Clustering with ' method_list{n_method} '
                cluster dissimilarity measure'],'fontsize'
                ,font)
            legend(hleglines,legends,'Location','
                northwest');
            set(gca,'fontsize',13)
        end
    end
end
```

A.2 Parameter Optimisation

A.2.1 Main: Parameter Optimisation

```
clc; clear all; close all;

RandStream.setDefaultStream ...
    (RandStream('mt19937ar','seed',sum(100*clock)));

resolutions = [10 5];
data_amounts = [500 1000];
remove_bound_percentage = 50;
cd_method = 'KL';
fac_method = 'KL';
reruns = 3;

no_fac = 0;
maxiter = 30;

cd_bound  = [1.0  2.5];
fac_bound = [0.2  1.5];

```

```
resolutions = round(linspace(resolutions(1),resolutions
    (2),reruns))
data_amounts = round(linspace(data_amounts(1),
    data_amounts(2),reruns))

cd_center  = mean(cd_bound);
fac_center = mean(fac_bound);
cd_range   = range(cd_bound);
fac_range  = range(fac_bound);

ID = floor(rand*1e6)+1;
disp(ID)

figure
for rerun = 1:reruns

    resolution  = resolutions(rerun);
    data_amount = data_amounts(rerun);

    cd_range    = range(cd_bound)*(remove_bound_percentage
        /100)^(rerun-1);
    fac_range   = range(fac_bound)*(
        remove_bound_percentage/100)^(rerun-1);

    cd_bound_current(1) = cd_center-cd_range/2;
    cd_bound_current(2) = cd_center+cd_range/2;

    if cd_bound_current(1)<cd_bound(1)
        cd_bound_current = cd_bound_current+(cd_bound(1)-
            cd_bound_current(1));
    elseif cd_bound_current(2)>cd_bound(2)
        cd_bound_current = cd_bound_current-(
            cd_bound_current(2)-cd_bound(2));
    end

    fac_bound_current(1) = fac_center-fac_range/2;
    fac_bound_current(2) = fac_center+fac_range/2;

    if fac_bound_current(1)<fac_bound(1)
        fac_bound_current = fac_bound_current+(fac_bound
            (1)-fac_bound_current(1));
    elseif fac_bound_current(2)>fac_bound(2)
        fac_bound_current = fac_bound_current-(
            fac_bound_current(2)-fac_bound(2));
    end

    cd   = linspace(cd_bound_current(1),cd_bound_current(2),resolution)
    fac  = linspace(fac_bound_current(1),fac_bound_current(2),resolution)

    if no_fac == 1
        fac = fac(1);
    end

    gdvalues = {cd fac};

    if rerun == reruns && maxiter~=0
        fminON = 1;
    else
        fminON = 0;
    end

    addarg = {cd_method, fac_method, data_amount};
    [x cost fval exitflag output x_all fval_all] = simpgdsearch('objfun',gdvalues,addarg,'temp.txt',fminON,maxiter);

    cd_center  = x(1);
    fac_center = x(2);
    %% save intermediate calcs
    mat_name = ['cd method ' cd_method ' fac method ' fac_method ' run ' num2str(rerun) ' out of ' num2str(reruns) ' ID ' num2str(ID) ];
    save(mat_name)

    %% Plot results
    if exist('no_fac') == 1 && no_fac == 1
        plot(cd,1-cost,'-o','linewidth',2)
        hold all
        xlim([cd_bound])
        ylim([0 1])
        if rerun == reruns
            plot(x_all(:,1),1-fval_all,'-o','linewidth',2)
            plot(x(:,1),1-fval,'-x','linewidth',10)
        end
    else
```

```
        optim_parameter_plotting_section
    end
end
```

A.2.2 Objective Function

```
function f = objfun(parameters,addarg)

f = 1-Main_wrapper(addarg{1},parameters(1),parameters(2),
    addarg{3},addarg{2});
```

A.2.3 Parameter Optimisation

```
function [x cost minval exitflag output x_all fval_all] =
     simpgdsearch(objfun,gdvalues,addarg,fileout,fminON,
    maxiter)
%% number of dimensions
nd = size(gdvalues,2);

%% create parameter step
for k = 1:nd
    paramstep(k,:) = gdvalues;
end

%% model: all the combination of the evaluated parameters
str = [];
for k = 1:nd
    str = [str 'gdvalues{' num2str(k) '},'];
end
str(length(str)) = [];
eval(['model = setprod(' str ');']);

%% grid search
time = 0;
cost = zeros(1,size(model,1));
wbar = waitbar(0,'Please wait..');
for l = 1:size(model,1)
    internal_time = tic;

    if(l~=1)
        average_runtime = time/(l-1);
        runs_left = size(model,1)-(l-1);
        time_left = runs_left*average_runtime;
        time_left_string = [' Time left: ' secs2hms(time_left)];
        waitbar(l/size(model,1),wbar,sprintf('%s',time_left_string))
        disp(time_left_string)
    end

    cost(l) = feval(objfun,model(l,:),addarg);

    time = time + toc(internal_time);
end
delete(wbar)

%% best fit model %%
[minval ind] = min(cost);
x_grid = model(ind,:);
x = x_grid;

exitflag=[];
output=[];
if fminON ==1
    %% Simplex method %%
    options = optimset('Display','iter','TolX',0.01,'TolFun',1e10,'MaxIter',maxiter,'OutputFcn',@outfun); %
    [x,fval,exitflag,output,x_all,fval_all]  = fminsearch(objfun,x,options,addarg);

    x_all = [x_grid; x_all];
    fval_all = [minval; fval_all];

    minval = fval;

    %% save data %%
    if nargin == 4
        foo = [model, cost'];
        foo = [foo; x,fval];
        save(fileout,'foo','-ascii');
    end
else
    x_all = x;
    fval_all = minval;
end
```

A.3 SCD Methods

A.3.1 Main: Gaussian approach section

```
if sum(strcmp(method_list,'KL'))~=0 || sum(strcmp(
    method_list,'DSD'))~=0
    % ML est of single gaussian on each segment
    if wrapper_OFF == 1
        wbar = waitbar(0,'ML estimate of sigma and mu');
    end
    ML_est_timer = tic;
    sigma = zeros(size(ceps,2),size(ceps,2),length(
        seg_start_samples));
    mu = zeros(size(ceps,2),length(seg_start_samples));

    for n_seg = 1:length(seg_start_samples)
        if wrapper_OFF == 1
            waitbar(n_seg/length(seg_start_samples));
        end

        ceps_seg = ceps_seg_func(n_seg,seg_start_samples,
            ceps,seg_length);
        sigma(:,:,n_seg) = cov(ceps_seg);
        mu(:,n_seg) = mean(ceps_seg);
    end
    if wrapper_OFF == 1
        delete(wbar)
    end
    time.ML_est = toc(ML_est_timer);
end

if sum(strcmp(method_list,'KL'))~=0
    % Kullback Leibler divergence
    if wrapper_OFF == 1
        wbar = waitbar(0,'KL');
    end
    KL_timer = tic;
    KLs = zeros(length(seg_start_samples)-seg_overlap,1);
    for n_seg = 1:length(KLs)
        if wrapper_OFF == 1
            waitbar(n_seg/length(KLs));
        end

        KLs(n_seg) = KL(sigma(:,:,n_seg),sigma(:,:,n_seg+seg_overlap),mu(:,n_seg),mu(:,n_seg+seg_overlap));
    end
    time.KL = toc(KL_timer);
    if wrapper_OFF == 1
        delete(wbar)
    end
end

if sum(strcmp(method_list,'DSD'))~=0
    % Divergence Shape Distance
    if wrapper_OFF == 1
        wbar = waitbar(0,'DSD');
    end
    DSD_timer = tic;
    DSDs = zeros(length(seg_start_samples)-seg_overlap,1);
    for n_seg = 1:length(DSDs)
        if wrapper_OFF == 1
            waitbar(n_seg/length(DSDs));
        end

        DSDs(n_seg) = DSD(sigma(:,:,n_seg),sigma(:,:,n_seg+seg_overlap));
    end
    time.DSD = toc(DSD_timer);
    if wrapper_OFF == 1
        delete(wbar)
    end
end
```

A.3.2 Kullback Leibler distance

```
function out = KL(sigma_A,sigma_B,mu_A,mu_B)
%KL: Kullback-Leibler Distance

out = 1/2*trace((sigma_A - sigma_B)*(inv(sigma_B) - inv(sigma_A)))+...
        1/2*trace((inv(sigma_A) + inv(sigma_B))*(mu_A - mu_B)*(mu_A - mu_B)'));

end
```

A.3.3 Divergence Shape Distance

```
function out = DSD(sigma_A,sigma_B)

out = 1/2*trace((sigma_A - sigma_B)*(inv(sigma_B) - inv(
    sigma_A)));

end
```

A.3.4 Main: Vector Quantization section

```

%% Calculate codebooks
warning('off','all');
if wrapper_OFF == 1
    wbar = waitbar(0,'K-means');
end
kmeans_timer = tic;
kmeans_opts = statset('MaxIter',10000);

centroids = zeros(K,size(ceps,2),length(seg_start_samples
    ));
for n_seg = 1:length(seg_start_samples)
    if wrapper_OFF == 1
        waitbar(n_seg/length(seg_start_samples));
    end

    ceps_seg = ceps_seg_func(n_seg,seg_start_samples,ceps
        ,seg_length);

    % Optimized by passing on old centroids, ~2.5 times
        faster. (OBS! This also smoothes the result, thus
        degrading it.)
    if n_seg ~= 1
%           [IDX,centroids(:,:,n_seg)] = kmeans(ceps_seg
    (:,:,n_seg),K,'Options',kmeans_opts,'emptyaction','
    singleton','start',centroids(:,:,n_seg-1));
        [IDX,centroids(:,:,n_seg)] = kmeans(ceps_seg,K,'
            Options',kmeans_opts,'emptyaction','singleton'
            ,'onlinephase','off','start',centroids(:,:,
            n_seg-1));
    else
```

```
%           [IDX,centroids(:,:,n_seg)] = kmeans(ceps_seg
    (:,:,n_seg),K,'Options',kmeans_opts,'emptyaction','
    singleton');
          [IDX,centroids(:,:,n_seg)] = kmeans(ceps_seg,K,'
              Options',kmeans_opts,'emptyaction','singleton'
              ,'onlinephase','off');
    end
end
time.kmeans = toc(kmeans_timer);
if wrapper_OFF == 1
    delete(wbar)
end
warning('on','all');

%% Calculate distortion measure

% VQD (Optimized) ~7 times faster
VQDs_optim = zeros(length(seg_start_samples)-seg_overlap
    ,1);
if wrapper_OFF == 1
    wbar = waitbar(0,'Vector Quantization Distortion
        Calculation');
end
VQD_timer = tic;
for n_seg = 1:length(VQDs_optim)
    if wrapper_OFF == 1
    waitbar(n_seg/length(VQDs_optim));
    end

    VQDs_optim(n_seg) = VQD_optim(centroids(:,:,n_seg),
        centroids(:,:,n_seg+seg_overlap));
end
time.VQDs_optim = toc(VQD_timer);
if wrapper_OFF == 1
    delete(wbar)
end
```

A.3.5 Vector Quantization Distortion

```
function out = VQD_optim(CA,B)
%VQD: Vector Quantization Distortion
%
% CA : Is a kmeans codebook on the previous segment
% B   : Is either the full next segment or the codebook of
    the next segment
%
% out: The calculated VQD
%
% Both CA and B must have features in colomns and vectors
    in rows

out = 0;
for n = 1:size(B,1)

    tmp = repmat(B(n,:),size(CA,1),1);

    tmp = CA-tmp;

    dists = sqrt(sum(tmp.^2,2));

    out = out + min(dists);
end
end
```

A.3.6 Main: Relative Density Ratio Estimation

```
addpath('RuLSIF_change_detection')
addpath('RuLSIF_change_detection\RuLSIF')

y = ceps';

RuLSIF_timer = tic;

[score1, sigma_chosen1, lambda_chosen1] =
    change_detection(y              ,n,1,step,alpha,'RuLSIF
    Forward'  ,k_folds,sigma_list,lambda_list,wrapper_OFF);
[score2, sigma_chosen2, lambda_chosen2] =
    change_detection(y(:,end:-1:1),n,1,step,alpha,'RuLSIF
    Backward',k_folds,sigma_list,lambda_list,wrapper_OFF);
score2 = score2(end:-1:1);

RuLSIFs = (score1 + score2)';

time.RuLSIF = toc(RuLSIF_timer);
```

A.3.7 Change Detectoion front-end for RuLSIF

```
function [SCORE, sigma_chosen, lambda_chosen] =
    change_detection(X, n, k, step, alpha, task,k_folds,
    sigma_list,lambda_list,wrapper_OFF)

SCORE=[];

WIN = sliding_window(X, k, 1);
nSamples = size(WIN,2);
t = n+1;
if wrapper_OFF == 1
    wbar = waitbar(0,task);
end

next_percentage = 0;
loopcount = 1;
while(t + n -1 <= nSamples)
    if wrapper_OFF == 1
        waitbar(((t - n)-1) / (nSamples-(n-1)));
    end

    new_percentage = floor(((t - n)-1) / (nSamples-(n-1))
        *100);

    if new_percentage == next_percentage
    disp([task ' ' num2str(new_percentage) '%'])
    next_percentage = next_percentage +1;
    end

    YTest = WIN(:, t : n + t -1 );
    YRef = WIN(:, t-n: t-1);
%       [pe]=RelULSIF(YRef,YTest,[],[],alpha,k_folds,
    sigma_list,lambda_list);
    [rPE, ~, ~, sigma_chosen(loopcount), lambda_chosen(
        loopcount)] = RelULSIF(YRef,YTest,[],[],alpha,
        k_folds,sigma_list,lambda_list);
    SCORE = [SCORE, rPE];
    t = t + step;
    loopcount = loopcount + 1;
end
if wrapper_OFF == 1
    delete(wbar)
end
```

```
end
```

A.3.8 RuLSIF

```
function [rPE, g_nu, g_re, sigma_chosen_index,
    lambda_chosen_index] = RelULSIF( x_de, x_nu, x_re,
    x_ce, alpha, k_folds,sigma_list,lambda_list)
%rng default
fold = k_folds;
[~,n_nu] = size(x_nu);
[~,n_de] = size(x_de);

% Parameter Initialization Section
if nargin < 4 || isempty(x_ce)
     b = min(100,n_nu);
     idx = randperm(n_nu);
     x_ce = x_nu(:,idx(1:b));
end

if nargin < 5
     alpha = 0.5;
end

% construct gaussian centers
[~,n_ce] = size(x_ce);
% get sigma candidates
x = [x_de,x_nu];
med = comp_med(x);
% sigma_list = med * [.6,.8,1.0,1.2,1.4];
sigma_list = med * sigma_list;
% get lambda candidates
% lambda_list = 10.^(-3:1:1);
lambda_list = 10.^lambda_list;

[dist2_de] = comp_dist(x_de,x_ce);
%n_de * n_ce
[dist2_nu] = comp_dist(x_nu,x_ce);
%n_nu * n_ce

%The Cross validation Section Begins
score = zeros(length(sigma_list),length(lambda_list));
for i = 1:length(sigma_list)
     k_de = kernel_gau(dist2_de,sigma_list(i));
```

```matlab
    k_nu = kernel_gau(dist2_nu,sigma_list(i));
    for j = 1:length(lambda_list)

        cv_index_nu=randperm(n_nu);
        cv_split_nu=floor([0:n_nu-1]*fold./n_nu)+1;
        cv_index_de=randperm(n_de);
        cv_split_de=floor([0:n_de-1]*fold./n_de)+1;

        sum = 0;
        for k = 1:fold
            k_de_k = k_de(cv_index_de(cv_split_de~=k),:)';
            %n_ce * n_de
            k_nu_k = k_nu(cv_index_nu(cv_split_nu~=k),:)';
            %n_ce * n_nu

            H_k = ((1-alpha)/size(k_de_k,2))*(k_de_k*k_de_k') + ...
                (alpha/size(k_nu_k,2))*(k_nu_k*k_nu_k');
            h_k = mean(k_nu_k,2);

            theta = (H_k + eye(n_ce)*lambda_list(j))\h_k;
            %theta = max(theta,0);

            k_de_test = k_de(cv_index_de(cv_split_de==k),:)';
            k_nu_test = k_nu(cv_index_nu(cv_split_nu==k),:)';
            % objective function value
            J = alpha/2 * mean((theta' * k_nu_test).^2)+ ...
                (1-alpha)/2*mean((theta'*k_de_test).^2)- ...
                mean(theta' * k_nu_test);
            sum = sum + J;
        end
        score(i,j) = sum/fold;
    end
end

%find the chosen sigma and lambda
[sigma_chosen_index,lambda_chosen_index] = find(score==min(score(:)));
```

```
sigma_chosen = sigma_list(sigma_chosen_index);
lambda_chosen = lambda_list(lambda_chosen_index);

%compute the final result
k_de = kernel_gau(dist2_de',sigma_chosen);
k_nu = kernel_gau(dist2_nu',sigma_chosen);

H = ((1-alpha)/n_de)*(k_de*k_de') + ...
     (alpha/n_nu)*(k_nu*k_nu');
h = mean(k_nu,2);

theta = (H + eye(n_ce)*lambda_chosen)\h;

g_nu = theta'*k_nu;
g_de = theta'*k_de;
g_re = [];
if ~isempty(x_re)
     dist2_re = comp_dist(x_re,x_ce);
     k_re = kernel_gau(dist2_re', sigma_chosen);
     g_re = theta'*k_re;
end
rPE = mean(g_nu) - 1/2*(alpha*mean(g_nu.^2) + ...
     (1-alpha)*mean(g_de.^2)) - 1/2;

end
```

A.4 False Alarm Compensation

A.4.1 Main: False Alarm Compensation section

```
if(buffer>(Ti/2))
     error('buffer should be smaller than or equal to Ti/2
         ')
end

any_FAC_updates = 0;

if sum(strcmp(method_list,'k-means'))~=0 && ~strcmp(
    fac_methods{1},'')
     tic;
     [ CP.VQD      FAC_change.VQD     ] = FAC(ceps, th_cd.VQD
          , alpha_fac.VQD     , alpha_cd.VQD     , CP.VQD     ,
        length(seg_start_samples), buffer,
        seg_shift_length,mfcc_opts.mfccprsec,
        Tmax_influence,seg_start_samples,seg_length,
        seg_overlap,fac_methods{1},'k-means',wrapper_OFF);
    any_FAC_updates = any_FAC_updates + FAC_change.VQD;
    time.FAC.VQD(any_FAC_updates_count) = toc;
end
if sum(strcmp(method_list,'KL'     ))~=0 && ~strcmp(
    fac_methods{2},'')
    tic;
    [ CP.KL      FAC_change.KL     ] = FAC(ceps, th_cd.KL
          , alpha_fac.KL      , alpha_cd.KL      , CP.KL      ,
         length(seg_start_samples), buffer,
        seg_shift_length,mfcc_opts.mfccprsec,
        Tmax_influence,seg_start_samples,seg_length,
        seg_overlap,fac_methods{2},'KL'      ,wrapper_OFF);
    any_FAC_updates = any_FAC_updates + FAC_change.KL;
    time.FAC.KL(any_FAC_updates_count) = toc;
end
if sum(strcmp(method_list,'DSD'    ))~=0 && ~strcmp(
    fac_methods{3},'')
    tic;
    [ CP.DSD     FAC_change.DSD    ] = FAC(ceps, th_cd.DSD
         , alpha_fac.DSD     , alpha_cd.DSD     , CP.DSD     ,
        length(seg_start_samples), buffer,
        seg_shift_length,mfcc_opts.mfccprsec,
        Tmax_influence,seg_start_samples,seg_length,
        seg_overlap,fac_methods{3},'DSD'     ,wrapper_OFF);
    any_FAC_updates = any_FAC_updates + FAC_change.DSD;
    time.FAC.DSD(any_FAC_updates_count) = toc;
end
if sum(strcmp(method_list,'RuLSIF' ))~=0 && ~strcmp(
    fac_methods{4},'')
    tic;
    [ CP.RuLSIF FAC_change.RuLSIF] = FAC(ceps, th_cd.
        RuLSIF, alpha_fac.RuLSIF, alpha_cd.RuLSIF, CP.
        RuLSIF, length(seg_start_samples), buffer,
        seg_shift_length,mfcc_opts.mfccprsec,
        Tmax_influence,seg_start_samples,seg_length,
        seg_overlap,fac_methods{4},'RuLSIF' ,wrapper_OFF);
    any_FAC_updates = any_FAC_updates + FAC_change.RuLSIF
        ;
    time.FAC.RuLSIF(any_FAC_updates_count) = toc;
end
```

A.4.2 False Alarm Compensation

```matlab
function [ CPs change] = FAC(ceps, th_cd, alpha_fac, alpha_cd, CPs, seg_amount, buffer, seg_shift_length, mfccprsec,Tmax_influence,seg_start_samples,seg_length,seg_overlap,method,metric,wrapper_OFF)
if sum(CPs==1) == 0 %no change points found
    change = 0;
    return;
end

remember_CPs = CPs;

buffer_influence = buffer*mfccprsec/seg_shift_length;

%% Find maximal data segments on which methods are reevaluated
CP_indexes = [1; find(CPs==1); seg_amount]; %indexes match %OBS this sentence removes buffer zones at beginning and end automatically
[back forward indexes] = locate_intervals(CP_indexes, buffer_influence,Tmax_influence,seg_overlap);

%% Set threshold of which values based on more data has to exceed

if strcmp(method,metric)
    th_fac = alpha_fac.*(th_cd(CPs==1)./alpha_cd);
elseif sum(CPs==1) > 5
    midpoints = CP_indexes(1:end-1)+floor(diff(CP_indexes)/2);

    temp_back = [CP_indexes(1:end-1)+buffer_influence midpoints];
    temp_back((temp_back(:,2)-temp_back(:,1)) > Tmax_influence, 1) = temp_back((temp_back(:,2)-temp_back(:,1)) > Tmax_influence, 2)-Tmax_influence;

    temp_forward = [midpoints CP_indexes(2:end)-buffer_influence];
    temp_forward((temp_forward(:,2)-temp_forward(:,1)) > Tmax_influence, 1) = temp_forward((temp_forward(:,2)-temp_forward(:,1)) > Tmax_influence, 2)-
```

```
        Tmax_influence;

    temp_indexes = find((temp_back(:,1)<temp_back(:,2)) &
         (temp_forward(:,1)<temp_forward(:,2)));

    value = inline_FAC(temp_indexes, temp_back ,
        temp_forward ,ceps, th_cd, alpha_fac, alpha_cd,
        CPs, seg_amount, buffer, seg_shift_length,
        mfccprsec,Tmax_influence,seg_start_samples,
        seg_length,seg_overlap,method,metric,wrapper_OFF);

    th_fac = alpha_fac.*median(value).*ones(size(th_cd(
        CPs==1)));
else %too few hypothesized changepoints to estimate a
    threshold
    change = 0;
    return;
end

%% Calculate new values
value = inline_FAC(indexes, back, forward, ceps, th_cd,
    alpha_fac, alpha_cd, CPs, seg_amount, buffer,
    seg_shift_length,mfccprsec,Tmax_influence,
    seg_start_samples,seg_length,seg_overlap,method,metric
    ,wrapper_OFF);

%% Check if new values exceed set threshold
placeholder = CPs(CPs==1);
for n2 = 1:length(indexes)
    n = indexes(n2);
    if value(n) > th_fac(n)
        placeholder(n) = 1; % true changepoint
    else
        placeholder(n) = 2; % false changepoint
    end
end

%% Assign false positives to falsified changepoints
CPs(CPs==1) = placeholder;

%% Check if any false positives were found (if so it may
    be prudent to rerun this function)
if sum(remember_CPs ~= CPs) == 0 %identical
    change = 0;
```

```
else
    change = 1;
end

end

function [back forward indexes] = locate_intervals(
    CP_indexes,buffer_influence,Tmax_influence,seg_overlap
    )
back = zeros(length(CP_indexes)-2,2);
forward = back;
for n = (1+1):(numel(CP_indexes)-1)

    lowest_back = CP_indexes(n-1)+buffer_influence;

    higest_back = CP_indexes(n);

    if(higest_back-lowest_back > Tmax_influence)
        lowest_back = higest_back-Tmax_influence;
    end

    back(n-1,:) = [lowest_back higest_back];

    lowest_forward = CP_indexes(n)+seg_overlap;

    highest_forward = CP_indexes(n+1)+1-buffer_influence;
        %+1 since the previous segment has the same
        number as the current CP

    if(highest_forward-lowest_forward > Tmax_influence)
        highest_forward = lowest_forward+Tmax_influence;
    end

    forward(n-1,:) = [lowest_forward highest_forward];
end
indexes = find((back(:,1)<back(:,2)) & (forward(:,1)<
    forward(:,2)));

end

function value = inline_FAC(indexes, back, forward, ceps,
     th_cd, alpha_fac, alpha_cd, CPs, seg_amount, buffer,
    seg_shift_length,mfccprsec,Tmax_influence,
    seg_start_samples,seg_length,seg_overlap,method,metric
    ,wrapper_OFF)

if wrapper_OFF == 1
    wbar = waitbar(0,['FAC on ' metric ' using ' method])
        ;
end
for n2 = 1:length(indexes);
    n = indexes(n2);
    if wrapper_OFF == 1
        waitbar(n/length(back));
    end

    previous_segment = ceps(seg_start_samples(    back(n
        ,1)):(seg_start_samples(    back(n,2))+(seg_length
        -1)),:);
    next_segment     = ceps(seg_start_samples(forward(n
        ,1)):(seg_start_samples(forward(n,2))+(seg_length
        -1)),:);

    value(n) = compare_segments(previous_segment,
        next_segment,method);
end
if wrapper_OFF == 1
    delete(wbar)
end
if strcmp('k-means', method)
    warning('on','all');
end
end
```

A.4.3 FAC Performance

```
fac_performance.alpha_f_measure = 0.5;
methods = fieldnames(CP);
fac_performance.TP = zeros(numel(methods),1);
fac_performance.FP = zeros(numel(methods),1);
fac_performance.FN = zeros(numel(methods),1);
fac_performance.TN = zeros(numel(methods),1);

for n_method = 1:numel(methods)

    accepted = x_axis(CP.(methods{n_method})==1);
    for n_acp = 1:length(accepted)
```

```
        if sum((changepoints_true-Ti/2) < accepted(n_acp) & accepted(n_acp) < (changepoints_true+Ti/2)) ~=0;
            fac_performance.TP(n_method) = fac_performance.TP(n_method) + 1;
        else
            fac_performance.FP(n_method) = fac_performance.FP(n_method) + 1;
        end
    end

    rejected = x_axis(CP.(methods{n_method})==2);
    for n_rej = 1:length(rejected)
        if sum((changepoints_true-Ti/2) < rejected(n_rej) & rejected(n_rej) < (changepoints_true+Ti/2)) ~=0;
            fac_performance.FN(n_method) = fac_performance.FN(n_method) + 1;
        else
            fac_performance.TN(n_method) = fac_performance.TN(n_method) + 1;
        end
    end

    fac_performance.PRC.(methods{n_method}) = fac_performance.TP(n_method)/(fac_performance.TP(n_method)+fac_performance.FP(n_method));
    fac_performance.RCL.(methods{n_method}) = fac_performance.TP(n_method)/(fac_performance.TP(n_method)+fac_performance.FN(n_method));

    fac_performance.F_measure.(methods{n_method}) = (fac_performance.RCL.(methods{n_method})*fac_performance.PRC.(methods{n_method}))/...
        (fac_performance.alpha_f_measure*fac_performance.RCL.(methods{n_method})+(1-fac_performance.alpha_f_measure)*fac_performance.PRC.(methods{n_method}));
end
```

A.5 Data Handling

A.5.1 ELSDSR front-end for Data Splicing

```
function [data time_info data_orig] = combineelsdsr(N,
    wrapper_OFF,type,shortest_length,fs_out,combine_method
    )

if strcmp(type,'test')
    data_path = 'data\elsdsr\Speaker_Identification_Test\
        ';
elseif strcmp(type,'train')
    data_path = 'data\elsdsr\
        Speaker_Identification_Training\';
end

list = dir([data_path '*.wav']);

% all_data = cell(numel(list),1);
% for n = 1:numel(list)
%     [all_data{n}, fs] = isp_audioread([data_path list(n
    ).name]);
% end
% save([data_path 'all_data'],'all_data')

load([data_path 'all_data'],'all_data')

fs = 16000;
file_lengths = zeros(numel(list),1);
for n = 1:numel(list)
    file_lengths(n) = length(all_data{n})/fs;
end

temp_file_lengths(file_lengths~=max(file_lengths)) =
    file_lengths(file_lengths~=max(file_lengths));
longest_length = max(temp_file_lengths); %set at second
    largest (otherwise above that it will only have a
    single speaker to pick from)
% longest_length = 15

%choose random length between a and b
desired_file_lengths = shortest_length + (longest_length-
    shortest_length).*rand(N,1); % ones(N,1)*0.5
```

```matlab

%choose random sample longer than the random length
random_sample = zeros(N,1);
for n = 1:N
    random_sample(n) = randsample(find(file_lengths >=
        desired_file_lengths(n)),1);
end

list = list(random_sample);

if wrapper_OFF == 1
    wbar = waitbar(0,'Combining data');
end
for n = 1:N
    if wrapper_OFF == 1
        waitbar(n/N);
    end
    tmp_data = all_data{random_sample(n)};

    %cut it to the desired length
    desired_length = floor(desired_file_lengths(n)*fs);
    start = randi([1,length(tmp_data)-desired_length]);
    tmp_data = tmp_data(start:(start+desired_length));

    %resample (placed here to split up the calculation
        into a host of small ones)
    if fs ~= fs_out
        tmp_data=resample(tmp_data,fs_out,fs);
    end

    if n == 1
        data = zeros(N*15*fs,1);
        data_length = length(tmp_data);
        data(1:data_length) = tmp_data;
    else
        [data data_length] = splice_data( data, tmp_data,
            50, combine_method, data_length);
    end

    if n ~= N
        %           time = numel(data)/fs_out;
        time = data_length/fs_out;

        minutes = floor(time/60);
        seconds = floor((time/60-minutes)*60);
        subseconds = floor(((time/60-minutes)*60-seconds)*100);

        %check whether it is a true change point
        if (strcmp(list(n).name(1:4),list(n+1).name(1:4))) %(p(n)+1)==p(n+1)) &&
            true_change = 0;
        else
            true_change = 1;
        end

        time_info.raw_secs(n)     = time;
        time_info.minutes(n)      = minutes;
        time_info.seconds(n)      = seconds+subseconds/100;
        time_info.true_change(n) = true_change;
    end
end
if wrapper_OFF == 1
    delete(wbar)
end

data(data_length+1:end) = [];
```

A.5.2 Data Splicing

```
function [data data_length] = splice_data( data_left, data_right, extend, method, data_length)
if strcmp(method,'TBNs')
    %   Combines 2 dataset by taking the <extend> last and first data from the
    %   series respectively. The datasets are then pushed towards the average
    %   of the last and first point of the set respectively. This pushing is
    %   done logarithmically. That is this pushing decays logarithmically away
    %   from the middle, with zero push at +-extend.
    left_rms  = sqrt(mean(data_left(end-(10-1):end).^2));
    right_rms = sqrt(mean(data_right(1:10 ).^2));

    threshold = 1e-4;

    if (left_rms>threshold && right_rms>threshold) || (left_rms<threshold && right_rms<threshold)
        left_factor = 2;
        right_factor = 2;
    elseif left_rms<threshold
        left_factor = inf;
        right_factor = 1;
    elseif right_rms<threshold
        left_factor = 1;
        right_factor = inf;
    end

    left = (((logspace(0,1,extend)-1)/9)*(data_right(1)-data_left(end))/left_factor)';
    right = (fliplr((logspace(0,1,extend)-1)/9)*(data_left(end)-data_right(1))/right_factor)';

    data_left(end-(extend-1):end) = left+data_left(end-(extend-1):end);
    data_right(1:extend)          = right+data_right(1:extend);


    data = [data_left; data_right];
elseif strcmp(method,'LHKs')
    changes_left = [diff(sign(data_left(data_length-999:data_length))); 0];
    index_before_down_left = 1000-max(find(changes_left==-2));
    index_before_up_left = 1000-max(find(changes_left==2));

    changes_right = [diff(sign(data_right(1:1000))); 0];
    index_before_down_right = min(find(changes_right==-2));
    index_before_up_right = min(find(changes_right==2));

    if index_before_up_left + index_before_up_right < index_before_down_left + index_before_down_right
        cut_off_end = index_before_up_left;

        data_length_new = (data_length+length(data_right(index_before_up_right:end)))-cut_off_end;
```

```matlab

        if length(data_left((data_length+1-cut_off_end):data_length_new))~=length(data_right(index_before_up_right:end))
            save errordump
        end

        data_left((data_length+1-cut_off_end):data_length_new) = data_right(index_before_up_right:end);
        data_length = data_length_new;
        data = data_left;
    else
        cut_off_end = index_before_down_left;

        data_length_new = (data_length+length(data_right(index_before_down_right:end)))-cut_off_end;

        if length(data_left((data_length+1-cut_off_end):data_length_new))~=length(data_right(index_before_down_right:end))
            save errordump
        end

        data_left((data_length+1-cut_off_end):data_length_new) = data_right(index_before_down_right:end);
        data_length = data_length_new;
        data = data_left;
    end
elseif strcmp(method,'None')
    data_length_new = (data_length+length(data_right));
    data_left((data_length+1):data_length_new) = data_right;
    data_length = data_length_new;
    data = data_left;
else
    error(['Unknown .wav combine method "' method '".'])
end
end
```

A.5.3 Main: Feature Extraction section

```
% Compute mel frequency cepstrum
% mfcc_opts.nMelbanks = 30;
mfcc_opts.nMelbanks = 20;
% mfcc_opts.nMelbanks = 13;
mfcc_opts.nc = 12; %mfcc_opts.nMelbanks-1 is max, -1
    since whether the first coef (coef 0) is included is
    decided in mfcc_opts.w
mfcc_opts.samplerate = fs;
% mfcc_opts.w = 'Mta' %EXCLUDE coef 0 since it encodes
    the energy of the window and i do not want to
    discriminate passed on the loudness of the segment
mfcc_opts.w = 'MtadD'; %include both delta and deltadelta
     coefficients%EXCLUDE coef 0 since it encodes the
    energy of the window and i do not want to discriminate
     passed on the loudness of the segment
% mfcc_opts.highorlowfilters = 'low'
mfcc_opts.highorlowfilters = 'high'

[ceps,mfcc_opts]=isp_mfccvb(data,mfcc_opts);
ceps = ceps';

%Normalize ceps
% Subtract the mean from the data
ceps = bsxfun(@minus, ceps,   mean(ceps));
% devide by standard deviation
ceps = bsxfun(@times, ceps, 1./std(ceps));
```

A.5.4 Modified version of ISP's MFCC Extraction

```
function [o,options]=isp_mfccvb(s,options)
%ISP_MFCCVB  MFCC implementation based on Mike Brookes'
    Voicebox
%
% SYNTAX
%   [mfccs,options]=isp_mfccvb(s,options ...)
%
% DESCRIPTION
%   Wrapper function for MELCEPST from Mike Brookes'
    Voicebox toolbox.
%
% INPUT
%   s:
%     Sound signal.
%  options ...:
%    Field/value pairs or structs with the following
    fields:
%    mfccprsec:
%      Number of MFCCs per second (default 100).
%    samplerate:
%      Minimum 22050 Hz (default 44100 Hz).
%    nc:
%      Number of mfccs besides the first (default 6).
%    frame:
%      Frame size ( default 20 ms = floor((FS/mfccprsec)
    *2) ).
%    hopsize:
%      Hop size (default 10 ms = floor(FS/mfccprsec)).
%    nMelbanks:
%      Number of mel banks (default 30).
%    lowercut:
%      Lowest frequency in fraction of sampling rate(
    default 0).
%    highcut:
%      Higest frequency in fraction of sampling rate (
    default 11025/FS > cutoff 55125 hz ).
%    w:
%      Any sensible combination of the following: (
    default 0Mta)
%       'R'  rectangular window in time domain
%       'N'      Hanning window in time domain
%       'M'      Hamming window in time domain (default)
%
%       't'  triangular shaped filters in mel domain (
    default)
%       'n'  hanning shaped filters in mel domain
%       'm'  hamming shaped filters in mel domain
%
%       'p'      filters act in the power domain
%       'a'      filters act in the absolute magnitude
    domain (default)
%
%       '0'  include 0'th order cepstral coefficient
%       'e'  include log energy
%       'd'      include delta coefficients (dc/dt)
%       'D'      include delta to delta coefficients (d^2c
    /dt^2)
%
```

```
%       'z'  highest and lowest filters taper down to
    zero (default)
%       'y'  lowest filter remains at 1 down to 0
    frequency and highest filter remains at 1 up to
    nyquist freqency
%       If 'ty' or 'ny' is specified, the total power in
    the fft is preserved.
%
% OUTPUT
%   mfccs:
%     Matrix where mfccs(:,n) is the n'th MFCC vector.
%   options:
%     The sum of input options and default options.
%
% EXAMPLE
%     [mfccs]=isp_mfcc(s)
%     [mfccs,options]=isp_mfcc(s,options)
%
% HISTORY
%   copyrights Intelligent sound 2006 and Mike Brookes
%   licence GPL?
%   author Tue Lehn Schioeler, ISP,IMM,DTU
%   date 08022006
%   version 1.0

% This program is free software; you can redistribute it
    and/or modify it
% under the terms of the GNU General Public License
    version 2 as published
% by the Free Software Foundation.


if nargin < 2
    par='';
end
if nargin == 2
    par=options;
end


par=setpars(par);

if par.samplerate<22050;
```

```
%      warning('Sampling rate below 22050, output cannot
    be compared to other mfccs')
     par.highcut=0.5;
end

[o]=isp_melcepst(s,par.samplerate,par.w,par.nc,par.
    nMelbanks,par.frame,par.hopsize,par.lowercut,par.
    highcut,par.highorlowfilters)';

options=par;


function par=setpars(par)
if (~isfield(par,'mfccprsec'))
     par.mfccprsec=100;
end
if (~isfield(par,'samplerate'))
     par.samplerate=44100;
end
FS=par.samplerate;
if (~isfield(par,'nc'))
     par.nc=6;
end
if (~isfield(par,'frame'))
     par.frame=floor((FS/par.mfccprsec)*2);
end
if (~isfield(par,'hopsize'))
     par.hopsize=floor(FS/par.mfccprsec);
end
if (~isfield(par,'nMelbanks'))
     par.nMelbanks=30;
end
if (~isfield(par,'lowercut'))
     par.lowercut=0;
end
if (~isfield(par,'highcut'))
     par.highcut=11025/FS;
end
if (~isfield(par,'w'))
     par.w='0Mta';
end
if (~isfield(par,'highorlowfilters'))
     par.highorlowfilters='0Mta';
end
```

```
par.timestamp=datestr(now);


function f=isp_enframe(x,win,inc)
%ENFRAME split signal up into (overlapping) frames: one
    per row. F=(X,WIN,INC)
%
%       F = ENFRAME(X,LEN) splits the vector X up into
%       frames. Each frame is of length LEN and occupies
%       one row of the output matrix. The last few frames
     of X
%       will be ignored if its length is not divisible by
     LEN.
%       It is an error if X is shorter than LEN.
%
%       F = ENFRAME(X,LEN,INC) has frames beginning at
    increments of INC
%       The centre of frame I is X((I-1)*INC+(LEN+1)/2)
    for I=1,2,...
%       The number of frames is fix((length(X)-LEN+INC)/
    INC)
%
%       F = ENFRAME(X,WINDOW) or ENFRAME(X,WINDOW,INC)
    multiplies
%       each frame by WINDOW(:)

%       Copyright (C) Mike Brookes 1997
%
%      Last modified Tue May 12 13:42:01 1998
%
%   VOICEBOX home page: http://www.ee.ic.ac.uk/hp/staff/
    dmb/voicebox/voicebox.html
%
%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%   This program is free software; you can redistribute
    it and/or modify
%   it under the terms of the GNU General Public License
    as published by
%   the Free Software Foundation; either version 2 of the
     License, or
%   (at your option) any later version.
%
```

```
%   This program is distributed in the hope that it will be useful,
%   but WITHOUT ANY WARRANTY; without even the implied warranty of
%   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
%   GNU General Public License for more details.
%
%   You can obtain a copy of the GNU General Public License from
%   ftp://prep.ai.mit.edu/pub/gnu/COPYING2.0 or by writing to
%   Free Software Foundation, Inc.,675 Mass Ave, Cambridge, MA 02139, USA.
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

nx=length(x);
nwin=length(win);
if (nwin == 1)
   len = win;
else
   len = nwin;
end
if (nargin < 3)
   inc = len;
end
nf = fix((nx-len+inc)/inc);
f=zeros(nf,len);
indf= inc*(0:(nf-1)).';
inds = (1:len);
f(:) = x(indf(:,ones(1,len))+inds(ones(nf,1),:));
if (nwin > 1)
    w = win(:)';
    f = f .* w(ones(nf,1),:);
end


function [x,mn,mx]=isp_melbankm(p,n,fs,fl,fh,w)
%MELBANKM determine matrix for a mel spaced filterbank [X,MN,MX]=(P,N,FS,FL,FH,W)
%
% Inputs:       p   number of filters in filterbank
%               n    length of fft
%               fs   sample rate in Hz
%               fl   low end of the lowest filter as a fraction of fs (default = 0)
%               fh   high end of highest filter as a fraction of fs (default = 0.5)
%               w    any sensible combination of the following:
%                          't'  triangular shaped filters in mel domain (default)
%                          'n'  hanning shaped filters in mel domain
%                          'm'  hamming shaped filters in mel domain
%
%                          'z'  highest and lowest filters taper down to zero (default)
%                          'y'  lowest filter remains at 1 down to 0 frequency and
%                               highest filter remains at 1 up to nyquist freqency
%
%                           If 'ty' or 'ny' is specified, the total power in the fft is preserved.
%
% Outputs:      x      a sparse matrix containing the filterbank amplitudes
%                          If x is the only output argument then size(x)=[p,1+floor(n/2)]
%                          otherwise size(x)=[p,mx-mn+1]
%               mn   the lowest fft bin with a nonzero coefficient
%               mx   the highest fft bin with a nonzero coefficient
%
% Usage:        f=fft(s);                        f=fft(s);
%               x=melbankm(p,n,fs);              [x,na,nb]=melbankm(p,n,fs);
%               n2=1+floor(n/2);                 z=log(x*(f(na:nb)).*conj(f(na:nb)));
%               z=log(x*abs(f(1:n2)).^2);
%               c=dct(z); c(1)=[];
%
```

```
% To plot filterbanks e.g.     plot(melbankm
  (20,256,8000)')
%


%      Copyright (C) Mike Brookes 1997
%
%      Last modified Tue May 12 16:15:28 1998
%
%   VOICEBOX home page: http://www.ee.ic.ac.uk/hp/staff/
    dmb/voicebox/voicebox.html
%
%
  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   This program is free software; you can redistribute
    it and/or modify
%   it under the terms of the GNU General Public License
    as published by
%   the Free Software Foundation; either version 2 of the
     License, or
%   (at your option) any later version.
%
%   This program is distributed in the hope that it will
    be useful,
%   but WITHOUT ANY WARRANTY; without even the implied
    warranty of
%   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
     See the
%   GNU General Public License for more details.
%
%   You can obtain a copy of the GNU General Public
    License from
%   ftp://prep.ai.mit.edu/pub/gnu/COPYING2.0 or by
    writing to
%   Free Software Foundation, Inc.,675 Mass Ave,
    Cambridge, MA 02139, USA.
%
  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% p=30;
% f1=0;
% fh=11025/fs;
% if fs<22100
%     warning('Samling rate below 22100, results may be
    wrong');
%     fh=0.5;
% end

if nargin < 6
  w='tz';
  if nargin < 5
    fh=0.5;
    if nargin < 4
      fl=0;
    end
  end
end
f0=700/fs;
fn2=floor(n/2);
lr=log((f0+fh)/(f0+fl))/(p+1);
% convert to fft bin numbers with 0 for DC term
bl=n*((f0+fl)*exp([0 1 p p+1]*lr)-f0);
b2=ceil(bl(2));
b3=floor(bl(3));
if any(w=='y')
  pf=log((f0+(b2:b3)/n)/(f0+fl))/lr;
  fp=floor(pf);
  r=[ones(1,b2) fp fp+1 p*ones(1,fn2-b3)];
  c=[1:b3+1 b2+1:fn2+1];
  v=2*[0.5 ones(1,b2-1) 1-pf+fp pf-fp ones(1,fn2-b3-1)
      0.5];
  mn=1;
  mx=fn2+1;
else
  b1=floor(bl(1))+1;
  b4=min(fn2,ceil(bl(4)))-1;
  pf=log((f0+(b1:b4)/n)/(f0+fl))/lr;
  fp=floor(pf);
  pm=pf-fp;
  k2=b2-b1+1;
  k3=b3-b1+1;
  k4=b4-b1+1;
  r=[fp(k2:k4) 1+fp(1:k3)];
  c=[k2:k4 1:k3];
  v=2*[1-pm(k2:k4) pm(1:k3)];
  mn=b1+1;
  mx=b4+1;
```

```
end
if any(w=='n')
  v=1-cos(v*pi/2);
elseif any(w=='m')
  v=1-0.92/1.08*cos(v*pi/2);
end
if nargout > 1
  x=sparse(r,c,v);
else
  x=sparse(r,c+mn-1,v,p,1+fn2);
end

function c=isp_melcepst(s,fs,w,nc,p,n,inc,fl,fh,
    highorlowfilters)
%MELCEPST Calculate the mel cepstrum of a signal C=(S,FS,
   W,NC,P,N,INC,FL,FH)
%
%
% Simple use: c=melcepst(s,fs)  % calculate mel cepstrum
    with 12 coefs, 256 sample frames
%                                c=melcepst(s,fs,'e0dD')
     % include log energy, 0th cepstral coef, delta and
    delta to delta coefs
%
% Inputs:
%     s   speech signal
%     fs   sample rate in Hz (default 11025)
%     nc   number of cepstral coefficients excluding 0'th
    coefficient (default 12)
%     n    length of frame (default power of 2 <30 ms))
%     p    number of filters in filterbank (default floor
    (3*log(fs)) )
%     inc frame increment (default n/2)
%     fl   low end of the lowest filter as a fraction of
    fs (default = 0)
%     fh   high end of highest filter as a fraction of fs
    (default = 0.5)
%
%               w   any sensible combination of the
    following:
%
%                                'R'   rectangular window
    in time domain
```

```
%                                'N'      Hanning window in
    time domain
%                                'M'      Hamming window in
    time domain (default)
%
%                      't'  triangular shaped filters in
   mel domain (default)
%                      'n'  hanning shaped filters in mel
   domain
%                      'm'  hamming shaped filters in mel
   domain
%
%                                'p'      filters act in
   the power domain
%                                'a'      filters act in
   the absolute magnitude domain (default)
%
%                           '0'  include 0'th order
   cepstral coefficient
%                                'e'  include log energy
%                                'd'      include delta
   coefficients (dc/dt)
%                                'D'      include delta to
   delta coefficients (d^2c/dt^2)
%
%                      'z'  highest and lowest filters
   taper down to zero (default)
%                      'y'  lowest filter remains at 1
   down to 0 frequency and
%                                   highest filter remains
   at 1 up to nyquist freqency
%
%                       If 'ty' or 'ny' is specified, the
   total power in the fft is preserved.
%
% Outputs:      c      mel cepstrum output: one frame per
   row. Log energy, if requested, is the
%                  first element of each row followed by
   the delta and then the delta to delta
%                  coefficients.
%


%      Copyright (C) Mike Brookes 1997
```

```
%
%      Last modified Mon May 20 10:35:32 2002
%
%   VOICEBOX is a MATLAB toolbox for speech processing. Home page is at
%   http://www.ee.ic.ac.uk/hp/staff/dmb/voicebox/voicebox.html
%
%   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   This program is free software; you can redistribute it and/or modify
%   it under the terms of the GNU General Public License as published by
%   the Free Software Foundation; either version 2 of the License, or
%   (at your option) any later version.
%
%   This program is distributed in the hope that it will be useful,
%   but WITHOUT ANY WARRANTY; without even the implied warranty of
%   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
%   GNU General Public License for more details.
%
%   You can obtain a copy of the GNU General Public License from
%   ftp://prep.ai.mit.edu/pub/gnu/COPYING2.0 or by writing to
%   Free Software Foundation, Inc.,675 Mass Ave, Cambridge, MA 02139, USA.
%   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

if nargin<2 fs=11025; end
if nargin<3 w='M'; end
if nargin<4 nc=12; end
if nargin<5 p=floor(3*log(fs)); end
if nargin<6 n=pow2(floor(log2(0.03*fs))); end
if nargin<10
    highorlowfilters = 'low';
    if nargin<9
        fh=0.5;
        if nargin<8
            fl=0;
            if nargin<7
                inc=floor(n/2);
            end
        end
    end
end

if ~strcmp(highorlowfilters,'low') && ~strcmp(
    highorlowfilters,'high')
    highorlowfilters = 'low';
    warning(['high or low filters unspecified, using '
        highorlowfilters ' filters.']);
end

if length(w)==0
   w='M';
end
if any(w=='R')
    z=isp_enframe(s,n,inc);
elseif any (w=='N')
    z=isp_enframe(s,hanning(n),inc);
else
    z=isp_enframe(s,hamming(n),inc);
end
f=isp_rfft(z.');
[m,a,b]=isp_melbankm(p,n,fs,fl,fh,w);
pw=f(a:b,:).*conj(f(a:b,:));
pth=1e-8;
if any(w=='p')
    y=log(max(m*pw,pth));
else
    ath=sqrt(pth);
    y=log(max(m*abs(f(a:b,:)),ath));
end
c=isp_rdct(y).';
nf=size(c,1);
nc=nc+1;
if p>nc
    if strcmp(highorlowfilters,'high')
        c(:,2:(end-nc+1))=[];
    elseif strcmp(highorlowfilters,'low')
        c(:,nc+1:end)=[];
    end
elseif p<nc
    c=[c zeros(nf,nc-p)];
end
if ~any(w=='0')
    c(:,1)=[];
    nc=nc-1;
end
if any(w=='e')
    c=[log(sum(pw)).' c];
    nc=nc+1;
end

% calculate derivative

if any(w=='D')
  vf=(4:-1:-4)/60;
  af=(1:-1:-1)/2;
  ww=ones(5,1);
  cx=[c(ww,:); c; c(nf*ww,:)];
  vx=reshape(filter(vf,1,cx(:)),nf+10,nc);
  vx(1:8,:)=[];
  ax=reshape(filter(af,1,vx(:)),nf+2,nc);
  ax(1:2,:)=[];
  vx([1 nf+2],:)=[];
  if any(w=='d')
     c=[c vx ax];
  else
     c=[c ax];
  end
elseif any(w=='d')
  vf=(4:-1:-4)/60;
  ww=ones(4,1);
  cx=[c(ww,:); c; c(nf*ww,:)];
  vx=reshape(filter(vf,1,cx(:)),nf+8,nc);
  vx(1:8,:)=[];
  c=[c vx];
end

if nargout<1
    [nf,nc]=size(c);
    t=((0:nf-1)*inc+(n-1)/2)/fs;
    ci=(1:nc)-any(w=='0')-any(w=='e');
    imh = imagesc(t,ci,c.');
    axis('xy');
    xlabel('Time (s)');
    ylabel('Mel-cepstrum coefficient');
        map = (0:63)'/63;
        colormap([map map map]);
        colorbar;
end


function y=isp_rdct(x,n)
%RDCT     Discrete cosine transform of real data Y=(X,N)
% Data is truncated/padded to length N.
%
% This routine is equivalent to multiplying by the matrix
%
%   rdct(eye(n)) = diag([sqrt(2) 2*ones(1,n-1)]) * cos
    ((0:n-1)'*(0.5:n)*pi/n)
%
% The rows and columns of the matrix are orthogonal but
    not unit modulus.
% Various versions of the DCT are obtained by
    premultiplying the above
% matrix by diag([b/a ones(1,n-1)/a]) and postmultiplying
     the
% inverse transform matrix by its inverse. A common
    choice is a=n and/or b=sqrt(2).
% Choose a=sqrt(2n) and b=1 to make the matrix orthogonal
    .
% If b~=1 then the columns are no longer orthogonal.
%
% see IRDCT for the inverse transform




%      Copyright (C) Mike Brookes 1998
%
%      Last modified Tue Apr 13 15:56:48 1999
%
%   VOICEBOX is a MATLAB toolbox for speech processing.
    Home page is at
```

```
%   http://www.ee.ic.ac.uk/hp/staff/dmb/voicebox/voicebox
    .html
%
%
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   This program is free software; you can redistribute
    it and/or modify
%   it under the terms of the GNU General Public License
    as published by
%   the Free Software Foundation; either version 2 of the
     License, or
%   (at your option) any later version.
%
%   This program is distributed in the hope that it will
    be useful,
%   but WITHOUT ANY WARRANTY; without even the implied
    warranty of
%   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
     See the
%   GNU General Public License for more details.
%
%   You can obtain a copy of the GNU General Public
    License from
%   ftp://prep.ai.mit.edu/pub/gnu/COPYING2.0 or by
    writing to
%   Free Software Foundation, Inc.,675 Mass Ave,
    Cambridge, MA 02139, USA.
%
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

fl=size(x,1)==1;
if fl x=x(:); end
[m,k]=size(x);
if nargin<2 n=m;
elseif n>m x=[x; zeros(n-m,k)];
elseif n<m x(n+1:m,:)=[];
end

x=[x(1:2:n,:); x(2*fix(n/2):-2:2,:)];
z=[sqrt(2) 2*exp((-0.5i*pi/n)*(1:n-1))].';
y=real(fft(x).*z(:,ones(1,k)));

```

```
if fl y=y.'; end

function y=isp_rfft(x,n,d)
%RFFT     FFT of real data Y=(X,N)
% Data is truncated/padded to length N if specified.
%   N even:     (N+2)/2 points are returned with
%                       the first and last being real
%   N odd:      (N+1)/2 points are returned with the
%                       first being real



%      Copyright (C) Mike Brookes 1998
%
%      Last modified Fri Mar  7 15:43:06 2003
%
%   VOICEBOX is a MATLAB toolbox for speech processing. Home page is at
%   http://www.ee.ic.ac.uk/hp/staff/dmb/voicebox/voicebox.html
%
%   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   This program is free software; you can redistribute it and/or modify
%   it under the terms of the GNU General Public License as published by
%   the Free Software Foundation; either version 2 of the License, or
%   (at your option) any later version.
%
%   This program is distributed in the hope that it will be useful,
%   but WITHOUT ANY WARRANTY; without even the implied warranty of
%   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
%   GNU General Public License for more details.
%
%   You can obtain a copy of the GNU General Public License from
%   ftp://prep.ai.mit.edu/pub/gnu/COPYING2.0 or by writing to
%   Free Software Foundation, Inc.,675 Mass Ave,
    Cambridge, MA 02139, USA.
%
  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

s=size(x);
if prod(s)==1
    y=x
else
    if nargin <3
        d=find(s>1);
        d=d(1);
        if nargin<2
            n=[];
        end
    end
    y=fft(x,n,d);
    y=reshape(y,prod(s(1:d-1)),s(d),prod(s(d+1:end)));
    s(d)=1+fix(s(d)/2);
    y(:,s(d)+1:end,:)=[];
    y=reshape(y,s);
end
```

A.5.5 Main: MFCC segmentation section

```
seg_length = l_aw*mfcc_opts.mfccprsec; % l_aw (length
    analysis window)
seg_shift_length = l_s*mfcc_opts.mfccprsec; %l_s (length
    shift) in  Lasse's

seg_overlap = seg_length/seg_shift_length;
if(floor(seg_overlap)~=seg_overlap)
    error('Overlap must be a whole number! Adjust l_aw
        and l_s.')
end

seg_start_samples = 1:seg_shift_length:(size(ceps,1)-(
    seg_length-1));

%% Calculate the length of Tmax and Ti in segments
Tmax_influence = Tmax*mfcc_opts.mfccprsec/
    seg_shift_length;
```

```
Ti_influence    = Ti   *mfcc_opts.mfccprsec/
    seg_shift_length;
```

A.5.6 MFCC segmentation

```
function ceps_seg = ceps_seg_func(n_seg,seg_start_samples
    ,ceps,seg_length)

seg = seg_start_samples(n_seg):(seg_start_samples(n_seg)
    +(seg_length-1));

ceps_seg = ceps(seg,:);

end
```

A.5.7 Main: Metric Peak Detection section

```
if sum(strcmp(method_list,'k-means'))~=0
     [ CP.VQD   , th_cd.VQD    ] = mark_change_points(
         VQDs_optim, alpha_cd.VQD   , Ti_influence,
         Tmax_influence, seg_start_samples, seg_overlap );
end

if sum(strcmp(method_list,'KL'))~=0
     [ CP.KL    , th_cd.KL     ] = mark_change_points( KLs
              , alpha_cd.KL    , Ti_influence,
         Tmax_influence, seg_start_samples, seg_overlap );
end

if sum(strcmp(method_list,'DSD'))~=0
     [ CP.DSD   , th_cd.DSD    ] = mark_change_points( DSDs
              , alpha_cd.DSD   , Ti_influence,
         Tmax_influence, seg_start_samples, seg_overlap );
end

if sum(strcmp(method_list,'RuLSIF'))~=0
     [ CP.RuLSIF, th_cd.RuLSIF] = mark_change_points(
         RuLSIFs    , alpha_cd.RuLSIF, Ti_influence,
         Tmax_influence, seg_start_samples, seg_overlap );
end
```

A.5.8 Metric Peak Detection

```
function [ CP, th_cd ] = mark_change_points( metric_data , alpha , Ti_influence , Tmax_influence , seg_start_samples , seg_overlap )
%% Calculate VQD_n,avr & th_cd,n (Lasse's report page 62)

movingavg = moving(metric_data,2*Tmax_influence+1); % calculate moving average with begin and end phase included

th_cd  = alpha*movingavg;

%% Find change-points

CP = zeros(length(seg_start_samples)-seg_overlap,1);
for n_seg = 1:length(metric_data)
    if(metric_data(n_seg)>th_cd(n_seg))
        %take start and end steps into account
        if(n_seg - Ti_influence < 1)
            Ti_influenced_segments = 1:n_seg+Ti_influence;
        elseif (n_seg + Ti_influence > length(metric_data))
            Ti_influenced_segments = n_seg-Ti_influence:length(metric_data);
        else
            Ti_influenced_segments = n_seg-Ti_influence:n_seg+Ti_influence;
        end
        %find largest local peak
        if(metric_data(n_seg)==max(metric_data(Ti_influenced_segments)))
            CP(n_seg) = 1;
        else
            CP(n_seg) = 0;
        end
    end
end

end
```

A.6 Miscellaneous

```
function value = compare_segments(segment_A,segment_B,
    method)

if strcmp('k-means', method)

    segment_A = segment_A(:,1:12);
    segment_B = segment_B(:,1:12);
    global K
    K_FAC = K

    kmeans_opts = statset('MaxIter',10000);

    warning('off','all');
    [~,previous_centroids] = kmeans(segment_A,K_FAC,'
        Options',kmeans_opts,'emptyaction','singleton','
        onlinephase','off');
    [~,next_centroids     ] = kmeans(segment_B,K_FAC,'
        Options',kmeans_opts,'emptyaction','singleton','
        onlinephase','off');
    warning('on','all');
    %value(n) = (VQD_optim(previous_centroids,segment_B)
        + VQD_optim(next_centroids,segment_A))/2;
    value = (VQD_optim(previous_centroids,next_centroids)
         + VQD_optim(next_centroids,previous_centroids))
        /2;
elseif strcmp('RuLSIF', method)

    alpha = .1; % Smoothing parameter
    sigma_list = 0.2:0.4:2; % Kernel Width
    lambda_list = -6:1:-2; % Reguralization parameter
    k_folds = 5;

    addpath('RuLSIF_change_detection')
    addpath('RuLSIF_change_detection\RuLSIF')

    [score1, ~, ~, ~, ~] = RelULSIF(segment_A',segment_B
        ',[],[],alpha,k_folds,sigma_list,lambda_list);
    [score2, ~, ~, ~, ~] = RelULSIF(segment_B',segment_A
        ',[],[],alpha,k_folds,sigma_list,lambda_list);

    value = score1 + score2;
elseif strcmp('DSD', method)
    previous_sigma = cov(segment_A)';
    next_sigma = cov(segment_B)';

    value = DSD(previous_sigma,next_sigma);
elseif strcmp('KL', method)
    previous_sigma = cov(segment_A)';
    previous_mu = mean(segment_A)';
    next_sigma = cov(segment_B)';
    next_mu = mean(segment_B)';

    value = KL(previous_sigma,next_sigma,previous_mu,
        next_mu);
elseif strcmp('BIC', method)

    previous_segment = segment_A;
    next_segment = segment_B;

    sigma_left  = cov(previous_segment);
    sigma_right = cov(next_segment);
    sigma_full  = cov([previous_segment; next_segment]);
    N1 = size(previous_segment,1);
    N2 = size(next_segment,1);
    N = N1+N2;
    R = N*log(det(sigma_full))-N1*log(det(sigma_left))-N2
        *log(det(sigma_right));
    d = size(previous_segment,2);
    P = 1/2*(d+1/2*d*(d+1))*log(N);
    lambda = 1; %BIC definition
    BIC = R-lambda*P;

    value = BIC;
elseif strcmp('ICR', method)

    sigma_A   = cov(segment_A);
    sigma_B = cov(segment_B);
    sigma_full  = cov([segment_A; segment_B]);
    N1 = size(segment_A,1);
    N2 = size(segment_B,1);
    N = N1+N2;

    ICR =      1/2*log(det(sigma_full)) -...
        1/N*1/2*(N1*log(det(sigma_A))+N2*log(det(sigma_B)
            ));

    d = size(segment_A,2);
    P = 1/2*(d+1/2*d*(d+1))*log(N);
    lambda = 1; % from paper (A Robust Stopping Criterion
        for Agglomerative Hierarchical Clustering in a
       Speaker Diarization System)
    value = ICR-lambda*P;
else
    error(['Unknown segment comparison method: "' method
        '".'])
end

end
```

A.6.1 F-measure

```
alpha_f_measure = 0.5;
methods = fieldnames(CP);
CP_TP = zeros(numel(methods),1);

for n_method = 1:numel(methods)
    CP_temp = x_axis(CP.(methods{n_method})==1);
    for n_CP = 1:numel(changepoints_true)
        if sum((CP_temp-Ti/2) < changepoints_true(n_CP) &
            changepoints_true(n_CP) < (CP_temp+Ti/2))~=0;
            CP_TP(n_method) = CP_TP(n_method) + 1;
        end
    end
    RCL.(methods{n_method}) = CP_TP(n_method)/numel(
        changepoints_true);
    PRC.(methods{n_method}) = CP_TP(n_method)/numel(
        CP_temp );
    F_measure.(methods{n_method}) = (RCL.(methods{
        n_method})*PRC.(methods{n_method}))/(
        alpha_f_measure*RCL.(methods{n_method})+(1-
        alpha_f_measure)*PRC.(methods{n_method}));
end
```

A.6.2 SCD Main Wrapper

```
function out = Main_wrapper(method,alpha_cd_input,
    alpha_fac_input,change_points,fac_method)

```

```
method_list = {method};

alpha_cd.(structname) = alpha_cd_input;
alpha_fac.(structname) = alpha_fac_input;

wrapper_ON = 1;
Main
clear wrapper_ON

out = F_measure.(structname);

end
```

A.6.3 Principal Component Analysis

```
% Subtract the mean from the data
% Y = bsxfun(@minus, ceps', mean(ceps')); %alrdy done in
    Feature extraction

% Obtain the PCA solution by doing a SVD of Y
[U, S, V] = svd(PCA_input, 'econ');

% Compute the projection onto the principal components
Z = U*S;

% Compute amount of variance explained by components
rho = diag(S).^2./sum(diag(S).^2);
```

A.6.4 Receiver Operator Characteristics

```
function [rx,ry,area] = rocarea(Ztest,Ytest,frac)
% function [rx,ry,area] = rocarea(Ztest,Ytest,frac)
%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% this function finds the ROC and the area under the
% ROC for labels in YTEST
% STEP 1 convert Y to 0-1 array
%
% INPUT
%   Ztest: values to be ROC'ed
%   Ytest: "True" labels obtained by thresholding Ytest
    at zero
%   frac: the frac of false positives for which the area
    is computed
% OUTPUT
%   rx:  False positive rates
%   ry:  True positive rates
%   area: the area under the curve for rx < frac
%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%  Lars Kai Hansen, Technical University of Denmark, (c)
    2004
%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


top=max(Ytest);
N=length(Ytest);
y=zeros(N,1);
indy=find(Ytest>0);
y(indy)=1;
Npos=sum(y);
Nneg=N-Npos;
%IZtest index for  order Ztest (small first)
[dummy,IZtest]=sort(-Ztest);
y=y(IZtest);
% COMPUTE SENSITIVITY
ry=cumsum(y);
% compute 1-specificity
rx=(1:N)-ry';
%normalize
ry=ry/Npos;
rx=rx./Nneg;
%whos
% compute area for rx<frac
index_max=max(find(rx<frac));
area=sum(ry(1:(index_max-1))'.*(rx(2:index_max)-rx(1:(
    index_max-1))))/frac;
```

A.6.5 Welch's t-test

```
function [SE DF tscore t95 f] = ttest_mine(A, B)

n1 = length(A);
x1 = mean(A);
s1 = std(A);

n2 = length(B);
x2 = mean(B);
s2 = std(B);

SE = sqrt(s1^2/n1 + s2^2/n2);

DF = (n1 - 1) + (n2 - 1);

tscore = abs(((x1 - x2)-0)/SE);

ci = 0.95;
alpha = 1 - ci;
t95 = tinv(1-alpha/2, DF);

f = tcdf(tscore,DF)-tcdf(-tscore,DF);
```

Bibliography

[1] Robert Alexander Adams and Christopher Essex. *Calculus: a complete course*, volume 4. Addison-Wesley Don Mills, Ontario, 1995.

[2] Stanley C Ahalt, Ashok K Krishnamurthy, Prakoon Chen, and Douglas E Melton. Competitive learning algorithms for vector quantization. *Neural networks*, 3(3):277–290, 1990.

[3] Peter Ahrendt, Anders Meng, and Jan Larsen. Decision time horizon for music genre classification using short time features. 2004.

[4] Jitendra Ajmera, Iain McCowan, and Hervé Bourlard. Robust speaker change detection. *Signal Processing Letters, IEEE*, 11(8):649–651, 2004.

[5] SM Ali and Samuel D Silvey. A general class of coefficients of divergence of one distribution from another. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 131–142, 1966.

[6] Daniel Aloise, Amit Deshpande, Pierre Hansen, and Preyas Popat. Np-hardness of euclidean sum-of-squares clustering. *Machine Learning*, 75(2):245–248, 2009.

[7] Jeronimo Arenas-Garcia, Vanessa Gomez-Verdejo, and Aníbal R Figueiras-Vidal. New algorithms for improved adaptive convex combination of lms transversal filters. *Instrumentation and Measurement, IEEE Transactions on*, 54(6):2239–2249, 2005.

[8] Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*, volume 1. springer New York, 2006.

[9] Christoph Boehm and Franz Pernkopf. Effective metric-based speaker segmentation in the frequency domain. In *Acoustics, Speech and Signal Processing, 2009. ICASSP 2009. IEEE International Conference on*, pages 4081–4084. IEEE, 2009.

[10] Steven Boll. Suppression of acoustic noise in speech using spectral subtraction. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 27(2):113–120, 1979.

[11] Leon Bottou and Yoshua Bengio. Convergence properties of the k-means algorithms. In *Advances in Neural Information Processing Systems 7*. Citeseer, 1995.

[12] Markus M Breunig, Hans-Peter Kriegel, Raymond T Ng, and Jörg Sander. Lof: identifying density-based local outliers. In *ACM Sigmod Record*, volume 29, pages 93–104. ACM, 2000.

[13] Mike Brookes et al. Voicebox: Speech processing toolbox for matlab. *Software, available [Mar. 2011] from www.ee.ic.ac.uk/hp/staff/dmb/voicebox/voicebox.html*, 1997.

[14] D Burton, J Shore, and J Buck. A generalization of isolated word recognition using vector quantization. In *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP'83.*, volume 8, pages 1021–1024. IEEE, 1983.

[15] David Byatt. Convergent variants of the nelder-mead algorithm. 2000.

[16] Jingdong Chen, Kuldip K Paliwal, and Satoshi Nakamura. Cepstrum derived from differentiated power spectrum for robust speech recognition. *Speech Communication*, 41(2):469–484, 2003.

[17] Shih-Sian Cheng and Hsin-Min Wang. A sequential metric-based audio segmentation method via the bayesian information criterion. In *INTERSPEECH*, 2003.

[18] Shih-Sian Cheng and Hsin-min Wang. Metric-seqdac: A hybrid approach for audio segmentation. In *Proc. International Conference on Spoken Language Processing*, 2004.

[19] Stephen M Chu, Hao Tang, and Thomas S Huang. Fishervoice and semi-supervised speaker clustering. In *Acoustics, Speech and Signal Processing, 2009. ICASSP 2009. IEEE International Conference on*, pages 4089–4092. IEEE, 2009.

[20] CoSound. A cognitive systems approach to enriched and actionable information from audio streams supported by the danish strategic research council, jan. 2012 – dec. 2015. More information online @http://www.cosound.dk/. Last checked, June 2013.

[21] I. Csiszár. Information-type measures of difference of probability distributions and indirect observations. *Studia Sci. Math. Hungar.*, 2:299–318, 1967.

[22] Neil R Davis. Method for audio editing, July 12 1988. US Patent 4,757,540.

[23] Steven Davis and Paul Mermelstein. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 28(4):357–366, 1980.

[24] Perrine Delacourt and Christian J Wellekens. Distbic: A speaker-based segmentation for audio data indexing. *Speech communication*, 32(1):111–126, 2000.

[25] John R Deller, John G Proakis, and John HL Hansen. *Discrete-time processing of speech signals*. Ieee New York, NY, USA:, 2000.

[26] Qiang Du, Vance Faber, and Max Gunzburger. Centroidal voronoi tessellations: Applications and algorithms. *SIAM review*, 41(4):637–676, 1999.

[27] Brian Everitt and Anders Skrondal. *The Cambridge dictionary of statistics*, volume 4. Cambridge University Press Cambridge, 2002.

[28] JE Muñoz Expósito, N Ruiz Reyes, S Garcia Galán, and P Vera Candeas. Speech/music classification based on distributed evolutionary fuzzy logic for intelligent audio coding. In *Pattern Recognition and Image Analysis*, pages 556–563. Springer, 2007.

[29] Gunnar Fant. *Analys av de svenska konsonantljuden: talets allmänna svängningsstruktur*. LM Ericsson, 1949.

[30] Ling Feng. *Speaker recognition*. PhD thesis, Technical University of Denmark, DTU, DK-2800 Kgs. Lyngby, Denmark, 2004.

[31] Ling Feng. English Language Speech Database for Speaker Recognition, ELSDSR, @http://www2.imm.dtu.dk. last checked, June 2013.

[32] Ling Feng, Andreas Brinch Nielsen, and Lars Kai Hansen. Vocal segment classification in popular music. In *ISMIR*, pages 121–126, 2008.

[33] JA Nolazco Flores and SJ Young. Continuous speech recognition in noise using spectral subtraction and hmm adaptation. In *Acoustics, Speech, and Signal Processing, 1994. ICASSP-94., 1994 IEEE International Conference on*, volume 1, pages I–409. IEEE, 1994.

[34] Edward W Forgy. Cluster analysis of multivariate data: efficiency versus interpretability of classifications. *Biometrics*, 21:768–769, 1965.

[35] Mark JF Gales and Stephen J Young. Robust continuous speech recognition using parallel model combination. *Speech and Audio Processing, IEEE Transactions on*, 4(5):352–359, 1996.

[36] Todor Ganchev, Nikos Fakotakis, and George Kokkinakis. Comparative evaluation of various mfcc implementations on the speaker verification task. In *Proceedings of the SPECOM*, volume 1, pages 191–194, 2005.

[37] John S. Garofolo. TIMIT Acoustic-Phonetic Continuous Speech Corpus, @http://www.ldc.upenn.edu/. Last checked, April 2013.

[38] D Geller, Reinhold Haeb-Umbach, and Hermann Ney. Improvements in speech recognition for voice dialing in the car environment. In *Speech Processing in Adverse Conditions*, 1992.

[39] Allen Gersho and Robert M Gray. *Vector quantization and signal compression*. Springer, 1992.

[40] Robert Gray. Vector quantization. *ASSP Magazine, IEEE*, 1(2):4–29, 1984.

[41] Donald D. Greenwood. Mailing list from 2009; discussion on bias in the Mel scale due to Hysteresis effects, @http://lists.mcgill.ca/scripts/wa.exe?A2=ind0907d&L=auditory&P=389. Last checked, May 2013.

[42] Kyu Jeong Han and Shrikanth S Narayanan. A robust stopping criterion for agglomerative hierarchical clustering in a speaker diarization system. In *INTERSPEECH*, pages 1853–1856. Citeseer, 2007.

[43] Lars Kai Hansen, Peter Ahrendt, and Jan Larsen. Towards cognitive component analysis. 2005.

[44] Lars Kai Hansen, Christian Liisberg, and Peter Salamon. The error-reject tradeoff. *Open Systems & Information Dynamics*, 4(2):159–184, 1997.

[45] Gunther Heidemann and Helge Ritter. Efficient vector quantization using the wta-rule with activity equalization. *Neural Processing Letters*, 13(1):17–30, 2001.

[46] Ramon E Henkel. *Tests of significance*, volume 4. Sage, 1976.

[47] Hynek Hermansky and Nelson Morgan. Rasta processing of speech. *Speech and Audio Processing, IEEE Transactions on*, 2(4):578–589, 1994.

[48] Hynek Hermansky, Nelson Morgan, Aruna Bayya, and Phil Kohn. Compensation for the effect of the communication channel in auditory-like analysis of speech (rasta-plp). In *Second European Conference on Speech Communication and Technology*, 1991.

[49] HG Hirsch, Peter Meyer, and Hans-Wilhelm Ruehl. Improved speech recognition using high-pass filtering of subband envelopes. In *Eurospeech*, volume 91, pages 413–416, 1991.

[50] Thomas Hofmann and Joachim M Buhmann. Competitive learning algorithms for robust vector quantization. *Signal Processing, IEEE Transactions on*, 46(6):1665–1675, 1998.

[51] Peter J Huber. *Robust statistics*. Springer, 2011.

[52] Anil Jain and Douglas Zongker. Feature selection: Evaluation, application, and small sample performance. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 19(2):153–158, 1997.

[53] Jesper H. Jensen, Anders Meng, Elvis Ramonovic, Kaare B. Petersen, Mads G. Christensen, Sigurdur Sigurdsson, and Tue Lehn-Schiøler. Intelligent Sound Processing Toolbox, ISP. Available online @http://kom.aau.dk/project/isound/. Last checked, May 2013.

[54] Kasper W Jørgensen, Lasse Mølgaard, and Lars Kai Hansen. Unsupervised speaker change detection for broadcast news segmentation. 2006.

[55] Kasper Winther Jørgensen and Lasse Lohilahti Mølgaard. *Tools for automatic audio indexing*. PhD thesis, Technical University of Denmark, DTU, DK-2800 Kgs. Lyngby, Denmark, 2006.

[56] Biing-Hwang Juang and Mazin G Rahim. Signal bias removal by maximum likelihood estimation for robust telephone speech recognition. *Speech and Audio Processing, IEEE Transactions on*, 4(1):19, 1996.

[57] Takafumi Kanamori, Taiji Suzuki, and Masashi Sugiyama. Statistical analysis of kernel-based least-squares density-ratio estimation. *Machine Learning*, 86(3):335–367, 2012.

[58] Takafumi Kanamori, Taiji Suzuki, and Masashi Sugiyama. Computational complexity of kernel-based density-ratio estimation: a condition number analysis. *Machine Learning*, 90(3):431–460, 2013.

[59] Yoshinobu Kawahara and Masashi Sugiyama. Sequential change-point detection based on direct density-ratio estimation. *Statistical Analysis and Data Mining*, 5(2):114–127, 2012.

[60] H Kim, Daniel Ertelt, and Thomas Sikora. Hybrid speaker-based segmentation system using model-level clustering. In *Proc. 2005 IEEE Int. Conf. Acoustics, Speech, and Signal Processing*, volume 1, pages 745–748, 2005.

[61] Tomi Kinnunen, TEEMU KILPELÄINEN, and Pasi FrÄnti. Comparison of clustering algorithms in speaker identification. *dim*, 1:2, 2000.

[62] Tomi Kinnunen and Haizhou Li. An overview of text-independent speaker recognition: From features to supervectors. *Speech communication*, 52(1):12–40, 2010.

[63] Teuvo Kohonen. *Self-organizing maps*, volume 30. Springer, 2001.

[64] Margarita Kotti, Emmanouil Benetos, and Constantine Kotropoulos. Automatic speaker change detection with the bayesian information criterion using mpeg-7 features and a fusion scheme. In *Circuits and Systems, 2006. ISCAS 2006. Proceedings. 2006 IEEE International Symposium on*, pages 4–pp. IEEE, 2006.

[65] Margarita Kotti, Emmanouil Benetos, and Costas Kotropoulos. Computationally efficient and robust bic-based speaker segmentation. *Audio, Speech, and Language Processing, IEEE Transactions on*, 16(5):920–933, 2008.

[66] Margarita Kotti, Luıs Gustavo PM Martins, Emmanouil Benetos, Jaime S Cardoso, and Constantine Kotropoulos. Automatic speaker segmentation using multiple features and distance measures: a comparison of three approaches. In *Multimedia and Expo, 2006 IEEE International Conference on*, pages 1101–1104. IEEE, 2006.

[67] E Kreyszig. Advanced engineering mathematics, 9th ednwiley. *New York, NY, USA*, 2005.

[68] Daben Liu Francis Kubala. Fast speaker change detection for broadcast news transcription and indexing. 1999.

[69] Solomon Kullback and Richard A Leibler. On information and sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86, 1951.

[70] Jeffrey C Lagarias, James A Reeds, Margaret H Wright, and Paul E Wright. Convergence properties of the nelder–mead simplex method in low dimensions. *SIAM Journal on Optimization*, 9(1):112–147, 1998.

[71] Michael S Lewicki. Efficient coding of natural sounds. *Nature neuroscience*, 5(4):356–363, 2002.

[72] Robert Michael Lewis, Virginia Torczon, and Michael W Trosset. Direct search methods: then and now. *Journal of Computational and Applied Mathematics*, 124(1):191–207, 2000.

[73] Yoseph Linde, Andres Buzo, and Robert Gray. An algorithm for vector quantizer design. *Communications, IEEE Transactions on*, 28(1):84–95, 1980.

[74] Song Liu. Prof. Sugiyama's webpage, @http://www.cs.titech.ac.jp. Last checked, April 2013.

[75] Song Liu, Makoto Yamada, Nigel Collier, and Masashi Sugiyama. Change-point detection in time-series data by relative density-ratio estimation. *Neural Networks*, 2013.

[76] Stuart Lloyd. Least squares quantization in pcm. *Information Theory, IEEE Transactions on*, 28(2):129–137, 1982.

[77] Lie Lu and Hong-Jiang Zhang. Speaker change detection and tracking in real-time news broadcasting analysis. In *Proceedings of the tenth ACM international conference on Multimedia*, pages 602–610. ACM, 2002.

[78] Lie Lu and Hong-Jiang Zhang. Unsupervised speaker segmentation and tracking in real-time audio content analysis. *Multimedia systems*, 10(4):332–343, 2005.

[79] Lie Lu, Hong-Jiang Zhang, and Hao Jiang. Content analysis for audio classification and segmentation. *Speech and Audio Processing, IEEE Transactions on*, 10(7):504–516, 2002.

[80] J Makhoul and L Cosell. Lpcw: An lpc vocoder with linear predictive spectral warping. In *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP'76.*, volume 1, pages 466–469. IEEE, 1976.

[81] Amit S Malegaonkar, Aladdin M Ariyaeeinia, and Perasiriyan Sivakumaran. Efficient speaker change detection using adapted gaussian mixture models. *Audio, Speech, and Language Processing, IEEE Transactions on*, 15(6):1859–1869, 2007.

[82] MATLAB. Statistics Toolbox - Exploratory Data Analysis - Cluster Analysis - K-Means Clustering, @http://www.mathworks.se/help/stats/kmeans.html. Last checked, July 2013.

[83] Martin F McKinney and Jeroen Breebaart. Features for audio and music classification. In *ISMIR*, volume 3, pages 151–158, 2003.

[84] Sylvain Meignier, Daniel Moraru, Corinne Fredouille, Jean-Francois Bonastre, and Laurent Besacier. < i> step-by-step</i> and< i> integrated</i> approaches in broadcast news speaker diarization. *Computer Speech & Language*, 20(2):303–330, 2006.

[85] Hugo Meinedo and Joao Neto. Audio segmentation, classification and clustering in a broadcast news task. In *Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03). 2003 IEEE International Conference on*, volume 2, pages II–5. IEEE, 2003.

[86] Mohammad H Moattar and Mohammad M Homayounpour. A review on speaker diarization systems and approaches. *Speech Communication*, 54(10):1065–1103, 2012.

[87] LL Mølgaard, KW Jørgensen, and Lars Kai Hansen. Castsearch-context based spoken document retrieval. In *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on*, volume 4, pages IV–93. IEEE, 2007.

[88] Pedro J Moreno, Bhiksha Raj, and Richard M Stern. A vector taylor series approach for environment-independent speech recognition. In *Acoustics, Speech, and Signal Processing, 1996. ICASSP-96. Conference Proceedings., 1996 IEEE International Conference on*, volume 2, pages 733–736. IEEE, 1996.

[89] Omer Mohsin Mubarak, Eliathamby Ambikairajah, and Julien Epps. Analysis of an mfcc-based audio indexing system for efficient coding of multimedia sources. In *ISSPA*, pages 619–622, 2005.

[90] Hiroshi Nakagawa and Tatsunori Mori. Automatic term recognition based on statistics of compound nouns and their components. *Terminology*, 9(2):201–219, 2003.

[91] John A Nelder and Roger Mead. A simplex method for function minimization. *The computer journal*, 7(4):308–313, 1965.

[92] Kazumi Ohkura, Masahide Sugiyama, and Shigeki Sagayama. Speaker adaptation based on transfer vector field smoothing with continuous mixture density hmms. In *Second International Conference on Spoken Language Processing*, 1992.

[93] Alan V Oppenheim. Superposition in a class of nonlinear systems. Technical report, DTIC Document, 1965.

[94] Alan V Oppenheim and Ronald W Schafer. From frequency to quefrency: A history of the cepstrum. *Signal Processing Magazine, IEEE*, 21(5):95–106, 2004.

[95] Karl Pearson. X. on the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 50(302):157–175, 1900.

[96] R Plomp, LCW Pols, and JP Van der Geer. Dimensional analysis of vowel spectra. *Journal of Acoustical Society of America*, 41:707–712, 1967.

[97] Louis CW Pols. Spectral analysis and identification of dutch vowels in monosyllabic words. 1977.

[98] Dimitrie C Popescu and Ilija Zeljkovic. Kalman filtering of colored noise for speech enhancement. In *Acoustics, Speech and Signal Processing, 1998. Proceedings of the 1998 IEEE International Conference on*, volume 2, pages 997–1000. IEEE, 1998.

[99] John G Proakis. *Digital Signal Processing: Principles, Algorithms, and Applications, 4/e.* Pearson Education India, 2007.

[100] Lawrence Rabiner and Biing-Hwang Juang. Fundamentals of speech recognition. 1993.

[101] Douglas A Reynolds. Channel robust speaker verification via feature mapping. In *Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03). 2003 IEEE International Conference on*, volume 2, pages II–53. IEEE, 2003.

[102] Walter Rudin. *Principles of mathematical analysis*, volume 3. McGraw-Hill New York, 1964.

[103] Franklin E Satterthwaite. An approximate distribution of estimates of variance components. *Biometrics bulletin*, 2(6):110–114, 1946.

[104] Burr Settles. Active learning literature survey. *University of Wisconsin, Madison*, 2010.

[105] Matthew A Siegler, Uday Jain, Bhiksha Raj, and Richard M Stern. Automatic segmentation, classification and clustering of broadcast news audio. In *Proc. DARPA Broadcast News Workshop*, page 11, 1997.

[106] Sigurdur Sigurdsson, Kaare Brandt Petersen, and Tue Lehn-Schiøler. Mel frequency cepstral coefficients: An evaluation of robustness of mp3 encoded music. 2006.

[107] Rohit Sinha, Sue E Tranter, Mark JF Gales, and Philip C Woodland. The cambridge university march 2005 speaker diarisation system. In *INTERSPEECH*, pages 2437–2440, 2005.

[108] F Soong, A Rosenberg, L Rabiner, and B Juang. A vector quantization approach to speaker recognition. In *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP'85.*, volume 10, pages 387–390. IEEE, 1985.

[109] Stanley S Stevens. On the psychophysical law. *Psychological review*, 64(3):153, 1957.

[110] Stanley S Stevens and JOHN Volkmann. The relation of pitch to frequency: A revised scale. *The American Journal of Psychology*, 53(3):329–353, 1940.

[111] Masashi Sugiyama, Taiji Suzuki, and Takafumi Kanamori. Density-ratio matching under the bregman divergence: a unified framework of density-ratio estimation. *Annals of the Institute of Statistical Mathematics*, 64(5):1009–1044, 2012.

[112] Masashi Sugiyama, Taiji Suzuki, Shinichi Nakajima, Hisashi Kashima, Paul von Bünau, and Motoaki Kawanabe. Direct importance estimation for covariate shift adaptation. *Annals of the Institute of Statistical Mathematics*, 60(4):699–746, 2008.

[113] Pang-Ning Tan et al. *Introduction to data mining.* Pearson Education India, 2007.

[114] SE Tranter, K Yu, G Everinann, and PC Woodland. Generating and evaluating segmentations for automatic speech recognition of conversational telephone speech. In *Acoustics, Speech, and Signal Processing, 2004. Proceedings.(ICASSP'04). IEEE International Conference on*, volume 1, pages I–753. IEEE, 2004.

[115] Alain Tritschler and Ramesh A Gopinath. Improved speaker segmentation and segments clustering using the bayesian information criterion. In *Eurospeech*, volume 99, pages 679–682, 1999.

[116] Saeed V Vaseghi and Ben P Milner. Noise compensation methods for hidden markov model speech recognition in adverse environments. *Speech and Audio Processing, IEEE Transactions on*, 5(1):11–21, 1997.

[117] William N Venables, Brian D Ripley, and WN Venables. *Modern applied statistics with S-PLUS*, volume 250. Springer-verlag New York, 1994.

[118] Olli Viikki and Kari Laurila. Cepstral domain segmental feature vector normalization for noise robust speech recognition. *Speech Communication*, 25(1):133–147, 1998.

[119] Dong Wang, Lie Lu, and Hong-Jiang Zhang. Speech segmentation without speech recognition. In *Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03). 2003 IEEE International Conference on*, volume 1, pages I–468. IEEE, 2003.

[120] Bernard L Welch. The generalization ofstudent's' problem when several different population variances are involved. *Biometrika*, 34(1/2):28–35, 1947.

[121] PC Woodland, MJF Gales, and D Pye. Improving environmental robustness in large vocabulary speech recognition. In *Acoustics, Speech, and Signal Processing, 1996. ICASSP-96. Conference Proceedings., 1996 IEEE International Conference on*, volume 1, pages 65–68. IEEE, 1996.

[122] Chung-Hsien Wu and Chia-Hsin Hsieh. Multiple change-point audio segmentation and classification using an mdl-based gaussian model. *Audio, Speech, and Language Processing, IEEE Transactions on*, 14(2):647–657, 2006.

[123] Ziyou Xiong, Regunathan Radhakrishnan, Ajay Divakaran, and Thomas S Huang. Comparing mfcc and mpeg-7 audio features for feature extraction, maximum likelihood hmm and entropic prior hmm for sports audio classification. In *Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03). 2003 IEEE International Conference on*, volume 5, pages V–628. IEEE, 2003.

[124] Min Xu, Ling-Yu Duan, Jianfei Cai, Liang-Tien Chia, Changsheng Xu, and Qi Tian. Hmm-based audio keyword generation. In *Advances in Multimedia Information Processing-PCM 2004*, pages 566–574. Springer, 2005.

[125] Makoto Yamada, Taiji Suzuki, Takafumi Kanamori, Hirotaka Hachiya, and Masashi Sugiyama. Relative density-ratio estimation for robust distribution comparison. *arXiv preprint arXiv:1106.4729*, 2011.

[126] Arthur Zimek, Erich Schubert, and Hans-Peter Kriegel. A survey on unsupervised outlier detection in high-dimensional numerical data. *Statistical Analysis and Data Mining*, 5(5):363–387, 2012.