BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF INFORMATION TECHNOLOGY
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

VOICE CONVERSION
KONVERZE HLASU

BACHELOR'S THESIS
BAKALÁŘSKA PRÁCE

AUTHOR — **PETER LUKÁČ**
AUTOR PRÁCE

SUPERVISOR — **Doc. Dr. Ing. JAN ČERNOCKÝ**
VEDOUCÍ PRÁCE

BRNO 2019

Brno University of Technology
Faculty of Information Technology

Department of Computer Graphics and Multimedia (DCGM) Academic year 2018/2019

Bachelor's Thesis Specification

21521

Student: **Lukáč Peter**
Programme: Information Technology
Title: **Voice Conversion**
Category: Speech and Natural Language Processing

Assignment:

1. Get acquainted with methods for speech analysis and synthesis. Get acquainted with parameters and methods for voice conversion.
2. Get acquainted with available toolkits for voice conversion and with VC Challenges.
3. Prepare data (for example from VC challenges, or own recordings).
4. Design and train a model for voice conversion from source to target speaker, perform the conversion and evaluate the results.
5. Suggest at least two non-trivial techniques for improvement of the conversion, implement them and evaluate.

Recommended literature:

- according to supervisor's advice

Requirements for the first semester:

- Items 1 to 4.

Detailed formal requirements can be found at http://www.fit.vutbr.cz/info/szz/

Supervisor: **Černocký Jan, doc. Dr. Ing.**
Head of Department: Černocký Jan, doc. Dr. Ing.
Beginning of work: November 1, 2018
Submission deadline: May 15, 2019
Approval date: May 10, 2019

Abstract

The matter of this thesis is voice conversion. Voice conversion is taking speech of one speaker, that we call source speaker and transforming it into speech that sounds as the speech of another speaker, that we call target speaker. This is accomplished using voice conversion system described in this thesis. As the framework for speech analysis and synthesis, we are using tool called STRAIGHT that was predominantly used in Voice Conversion Challenge 2016. Our voice conversion system is based on spectral conversion using feed-forward neural network and parallel training.

Abstrakt

Predmetom tejto práce je konverzia hlasu. Konverzia hlasu predstavuje preberanie reči jedného rečníka, ktorého nazývame zdrojový rečník a transformovanie tejto reči na reč ktorá znie ako reč druhého rečníka, ktorého nazývame cieľový rečník. Toto je dosiahnuté pomocou systému pre konverziu hlasu, ktorý je popísaný v tejto práci. Ako framework pre analýzu a syntézu reči používame STRAIGHT, ktorý bol dominantne používaný vo Voice Conversion Challenge 2016. Náš system pre konverziu hlasu je založený na konverzii spectra použitím doprednej neurónovej siete a paralelného trénovania.

Key words

Voice conversion, STRAIGHT, Voice Conversion Challenge, speech analysis, speech synthesis, DTW, Mel Filter Banks, A-Weighting, Regression Neural Network, Phoneme Recognizer

Kľúčové slová

Konverze hlasu, STRAIGHT, Voice Conversion Challenge, analýza reči, syntéza reči, DTW, Mel Filter Banky, A-Váhovanie, Regresná Neurónová Sieť, Fonémový Rozpoznávač

Reference

LUKÁČ, Peter *Voice Conversion.* Brno, 2019. Bachelor's thesis. Brno University of Technology, Faculty of Information Technology. Supervisor Doc. Dr. Ing. Jan Černocký.

Rozšírený Abstrakt

Konverzia hlasu predstavuje preberanie hlasu jedného rečníka (zdrojový rečník) a transformovanie tohto hlasu na hlas druhého rečíka (cieľový rečník). Konverzia hlasu je podkategória vačšieho oboru tzv. transformácia reči. V konverzii hlasu sa sústredíme na kvalitatívnu stránku reči, pričom linguistický obsah ponechávame nezmenený. Pre konverziu hlasu potrebujeme vyvinúť systém pre konverziu hlasu, ktorým sa zaoberá táto práca.

Úvod do Konverzie Hlasu

Konverzia hlasu je odvetvie, ktoré je v súčastnosti v rozvojovom stave. Súčastný stav vývoja monžno sledovať napr. vo Voice Conversion Challenge, na ktorom sa zúčastňujú vývojari systémov pre konverziu hlasu. Súhrn Voice Conversion Challenge 2016 poskytuje prehľad zúčastnených systémov. Vidíme že typicky je spektrum konvertované vo forme Mel cepstrálnych koeficientov alebo priamo ako celé spektrum. Na analýzu a syntézu reči sa veľmi často používa framework STRAIGHT.

Táto práca sa zemeriava na tzv. jedna k jednej konverziu tj. konvertujeme páry, ktoré pozostavajú z hlasu práve jedného zdrojového rečníka na hlas práve jedného cieľového rečníka. Alternatívne k tomuto existujú mnohozdrojové systémy ktoré konvertujú akéhokoľvek zdrojového rečníka na jediného cieľového rečníka.

Pre trénovanie používame nahrávky použité vo Voice Conversion Challenge 2016. Poskytnutí sú 4 zdrojoví a 4 cieľoví rečníci. Celková približná dlžka nahrávok jedného rečníka sú 4 minuty. Využívame paralelné trénovacie data, kde trénovaní rečníci prednášajú rovnaké vety. Alternatívne k tomu existujú neparalelné systémy, ktoré môžu byť trénované na rôznych vetách od zdrojového a cieľového rečníka.

Mnohozdrojové a neparalelné systémy prekračujú rozsah tejto práce.

Použité Nástroje

Pre analýzu a syntézu reči používame framework STRAIGHT. Výstupom analýzi je spektrogram, základný tón a aperiodická mapa, pričom sa zameriavame na konverziu spektrogramu a základného tónu.

PhnRec je fonémový rozpoznávač, ktorý používame pre rospoznanie foném, aby sme ich mohli analyzovať a upravovať ich dĺžku. V trénovaní tatktiež použijeme výstup rozpoznávača na odstránenie ticha.

Konverzia spektrogramu

Hlavným článkom systému pre konverziu hlasu je podsystém pre konverziu spektrogramu. Pred začatím konverzie musíme vytoriť trénovací systém. Konverzia aj trénovací systém používajú veľmi podobné techniky pre spracovanie spektrogramu. Tieto techniky musia mať reverzné operácie pre prevedenie konvertovaného spektrogramu na spektrogram, ktorý je vhodný na syntézu.

Prvým krokom pri trénovaní je zarovnanie zdrojového a cieľového spektrogramu, tak aby fonémy vyskytujúce sa v spektrogramoch boli na rovnakých indexoch. Nato sa používa Dynamic Time Warping (dynamické bortenie času). DTW lokálne predlžuje spektrogram a vytvára statické časti spektrogramu. Preto DTW je nasledované dodatočným spracovaním, tak aby zdrojový spektrogram bol zostal nezmenený a cieľový spektrogram sa dosadil na ten zdrojový.

Zarovnané spektrogramy sú nasledne prevedené do Mel frekvenčnej škály. Mel frekvenčná škála zvšuje rozlíšenie v nízkych frekveniach a znižuje vo vysokých, tak aby prevedený spektrogram reprezentoval ľudské frekvenčné vnímanie. Následne sa aplikujú A-váhy a logarimizuje sa výkon spektra.

V spektrograme sa následne normalizujú rámce spektrogramu na strednú hodnotu, aby sme odstránili energiu rámca ako parameter, ktorý musíme konvertovať. Následne normalizujeme spektrogram naprieč časom na strednú hodnotu a smerodatnú odchylku.

Na konverziu takto upraveného spektrogramu používame neurónovu sieť, ktorá je nastavená na vykonávanie regresie. Ako vstup používame zdrojový spektrogram, z ktorého berieme kontextové okno so šírkou približne 45ms. Ako výstup trénujeme cieľový spektrogram, ktorý sme rovnako upravili rovnakými technikami.

Pre trénovanie neurónovej siete poižívame Adam optimalizátor, ktorý je založený na stochastickom spáde gradientu. Tiež aplikuje adaptívný stupeň učenia a adaptívne momentum.

Konverzia základného tónu

Pvrotné experimenty s konverziou základného tónu predpokladali použitie neurónovej siete s podobnou architektúrou ako neurónová sieť pre konverziu spektrogramu. Ukázalo sa že priebeh základného tónu od zdrojového a cieľového rečníka má nízku koreláciu, preto sa od neurónovej siete upustilo. Používa sa konverzia na základe normalizácie (podľa zdrojového rečníka) konvertovaného základného tónu na následnej denormalizácie (podľa cieľového rečníka).

Zmena Dĺžky Foném

Dĺžka s akou jednotlivý rečníci vyslovujú fonémy sa mení. Pomocou fonémového rozpoznávač môžeme analyzovať ich dĺžku. Trvanie fonémy jedného typu má iné normálne rozloženie u zdrojového a cieľového rečníka. Na základe tohto môžeme pri konverzii zmeniť dĺžku foném normalizovaním strednou hodnotou a smerodatnou odchylkou zdrojového rečníka a denormalizovaním strednou hodnotou a smerodatnou odchylkou cieľového rečníka.

Experimenty a Výsledky

Pred použitím konverzie spektrogramu pomocou neurónovej siete sme experimentovali s konverziou pomocou transformačnej matice, ktorá je vypočítana ako minimálne kvadratické riešenie rovnice $SX = T$ kde S je zdrojový spektrogram a T je cieľový spektrogram a X je transformačná matica. Toto riešenie nedokázalo vypočítať X pre trénovacie data s dostatočnou presnosťou, ešte horšie generalizovať pre evaluačné data.

Finálne riešenie používajúce neurónovú sieť dosahuje lepšie výsledky. Napriek tomu stále je čo zlepšovať. Vlastnosti konvertovaného spektrogramu sú také, že syntetizovaná reč sa podobá tej cieľového rečníka. Avšak konvertovaný spektrogram trpí strátou detailu reči. To sa hlavne prejavuje nevýraznými až úplne vytratenými formantami. V syntetizovanej reči sa stráta formant prejavuje nízkou kvalitou hlasu, ťažko zrozumenuteľnými slovami a celkovou otupenosťou.

Dôvody takýchto výsledkov môžu byť rôzne. Jedným z nich môže byť malý obsah trénovacích dát, ktoré majú len okolo 4 minút. Ďalším dôvodom môže byť charakter konvertovaných spektrogramov a to taký, že jednotlivé časti ktoré sa na seba konvertujú sú medzi sebou tak dekorelované, že neurónová sieť nedokáže nájsť ideálne riešenie.

Voice Conversion

Declaration

I declare that this thesis is the original work of the author and was supervised Doc. Dr. Ing. Jan Černocký. All sources used in creating this thesis are acknowledged by references.

.....
Peter Lukáč
May 15, 2019

Acknowledgments

I would like to thank my supervisor Jan Černocký, who provided me with abundance of information and resources for speech processing, including phoneme recognizer. I would also like to thank Dr. Hideki Kawahara, the creator of STRAIGHT, who provided us with source code for this tool.

Contents

Chapter 1

Introduction

Voice conversion is taking voice of one speaker (the source speaker) and transforming it into voice that sounds as the voice of another speaker (the target speaker). Voice conversion is subcategory of larger field called voice transformation. Voice transformation is broad range of modifications we can apply to the human voice. In voice transformation we focus on non-linguistic information in voice. In voice conversion, this means we focus on features that make voice of one speaker sound distinct from the voice of another speaker.

1.1 Introduction Into the Voice Conversion

The voice conversion is currently developing field in the speech processing. Development of the voice conversion has potential to be utilized in different areas of the speech processing such as generating expressive speech, voice assistants and more. State-of-the-art of the voice conversion is well shown at the Voice Conversion Challenge [1]. The Voice Conversion Challenge summary and analysis [2] provides overview of the participating systems. The voice conversion systems usually use Mel cepstral coefficients as spectral envelope that is converted usually using Gaussian mixture models or deep neural networks.

This thesis focuses on one-to-one voice conversion. This means, the voice conversion system will be able to be trained for pairs of one source speaker and one target speaker for which we have to acquire speech. After training, the voice conversion will be evaluated on evaluation speech. Our voice conversion system will be trained on parallel training data meaning that both source and target speaker training speech need to uttered the same sentences. This kind of voice conversion training and evaluation is inspired by Voice Conversion Challenge from which we have acquired training and evaluation data.

There is an alternative form of training to the parallel training that is called non-parallel training. In non-parallel training we collect training speech of the source and target speaker that doesn't contain the same sentences. Instead we collect any sentences from the speakers and use e.g. cross-referencing training system. There is also alternative to the one-to-one voice conversion called any-to-one voice conversion in which we can convert any source speaker that we haven't used in training and convert it into the target speaker. Non-parallel training and any-to-one voice conversion systems are beyond the scope if this thesis.

1.2 Baseline System

The design of our voice conversion system is following a simple scheme. The scheme consists of three parts: analysis, modification and synthesis. The analysis 3 and synthesis are provided by STRAIGHT framework. This is described in detail in separate chapter.

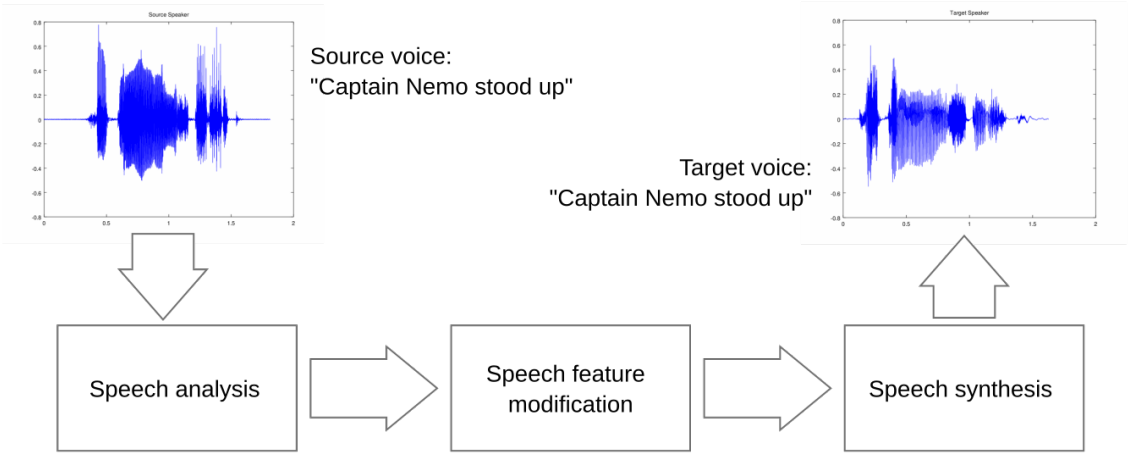


Figure 1.1: Voice conversion baseline system

Chapter 2

Voice production

Before we can extract and modify voice features we have to understand what kind of features we deal with and how they matter in voice conversion. The most common way to understand speech production in the context of voice transformation is the use of source-filter model [3].

2.1 Source-filter theory model

As the name suggests, this model is composed of two parts: source and filter. The source (vocal cords) produces excitation signal that passes through time-varying filter (vocal tract) that damps, or emphases frequencies in that signal, thus creates output signal (voice).

Depending on kind of phoneme being process, we distinguish two kinds of phonemes: voiced and unvoiced. Excitation signal for voiced phonemes is composed of fundamental frequency and multiples of that frequency. For unvoiced phonemes, the excitation signal reminds white noise and fundamental won't be relevant for modification of the frames.

2.1.1 Excitation signal

An accurate way to model the excitation signal is suggested by the family of speech representations called sinusoidal models. This model suggests that the excitation signal $e(t)$ is represented by a sum of sinusoids defined as follows:

$$e(t) = \sum_{k=0}^{K(t)} a_k(t) e^{i\phi_k(t)}, \tag{2.1}$$

"*where $a_k(t)$ and $\phi_k(t)$ is are the instantaneous excitation amplitude and phase of the k-th sinusoid and $K(t)$ is the number of sinusoids, which may vary in time*" [3]. Now we can specify the most important feature of the excitation signal is pitch that we will refer to as fundamental frequency: $f_0(t)$. The fundamental frequency is added in the sum of sinusoids as follows:

$$\phi_k(t) = 2\pi k f_0(t), \tag{2.2}$$

The excitation signal therefore consists of fundamental frequency $f_0(t)$ and multiples of the fundamental frequency k, while the higher frequencies get gradually dumped.

2.1.2 Filter model

The filter that is provided by vocal tract is time-varying filter which creates distinct phonemes over time. Voice analysis is performed by frame-by-frame approach with frame size raging from 20ms to 50ms. In each frame we assume that voice characteristics including filter for excitation signal are stationary.

2.1.3 Output signal

Consider that in a frame we have time invariant filter $h(t)$. The output signal(voice) $s(t)$ is then convolution of excitation signal $e(t)$ and the impulse response of the vocal tract filter $h(t)$:

$$s(t) = \int_0^{\tau} h(t-\tau)e(\tau)d\tau \tag{2.3}$$

The voice production process can be visualized in the spectral domain as follows:

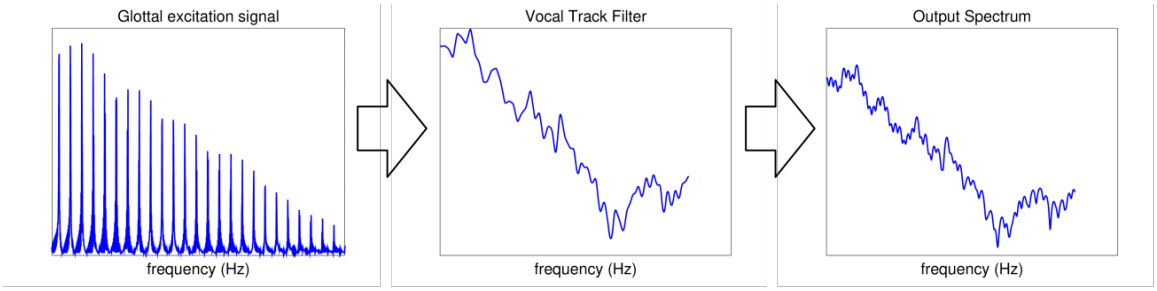


Figure 2.1: Source Filter theory model

Chapter 3

STRAIGHT

STRAIGHT is a tool for speech analysis, manipulating voice quality and speech synthesis. STRAIGHT was predominantly used tool in Voice Conversion Challenge 2016 [1] and systems using this tool achieved outstanding results. The main justification for using STRAIGHT is its approach for voice speech analysis.

STRAIGHT decomposes speech into three components: fundamental frequency, aperiodicity map and interference and fundamental frequency free spectrogram. These three components are used for speech synthesis:

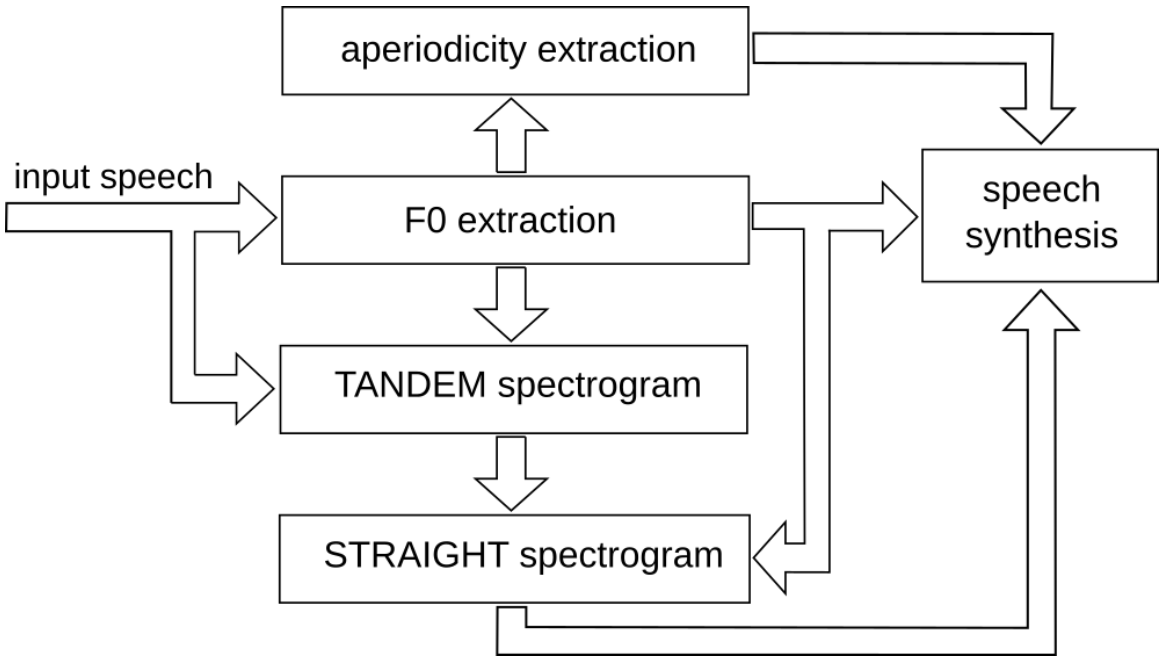


Figure 3.1: Diagram of STRAIGHT speech analysis and synthesis [4]

STRAIGHT allows us to set settings for speech analysis such as minimal and maximal fundamental frequency, fast Fourier transform size etc. Our system is using fft size 1024, and frame offset 3 milliseconds.

3.1 Fundamental frequency

Fundamental frequency is the first component to be estimated, because aperiodicity estimation and spectrogram depend on it. STRAIGHT estimates fundamental frequency for all frames including silent and unvoiced frames. Detection of silent and unvoiced frames is applied later.

F_0 extractor [4] uses division of two spectrums: TANDEM spectrum and STRAIGHT spectrum. However TANDEM and STRAIGHT spectra depend on fundamental frequency. Therefore F_0 creates band table including frequencies throughout expected frequency range (typically 50Hz - 600Hz). Each frequency from band table is used as input frequency for TANDEM and STRAIGHT spectrum and spectra are divided. The closer the band frequency is to the actual F_0 the more power in divided spectrum we get. Divided spectra are cumulated and F_0 candidates are extracted based on gradient of cumulative frequency. Final F_0 is acquired by auto-tracking F_0 candidates.

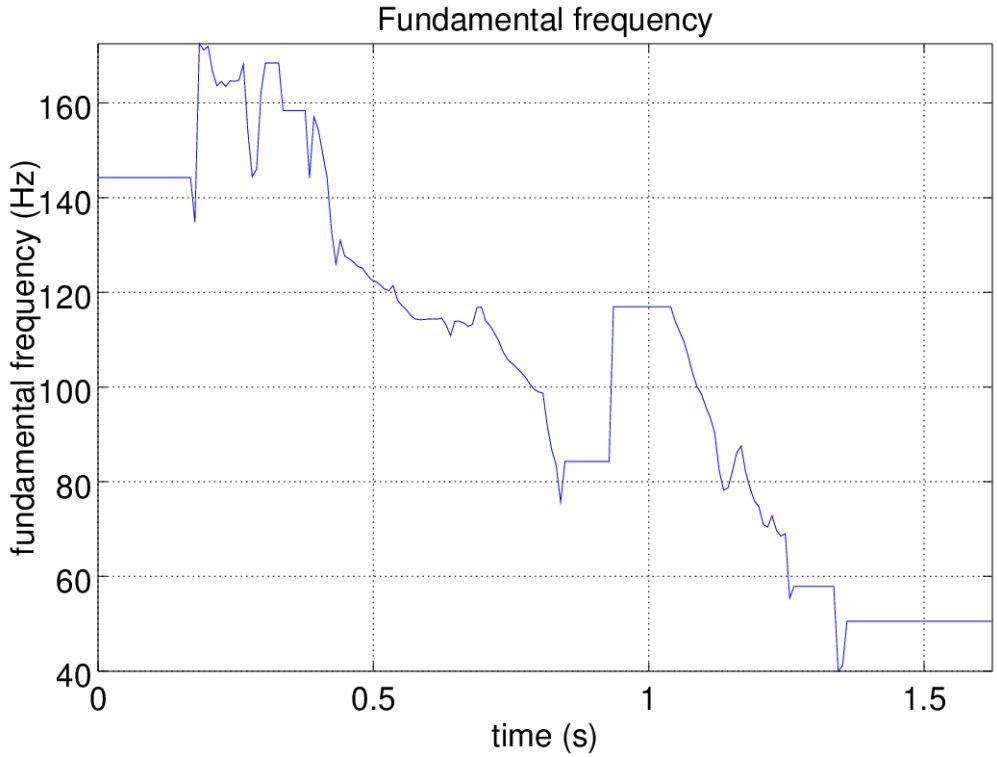


Figure 3.2: Fundamental frequency of speech: "Captain Nemo stood up."

3.2 Aperiodicity map

Aperiodicity map is estimation that describes how how random is power ratio between periodic component(F_0) and random component across spectrum. Parts of spectrum with low aperiodicity represent high presence of periodic excitation signal. Parts with of spectrum with high aperiodicity represent low presence of periodic excitation signal. Aperiodicity map is necessary for speech synthesis in order to create excitation signal.

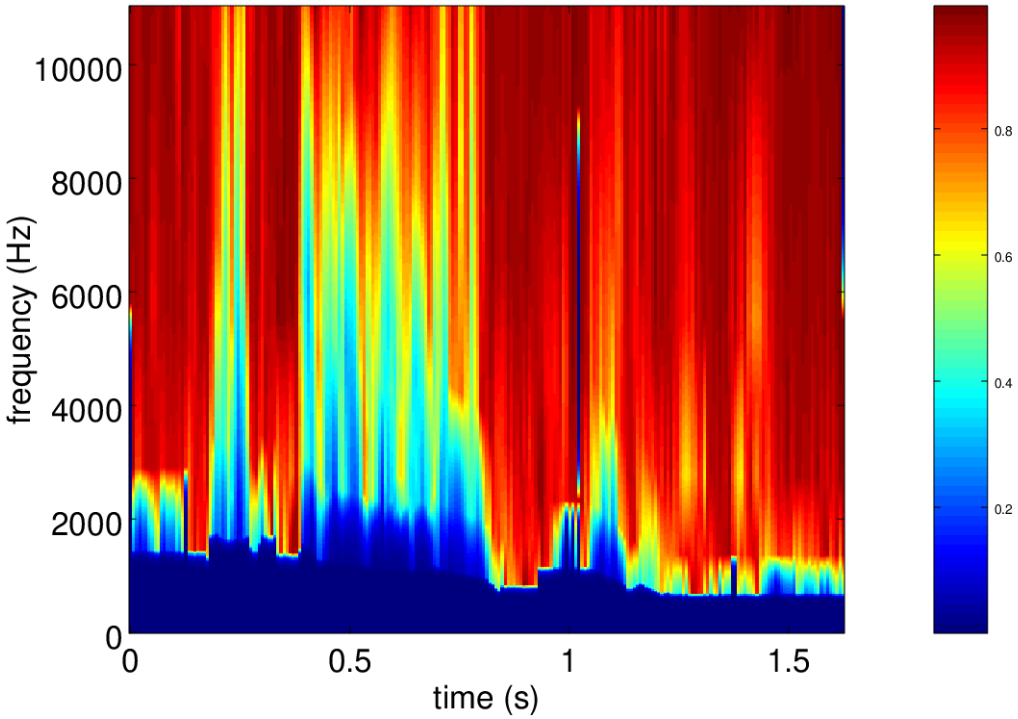


Figure 3.3: Aperiodicity map of speech: "Captain Nemo stood up."

3.3 STRAIGHT spectrogram

Spectrogram extraction has two steps. First TANDEM spectrogram is extracted and then final STRAIGHT spectrogram is estimated. TANDEM spectrogram is composed of static power spectrum that has greatly reduced temporal variation. STRAIGHT spectrogram builds on TANDEM spectrogram and smoothens periodic component.

3.3.1 TANDEM spectrogram

TANDEM spectrogram($P_T(w,\tau)$) [4] removes temporal variation by using average spectrum of two complementary windows:

$$S(\omega,\tau) = \int x(\tau)w(\tau - t)e^{-jw\tau}d\tau \tag{3.1}$$

$$P_T(w,\tau) = \frac{|S(w,t - T_0/4)|^2 + |S(w,t + T_0/4)|^2}{2}, \tag{3.2}$$

where $x(t)$ and $w(t)$ represent waveform and T_0 represents reciprocal of fundamental period.

3.3.2 TANDEM to STRAIGHT spectrogram

STRAIGHT spectrogram removes fundamental frequency and its multiplications that are present in TANDEM spectrogram. That makes STRAIGHT spectrogram ideal represen-

tation of vocal tract filter. STRAIGHT spectrogram $P_{ST}(\omega, t)$ [4] is obtained by spectral smoothing of the TANDEM spectrogram P_T:

$$C(\omega, t) = \int_{\omega L}^{\omega} P_T(\lambda, t) d\lambda \tag{3.3}$$

$$L(\omega, t) = \ln(C(\omega + \omega_0/2, t) - C(\omega - \omega_0/2, t)), \tag{3.4}$$

followed by consistent sampling:

$$P_{ST}(\omega, t) = \exp(\tilde{q_1}(L(\omega + \omega_0, t) + L(\omega - \omega_0, t)) + \tilde{q_0} L(\omega, t)), \tag{3.5}$$

where ω_0 represents fundamental angular frequency. $\tilde{q_0}$ and $\tilde{q_1}$ represent compensation constants calculated from auto-correlation of the Fourier transform [4].

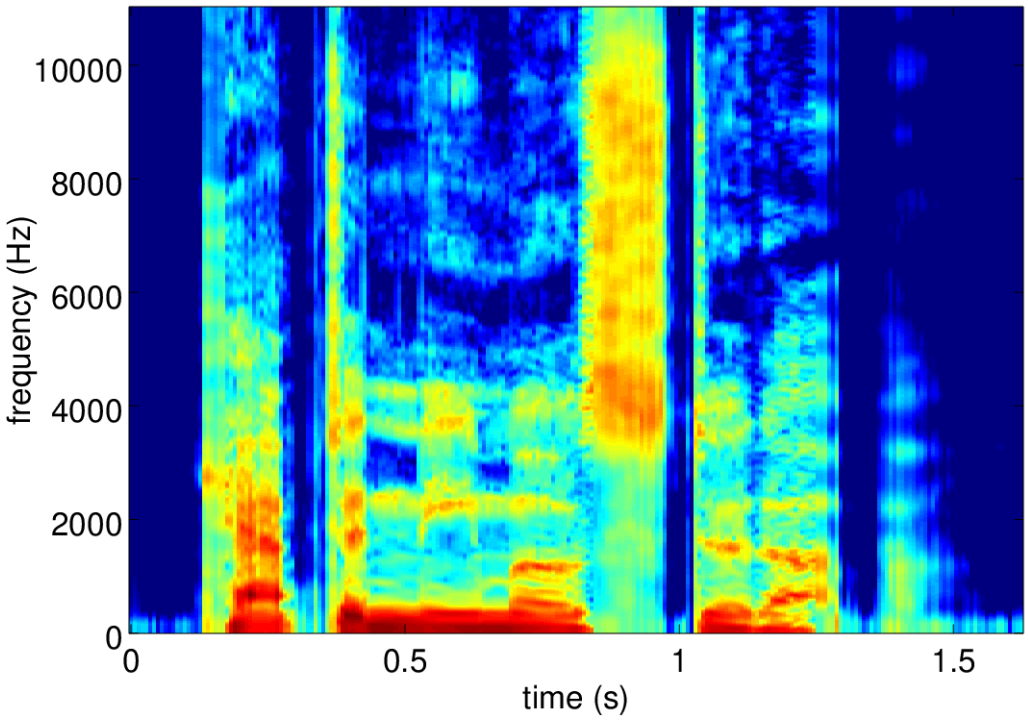


Figure 3.4: STRAIGHT spectrogram of speech: "Captain Nemo stood up."

Chapter 4

Phoneme Recognition

A phoneme is the smallest unit of sound of language that can be distinguished by a listener. There are more phonemes than letters in alphabet because some phonemes are created by combinations of multiple letters. Phonemes can be represented by phonetic labels associated with example of the sound of that phoneme. Most common phonetic labels are IPA [5], Worldbet [5], and OGIbet [5].

When spoken, phonemes have different length, depending on the speed of speakers speech. Frequency of phoneme appearance can also change speaker to speaker. Those are features we want to observe and change in the voice conversion system.

4.1 Phoneme recognizer based on long temporal context

Phoneme recognition is achieved using phoneme recognizer. Phoneme recognition is complex task deserving its own thesis. For this purpose, we were provided with phoneme recognizer based on long temporal context developed at Brno University of Technology, Faculty of Information Technology [6]. The phoneme recognizer is based on hybrid ANN/HMM approach, where artificial neural networks (ANN) are used to estimate posterior probabilities of phonemes from Mel filter bank log energies using the context of 310ms around the current frame [7].

4.1.1 Phoneme Recognizer Output

The recognizer provides 4 systems for 4 languages (Czech, Hungarian, Russian, English). The output of the recognizer provides sequence of OGIbet phonetic labels with their temporal positions and likelihood. Example of recognized phonemes with English system in speech *"Tall, black Crosewood bookcases."* looks like:

```
000000 1200000 pau -12.011214
1200000 2300000 t -14.992387
2300000 4000000 hh -42.520073
4000000 4400000 l -8.518959
4400000 4800000 ah -9.891708
4800000 5600000 m -14.068726
...
```

Chapter 5

Spectrogram Conversion

Spectrogram is the main feature that gives s speakers voice its characteristic. Changing spectrogram is the main focus of most of the voice conversion systems and usually it is the largest part, consisting of multiple stages. All the stages are described in separate chapter 6

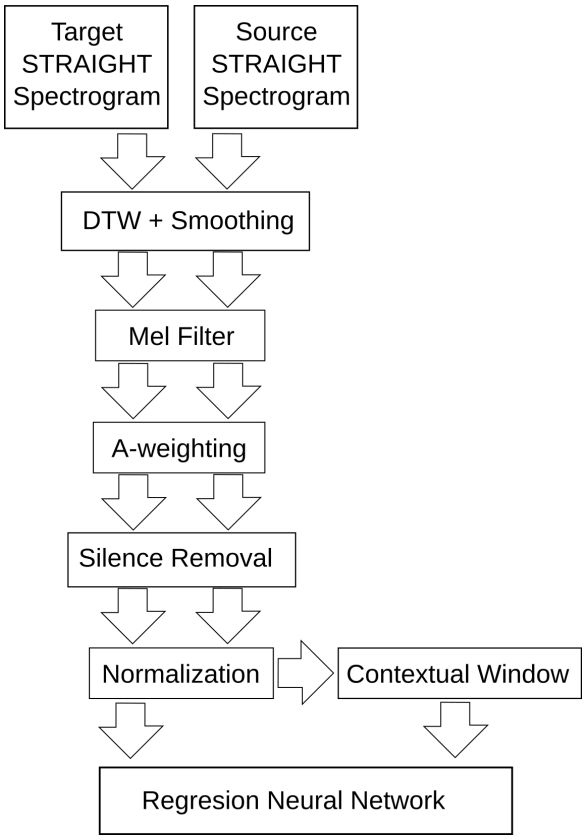


Figure 5.1: Diagram of the training system pipeline

5.1 Training System

Before we can proceed to the actual conversion, we have to collect information that will provide data for the conversion system. The training system will require training samples

of the source and target speaker in order to collect information. We are developing parallel conversion system, meaning, we require that the training system gets accurate conversion parameter from the training samples consisting of speech of the pair of speakers speaking the same transcripts.

5.1.1 Training System Pipeline

Training is accomplished using training system is composed of similar components as the conversion system will be composed. The input of the training is a pair of spectrograms (source and target). During the training, we save trained parameters, namely regression neural network weights, normalization parameters (means and standard deviations). Each part of the pipeline is described in the following chapter 6.

5.2 Spectrogram Conversion System

Spectrogram conversion requires that we have accomplished training and saved parameters acquired during training. Spectrogram conversion will use similar pipeline, however trained parameters will be used to perform conversion. Also, the input for the pipeline is only source spectrogram that from the evaluation data-set. After conversion reversed operations will be applied to get fully converted spectrogram ready for synthesis.

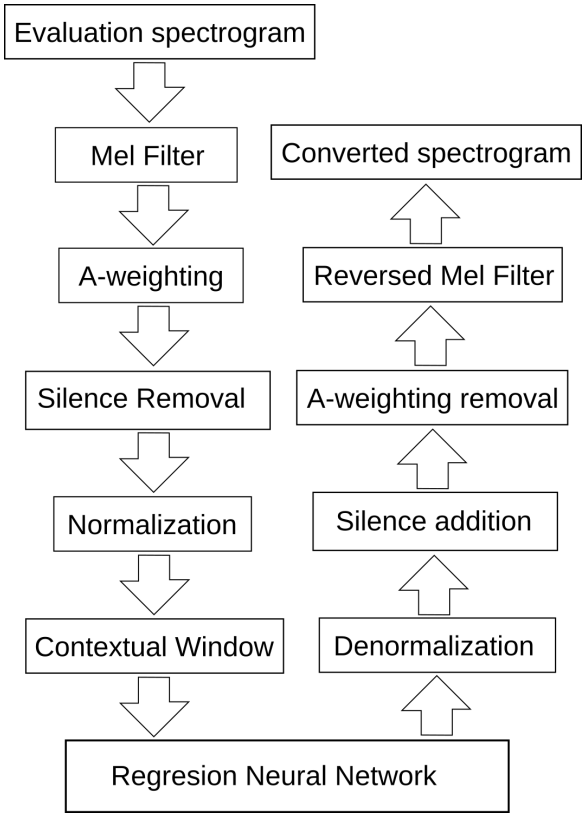


Figure 5.2: Conversion pipeline

5.2.1 Conversion Pipeline

Conversion pipeline removes the pair of the spectrograms as the input since we are not performing training anymore. The regression neural network is set to perform prediction of the input. The output is converted spectrogram that we have to de-process. De-processing is composed of the same techniques however we apply them in the reversed order. Also each technique must provide reversed operation. De-processed spectrogram is ready for synthesis.

Chapter 6

Techniques for Spectrogram Conversion

In order to make STRAIGHT spectrogram suitable for conversion we need to apply number of modification to it. This modifications focus on enhancing spectrogram into how it perceived by human ear and adjusting it to be processed by voice conversion training subsystem. The techniques also need have inverse transformations used by voice conversion system.

6.1 Alignment of Spectrograms

Training data from source and target speaker that are passed to the training subsystems have to be aligned. More specifically we match indexes of the frames of the source and target spectrograms so that frames correspond to the same segment of the speech. Procedure we use is called dynamic time warping [8].

6.1.1 Dynamic Time Warping

Since STRAIGHT spectrum is only power spectrum that was processed to remove harmonic component, for dynamic time warping it has to be represented in a form that reflects human perception of frequencies and magnitudes and also improves speech processing.

Two essential steps that are used often in the voice conversion system are applying Mel filter banks 6.3 and A-weighting 6.4 or taking logarithm of Mel filter banks. This two steps must be applied to the STRAIGHT spectrum before we can achieve good results with DTW.

6.1.2 Local Distance Matrix

At first we calculate local distance matrix. Local distance matrix contains distance of each source frame compared to each target frame. Distance between frames is calculated as Euclidean distance.

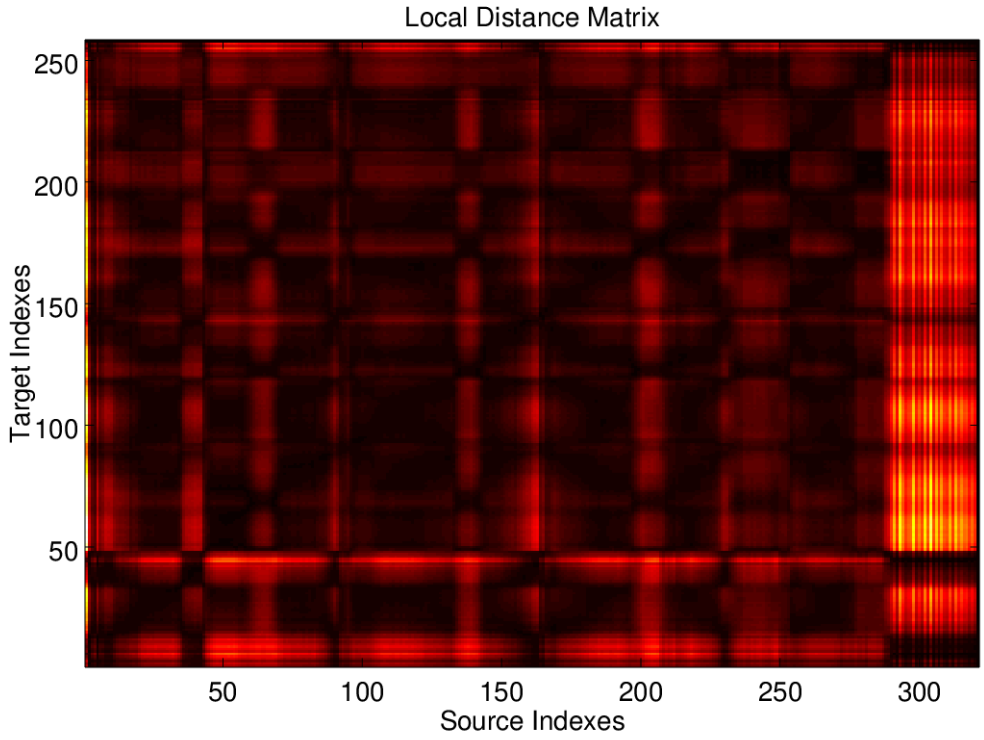


Figure 6.1: Local distances between all frames of source and target spectrograms

6.1.3 Cumulated Distance Matrix and Backtracking Matrix

While Local distance matrix provides euclidean distance between frames without context, we need to track distance based on position we come from. For this purpose we calculate cumulated distance matrix (CDM) using local distance matrix and local path restrictions. CMD is calculated column by column from the place we start (indexes (1, 1)) and simultaneously we create backtracking matrix we use later. Local path restrictions set three constraints: a) path for reaching new place b) coefficients paths c) path tagging.

New calculated distance $g(m+1, n)$ depends on value of the current location $g(m, n)$ and values of the surrounding locations: $g(m+1, n)$, $g(m, n+1)$. New distance is the lowest value g. Also, we save tag of the associated path in the backtracking matrix.

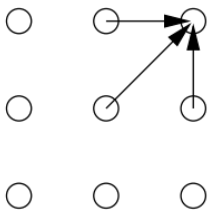
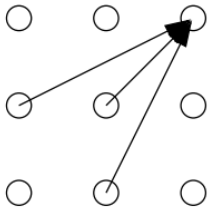
Typ		α	β	Typ $w(k)$	$g(n,m)$
I.		0	∞	a	$\min \begin{cases} g(n,m-1)+d(n,m) \\ g(n-1,m-1)+2d(n,m) \\ g(n-1,m)+d(n,m) \end{cases}$
				d	$\min \begin{cases} g(n,m-1)+d(n,m) \\ g(n-1,m-1)+d(n,m) \\ g(n-1,m)+d(n,m) \end{cases}$
II.		$\frac{1}{2}$	2	a	$\min \begin{cases} g(n-1,m-2)+3d(n,m) \\ g(n-1,m-1)+2d(n,m) \\ g(n-2,m-1)+3d(n,m) \end{cases}$
				d	$\min \begin{cases} g(n-1,m-2)+d(n,m) \\ g(n-1,m-1)+d(n,m) \\ g(n-2,m-1)+d(n,m) \end{cases}$

Figure 6.2: Different types of path restrictions [8]. We are using Type I.a

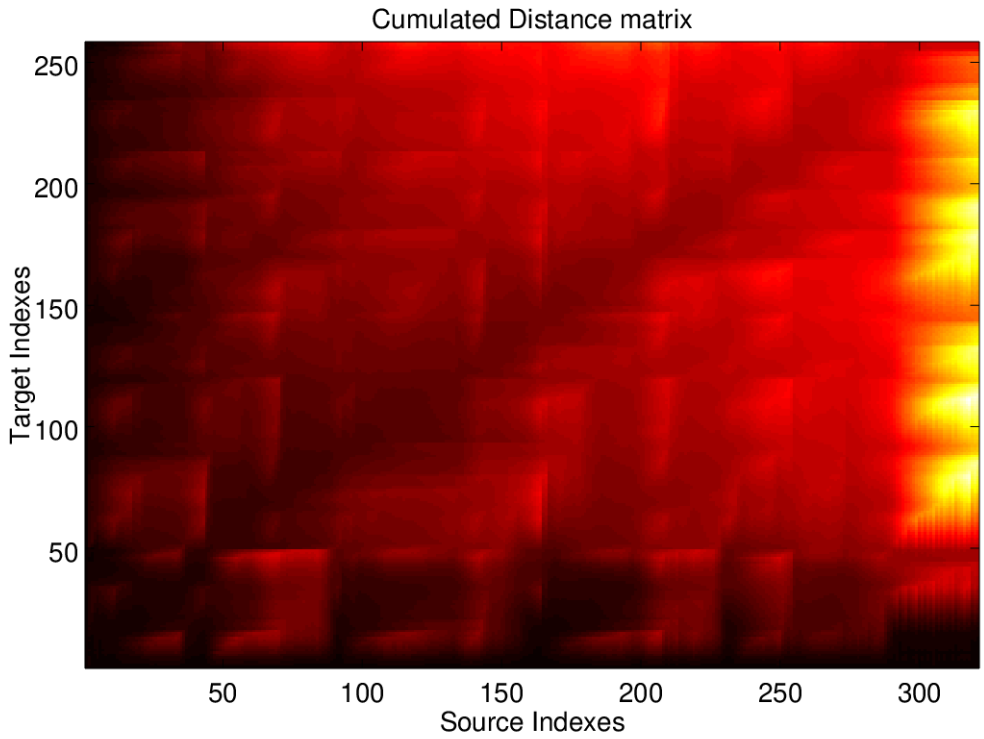


Figure 6.3: Cumulated distances between all frames of source and target spectrograms

6.1.4 Backtracking

In backtracking we create two arrays of paired indexes based on backtracking matrix. Cells of backtracking matrix contain tag that says where we get to the given cell from. Backtracking starts from the end of the path (maximum indexes) and using tagged directions leads to beginning. The reversed arrays then represent the shortest path.

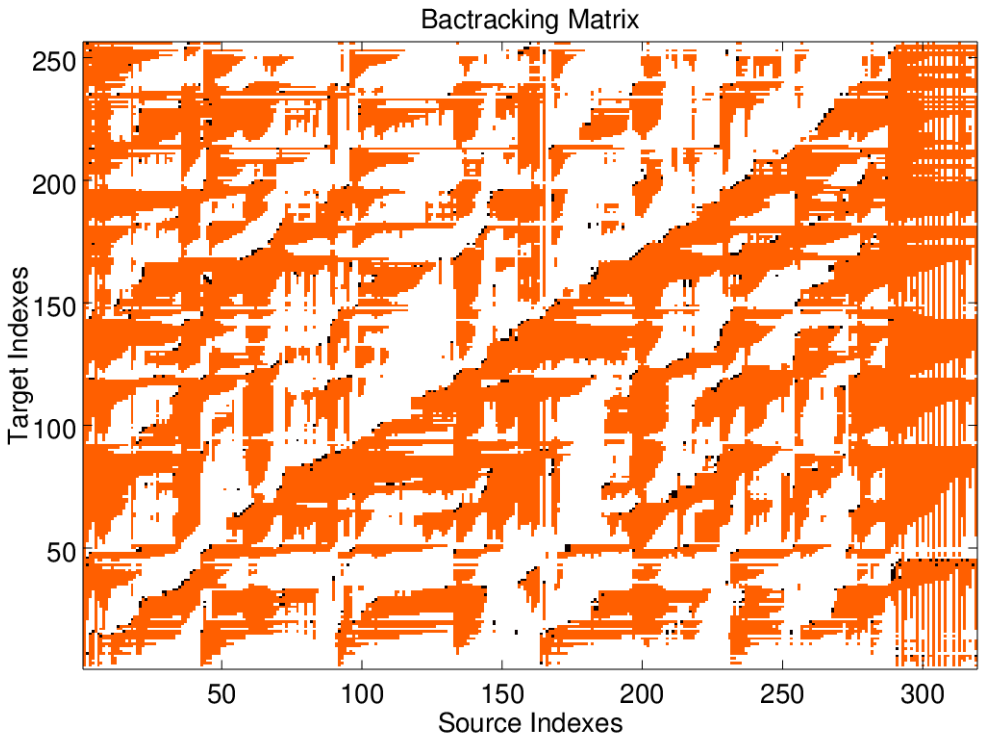


Figure 6.4: Backtracking Matrix indicates direction we approach cells from. Black cells are approached diagonally. Orange cells are approached from the bottom. White cells are approached from the left.

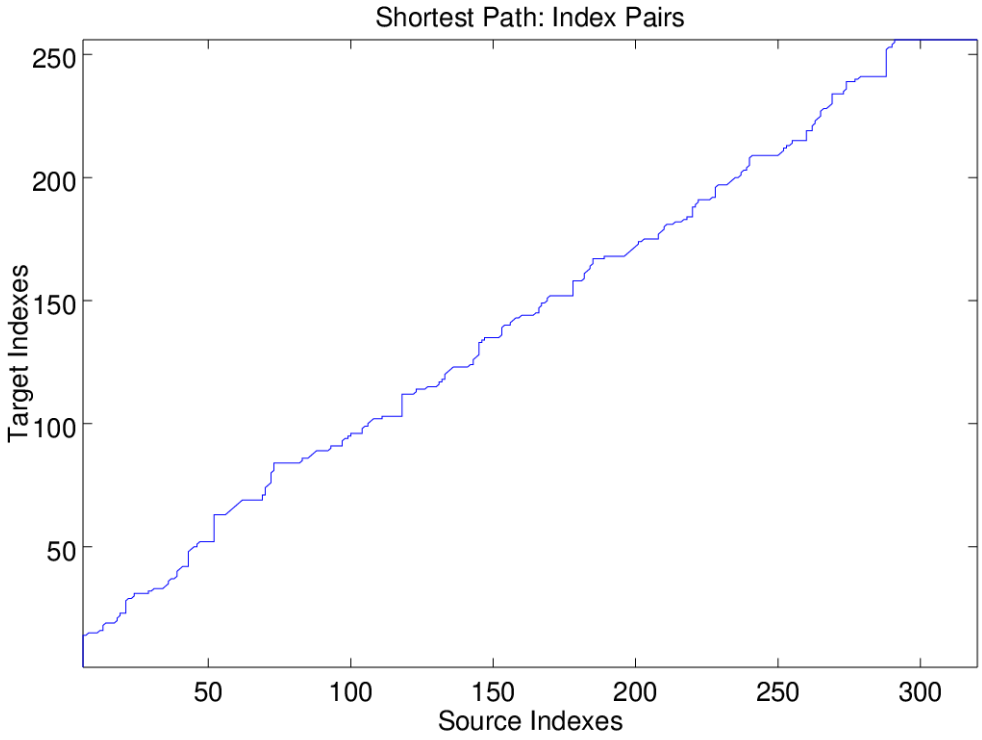


Figure 6.5: Shortest Distance Path

6.2 Postprocessing

DTW provides shortest path between two spectrograms, however, it also creates redundancy i.e. duplicated frames that create still sequences of frames in the spectrogram. Duplicated frames extend spectrogram locally that creates representation not suitable for context dependent training subsystems. For better representation of aligned spectrograms, we require that the source spectrogram is the same sequence of frames as the original spectrogram and the target spectrogram is aligned to fit the source spectrogram.

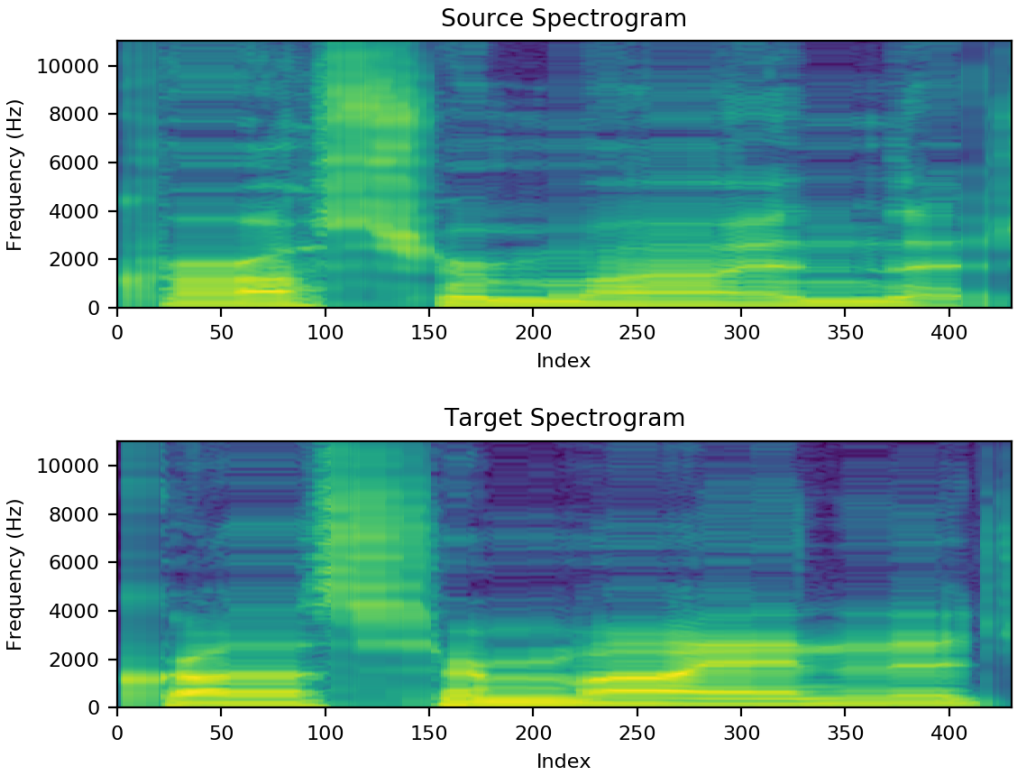


Figure 6.6: Spectrograms aligned using DTW. Static areas disrupt context e.g. the source spectrogram in the frames 25-50 or the target spectrogram in the frames 50-100 are locally extended

6.2.1 Duplicated Frames Removal

The first step of postprocessing is removal of duplicated frames in both source and target arrays. The shortest path with removed duplicated frames now contains cuts in spectrogram opposed to the spectrogram where duplicated frames were removed and therefore this representation is still not suitable for context-depended trained subsystem. The shortest path with removed duplicated frames contains only pairs of frames with the smallest distance.

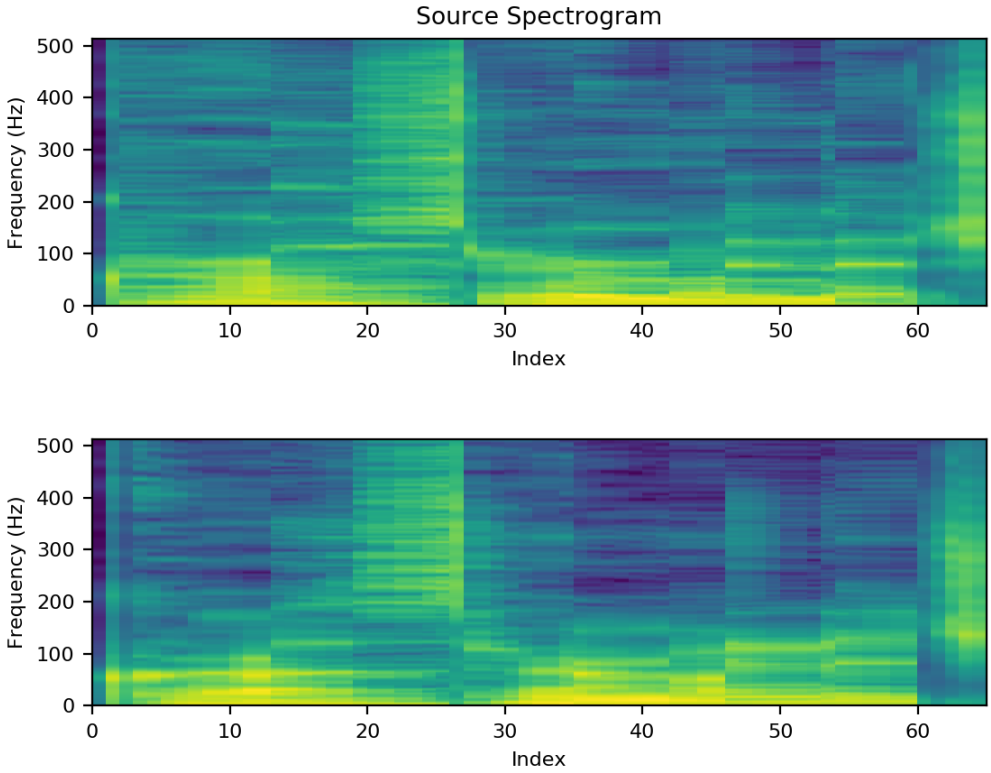


Figure 6.7: Spectrograms aligned using DTW with removed duplicities. Lots of content is removed because duplicity on one side also removes good frames on the other side

6.2.2 Smoothing Shortest Path

Ultimately we want to have unchanged source spectrogram and target spectrogram aligned to it. We use shortest path with removed duplicated frames, from which we take set of index pairs, spaced by small distance, 40-60ms. Parts of target spectrogram are resized with interpolation so that distance between neighbor target indexes match distance between responding source indexes.

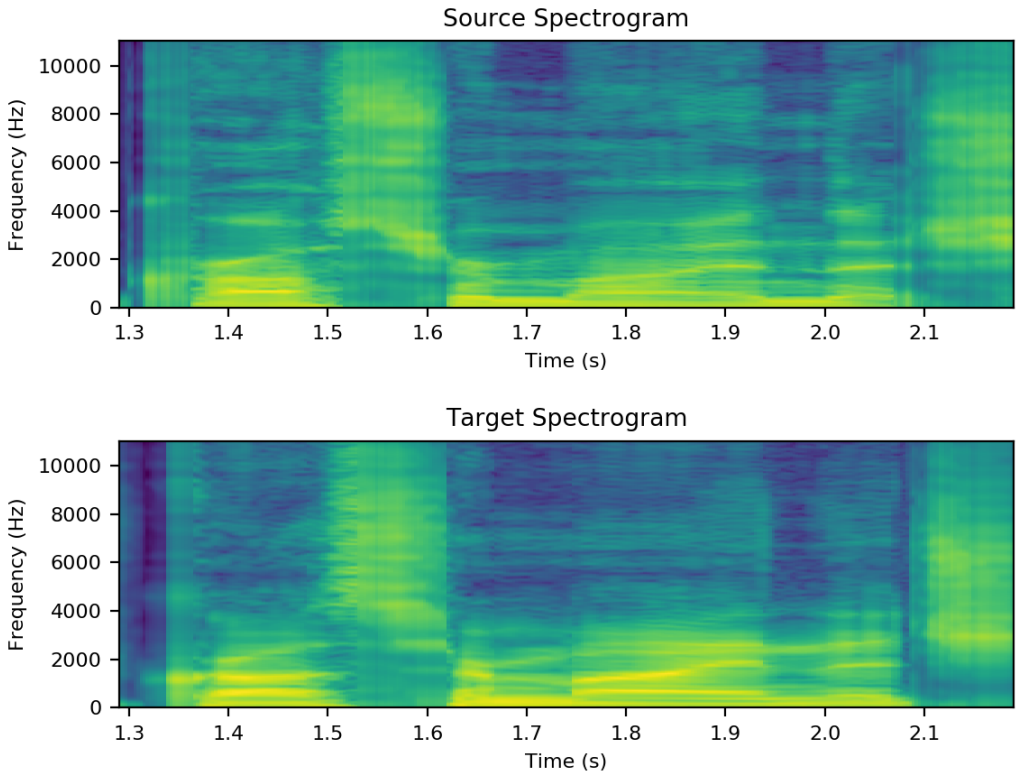


Figure 6.8: Source spectrogram remains unchanged from the analysis. Parts of the target spectrogram were resized according to the extracted index pairs of the shortest path.

6.3 Mel Filter Banks

Mel filter banks transform power spectrum to Mel power spectrum that has different frequential representation. Mel frequential representation allows further processing algorithms to process spectrum as if frequencies were perceived by human rather than fully linear representation.

6.3.1 Mel Scale

Human perception has higher frequential definition on lower frequencies than on higher frequencies. For instance two of musical notes (one octave apart) in low registers i.e. C_1 and C_2 are 32.70 Hz apart however same notes in higher registers i.e. C_5 and C_6 are 523.25 Hz apart. Frequencies of the power spectrum will be converted to the Mel scale [9]:

$$F_{Mel} = 2959 \log_{10}(1 + \frac{F_{Hz}}{700}) \tag{6.1}$$

or:

$$F_{Mel} = 1125 \ln(1 + \frac{F_{Hz}}{700}) \tag{6.2}$$

For Mel to frequency we use following formula:

$$F_{Hz} = 700(\exp(F_{Mel}/1125) - 1) \tag{6.3}$$

6.3.2 Filter Banks

In order to rescale frequencies to Mel, we can take range of evenly spaced Mel scale values and convert them to frequencies. Linear spaced Mel values will be nonlineary spaced on frequency scale, having smaller spaces at lower frequencies and bigger spaces at higher frequencies accounting for the human ear frequency definition.

New Mel converted spectrum can be calculated by taking the sum of the power of the parts preserved by rescaled frequencies, therefore low Mel registers taking smaller parts of spectrum and high Mel registers taking bigger parts of spectrum. Resulting in higher frequencies of spectrum being shrinked and lower frequencies of spectrum being enlarged. The most popular way to select preserved parts of spectrum is to use triangular shape filter.

Each triangular filter spans over 3 rescaled frequencies (up slope on 1-2 and down slope on 2-3) so that the filters overlap. Each filter provides 1 value for Mel spectrum by summing parts of the spectrum it spans. The triangular shape is linear function that multiples the value it is spanning over, parts of the spectrum under the ends of bank being taken by smaller margin than value under the top of the bank. The maximum value of the filter can also by adjusted. The maximum value of the filter can be normalized by its width, wider banks being lower. This accounts for the size of the area of the spectrum so that wider banks don't get too high values.

Normalized Mel filter banks are used during spectrum to Mel conversion. Denormalized Mel filter banks are used during backward conversion to full spectrum. During backward conversion we take value of each Mel banks and spread its value to the spectrum according to the shape of the bank.

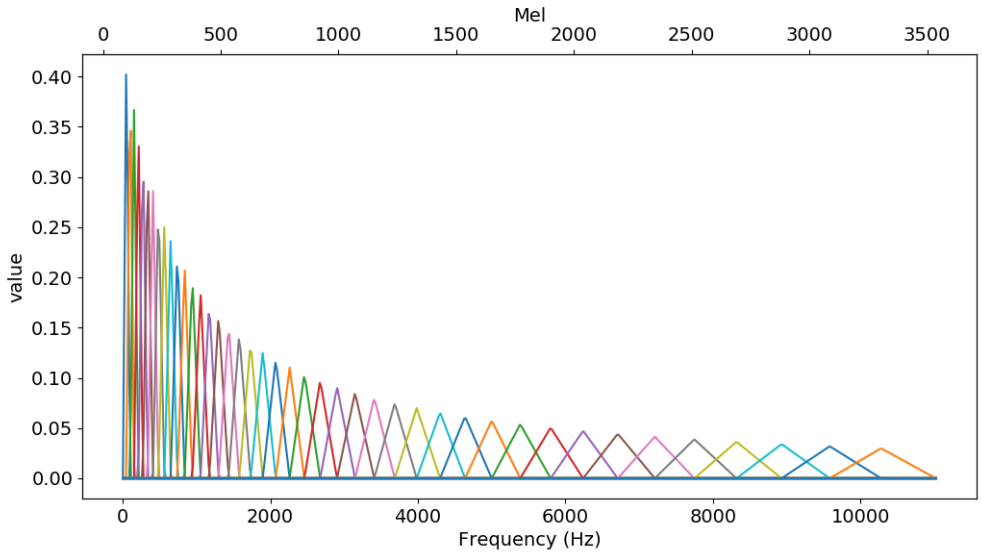


Figure 6.9: Normalized Mel Filter Banks with 42 banks. Also shown with Mel scale on the top

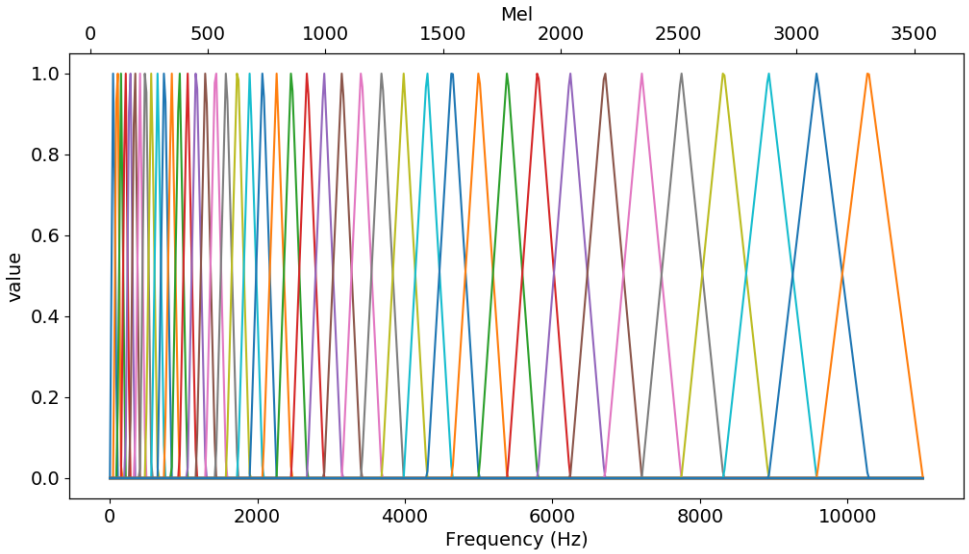
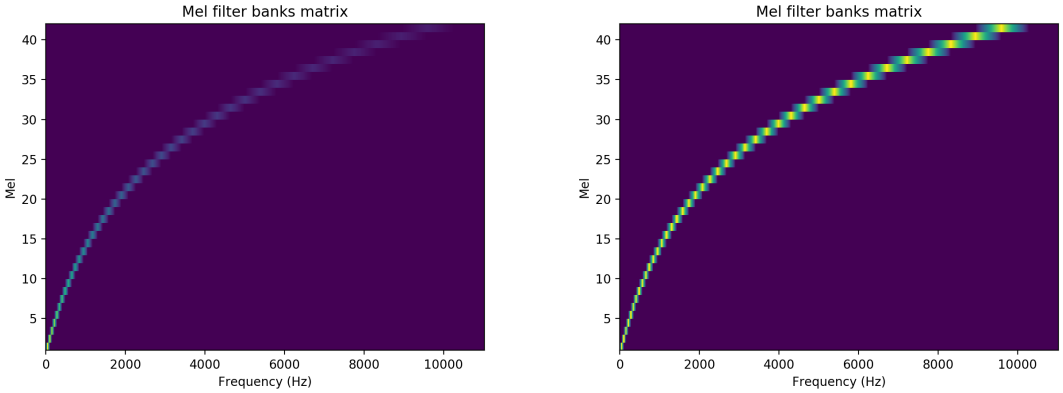


Figure 6.10: Denormalized Mel Filter Banks with 42 banks. Also shown with Mel scale on the top

The Mel filter banks are stored as matrix that is easily applied to the spectrogram for frequency to Mel conversion and also reversed Mel to frequency conversion.



(a) Normalized matrix for frequency to Mel conversion

(b) Denormalized matrix for Mel to frequency conversion

The Mel filter bank matrix is applied as:

$$\mathbf{S}_M^T = \mathbf{S}_F^T \mathbf{M}^T, \tag{6.4}$$

$$\mathbf{S}_F^T = \mathbf{S}_M^T \mathbf{M}, \tag{6.5}$$

where $\mathbf{S}_F$ is spectrogram in normal frequency, $\mathbf{S}_M$ is spectrogram is Mel scale and $\mathbf{M}$ is the Mel filter bank matrix.

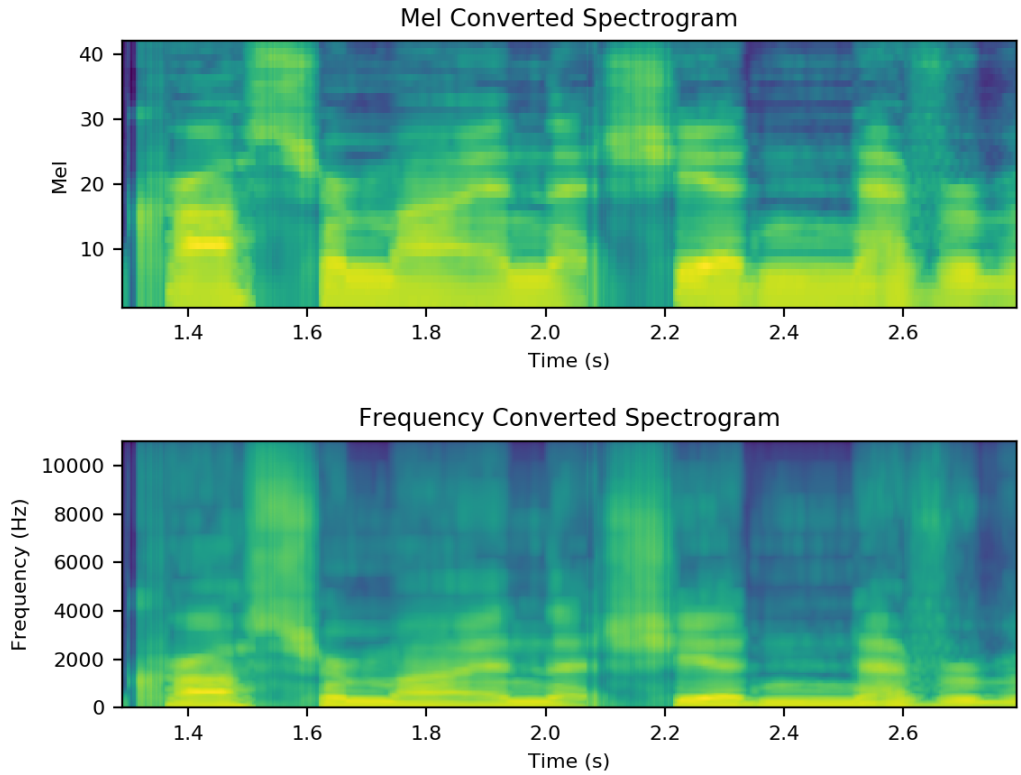


Figure 6.12: The first spectrogram has Mel filter banks applied. The second one has been reversed from Mel spectrogram

6.4 A-weighting

Power spectrum that is provided by the STRAIGHT framework and/or Mel spectrum doesn't represent the loudness perceived by human ear. Power spectrum is perceived in logarithmic fashion. Also sensitivity varies along different frequencies. Standard [10] provides standards for measurement of sound pressure, including A-, B-, C- and D- weightings. A frequency weighting can be achieved using parameterized filter[11] H_A thats defined as:

$$H_A(f) = \frac{\omega_4^2 f^4}{(f+\omega_1)^2(f+\omega_2)(f+\omega_3)(f+\omega_4)^2} \tag{6.6}$$

Coefficients for A-weighting ω_n are: $\omega_1 = 20.598997$, $\omega_2 = 107.65265$, $\omega_3 = 737.86223$, $\omega_3 = 12194.217$. Full A-weighting is finished using frequency logarithmization done by using common formula:

$$A(f) = 20\log_{10}(H_A(f)) + 2 \tag{6.7}$$

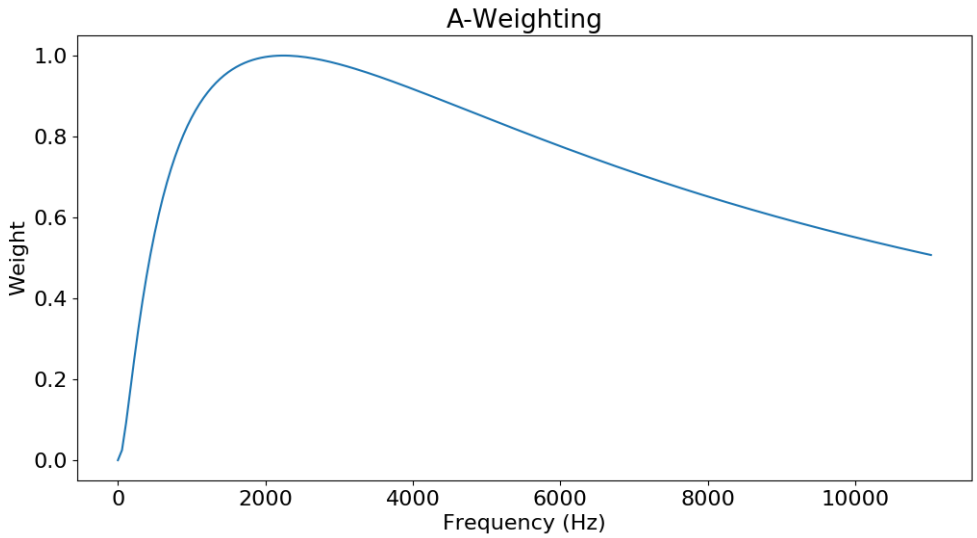


Figure 6.13: Showcase of A-Weighting weights across the frequencies

Inverse transformation of A-weighting is applied as:

$$A_R(f) = 10^{\frac{f-2}{20}}, \tag{6.8}$$

$$H_{AR}(f) = \frac{A_R(f)}{H_A(f)}, \tag{6.9}$$

where $A_R(f)$ is reversed logarithmization and $H_{AR}(f)$ is reversed A-weightings.

6.5 Silence Removal

In speech processing, the silence feature thats undesired to be processed. In the voice conversion silence is feature that won't be converted. Training usually contain 15-25% of silence. Training using silence will lead the conversion system to be performing input regression on the regular phonemes and input propagation on the silent frames, therefore compromising its parameters for the sake of unnecessary conversion. We remove the silence using recognized phonemes (see section 4.1) where silence is recognized as *pau*. Silence removal is performed on aligned spectrograms. When we remove silence in the source spectrogram we have to remove same part in the target spectrogram as well not to mismatch the alignment.

6.6 Normalization

Before we feed spectrum (that we have filtered using male banks and applied A-weighting and logarithmic transformation) into our conversion subsystem, we optimize input i.e. the prepared spectrum. Input optimization is the last step before input is fed into the conversion subsystem. Optimization will enable the conversion subsystem to fully focus on relevant information and information will be provided in expected zero-mean, unit-variance format.

6.6.1 Energy Normalization

In the spectrogram, same phonemes appear with different energy, meaning the have different mean value. The conversion subsystem would have to be trained to convert same phonemes with difference energy, therefore increasing variability of the input that could have the same transformation. We normalize energy by subtracting mean value μ of each frame $\mathbf{x}(t)$.

6.6.2 Spectrum Normalization

When we feed spectrogram into the conversion subsystem each input is receiving series of one part of the spectrum across the time. All series have different mean values and standard deviation. If we normalize spectrum across the time the conversion subsystem will be receiving more balanced input. At first we normalize mean values of the spectrum $x(t)$:

$$x_i(t) = x_i(t) - \mu_i \tag{6.10}$$

Then we normalize standard deviation of the spectrum $x(t)$:

$$x_i(t) = \frac{x_i(t)}{\sigma_i} \tag{6.11}$$

After conversion, the spectrogram has to be denormalized by inverse formula:

$$x_i(t) = x_i(t)\sigma_i + \mu_i \tag{6.12}$$

6.7 Regression Neural Network

As a spectrum conversion subsystem we use feed forward neural network that is set up for regression. Topology for the neural network is inspired by DNN-based auto-encoder for speech enhancement, dereverberation and denoising [12]. The input of the neural network is a series of source speaker frames. Output is one target speaker frame.

6.7.1 Neural Network Topology

As the input for the Neural Network (NN) we provide contextual window of the pre-processed source spectrum. The specific frame that is being converted may depend on surrounding frames and contextual window provides couple upcoming and forthcoming frames for the input. Usually, the contextual window spans 15 frames, 7 frames to the left and 7 frames to the right with one central frame that has the same index as the target frame. Therefore we have 15 frames in the input.

The output is target spectrum we have aligned to the source spectrum in previous chapter. Target spectrum is also pre-processed.

The NN has 3 densely-connected layers: input layer, hidden layer and output layer. Activation functions for the input and hidden layer are hyperbolic tangent activation function (tanh). Tanh is zero centered and output values are in range from -1 to 1. This is makes it suitable for the regression task. The output layer uses linear activation function that provides converted frame.

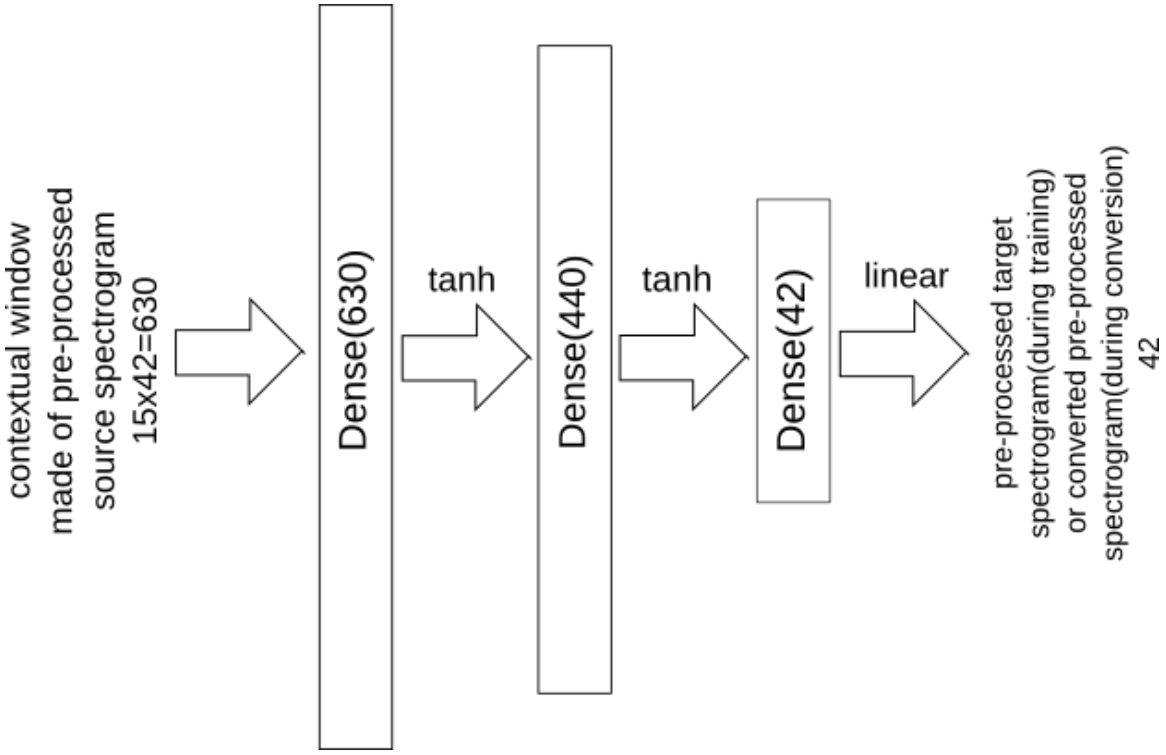


Figure 6.14: Scheme of the regression neural network

6.7.2 Neural Network Training

The NN training is performed in epochs. During one epoch all training data are passed through the NN exactly once. The whole data usually takes to much memory, therefore it is divided into batches. During one epoch batches are passed throughout the NN in iterations. After each epoch, a training algorithm based on stochastic gradient descent updates the NN weights.

Loss function

In order for stochastic gradient descent based training algorithms to converge, the loss function is required to evaluate output accuracy during training. The loss function is non-negative value and its value decreases as the accuracy of the output increases. The loss is represents difference between training value y and the predicted value $\tilde{y}$.

For the purpose of regression task we use mean square error (MSE) loss function. MSE is ideal for the task of regression because it represents the distance of the training value y and the predicted value $\tilde{y}$ by subtracting and squaring them and taking mean of all distances. Squaring the distance makes distance parabolic and the NN will converge faster.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \tilde{y}_i)^2 \tag{6.13}$$

6.7.3 Training Data

As training data we acquired training data [13] used in Voice Conversion Challenge 2018. This data-set provides data for both parallel and nonparallel training. We will use parallel data-set that provides 4 source speakers (2 male and 2 female) and 4 target speakers (2 male and 2 female), thus providing 16 conversion pairs. There are 81 sentences provided by each speaker and overall speech length is about 4 minutes.

6.7.4 Data Set Preparation

Pre-processed data from previous chapter are divided into training data and validation data with split 0.2 (80%training data, 20%validation). Training data are data used to calculate loss and update weights. Validation data serve to compute validation loss after each epoch. Validation loss indicates ability of the NN to predict unseen testing data. Divergence of the validation loss allows us to detect overfitting of the NN and adjust optimizer.

6.7.5 Adam optimizer

Adam [14] is optimizer using stochastic gradient descent optimization. The name is derived from adaptive moment estimation. It builds on two popular optimizes: AdaGrad and RMSProp.

AdaGrad optimizer stands for adaptive gradient. It introduces adaptive learning rate based on the parameters. Each parameter is updated with different learning rate. The infrequent parameters are updated with bigger learning rate. The frequent parameters are updated with smaller learning rate. This approach works great for tasks with sparse gradients.

RMSprop optimizer introduces use of the momentum during gradient descent. Momentum restricts oscillation in non-convergent direction in the steps during the gradient descent. This allows us to user greater learning rate.

The recommended settings for Adam optimizer are: $\alpha(learningrate) : 0.001, \beta_1 : 0.9, \beta_2 : 0.999$ (Exponential decay rates for the moment estimates) and $\epsilon = 1e^{-8}$.

6.7.6 Training Results

Training session is considered successfully when validation loss is converging along with training loss. If training loss diverges training is stopped. The epoch with the minimal training loss is taken with its NN wights for the conversion as it has the best ability to generalize unseen input.

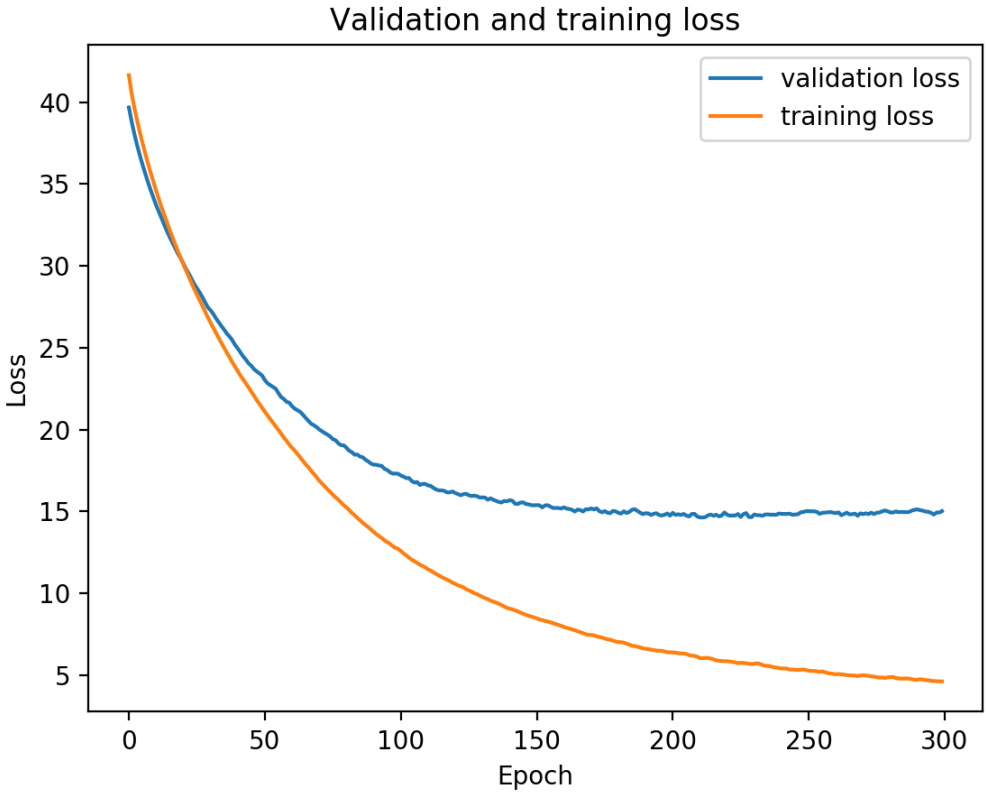


Figure 6.15: Validation and training loss during training

Chapter 7

Fundamental Frequency Conversion

First attempts to perform fundamental frequency $f_0(t)$ conversion involved conversion using the similar neural network conversion system as we use for the spectrogram conversion.

7.1 Fundamental Frequency Preprocessing

Before performing the conversion we perform preprocessing on the source fundamental frequency $f_0(t)_s$ and the target fundamental frequency $f_0(t)_t$ that is similar to the spectrogram preprocessing. The $f_0(t)_t$ is aligned to the $f_0(t)_s$ based on the shortest path and postprocessing of the shortest path that we have computed during spectrogram alignment.

STRAIGHT analysis provides a vector of hard decision (either 0 or 1) weather the analysed frame is voiced or unvoiced/silent. Similar to the silence removal in the spectrogram conversion, we remove unvoiced or silent frames from $f_0(t)_s$ and $f_0(t)_t$, since we are not interested in training or converting those data.

7.2 Problems With Conversion using NN

The results were compromised by the training. The validation loss couldn't reach satisfying levels. The conversion tests provided output that couldn't be recognized as $f_0(t)$, due to having too much noise and appearing to have random variance. The reason for this behavior might inconsistencies of the components variation. In another words the variance of the source $f_0(t)$ has no correlation with the variance of the target $f_0(t)$.

7.3 Simplified Fundamental Frequency Conversion

If the $f_0(t)$ can't be fully converted we can at least we can observe mean and standard deviation in both source and target $f_0(t)$. Then we can denormalize the $f_0(t)$ that we are converting using the source $f_0(t)$ mean μ_s and standard deviation σ_s and then denormalize using target mean μ_t and standard deviation σ_t:

$$f_0(t)_c = \frac{f_0(t)_c - \mu_s}{\sigma_s}\sigma_t + \mu_t \tag{7.1}$$

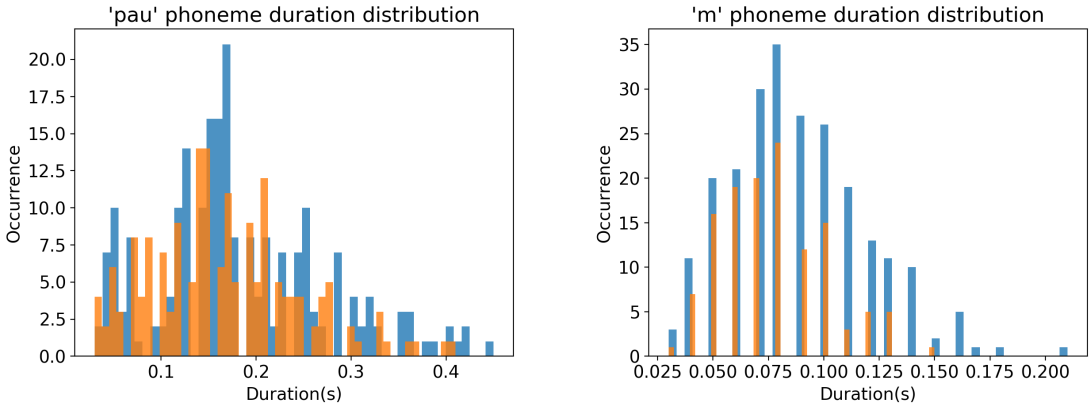
Chapter 8

Phoneme Length Change

Phonemes pronounced by the speakers have different duration and variance of the duration. The slower the speaker pronounces the speech the longer the phonemes have to last and vice versa. Slower speakers also makes longer pauses. Different speaker pronounce at different speed and we want to reflect that in the voice conversion.

8.1 Phoneme Duration and Variance

Using the phoneme recognizer (see section 4.1) we can get phonemes from all training data and analyze them. Some of the most common phoneme are *pau* and *m*. If we measure mean duration and standard deviation of the duration of both source and target speaker we see the difference:



(a) The mean duration of pause is 0.1816s and standard deviation is 0.0884

(b) The mean duration of pause is 0.0771s and standard deviation is 0.0237s

The mean duration (0.1816s) and standard deviation (0.0884s) of the source speaker pause is greater than that of the target speaker (0.1656s, 0.0777s). The mean duration (0.0883s) and standard deviation (0.0311s) of the source speaker *m* is also greater than that of the target speaker (0.0771s, 0.0237s).

8.2 Changing Phoneme Length

In changing phoneme length, we utilize the features of the STRAIGHT framework. After analysis, STRAIGHT provides array of the temporal positions matching the size of the other parameters (spectrogram and aperiodicity). The temporal positions mark the time of the event. Changing the temporal position will change the time of the event in the synthesis e.g. multiplying the temporal position by two will make synthesized speech twice as long.

8.2.1 Changing the Temporal Positions

Knowing the means and standard deviations of phoneme durations we have gained by analyzing phonemes of all the source and target training data, we can change the duration of the phonemes during conversion. The temporal positions will be changed locally by normalizing the duration of the phoneme being converted t_c by the source speaker mean μ_s and standard deviation σ_s and denormalizing it by the target speaker mean μ_t and standard deviation σ_t:

$$t_c = \frac{t_c - \mu_s}{\sigma_s}\sigma_t + \mu_t \tag{8.1}$$

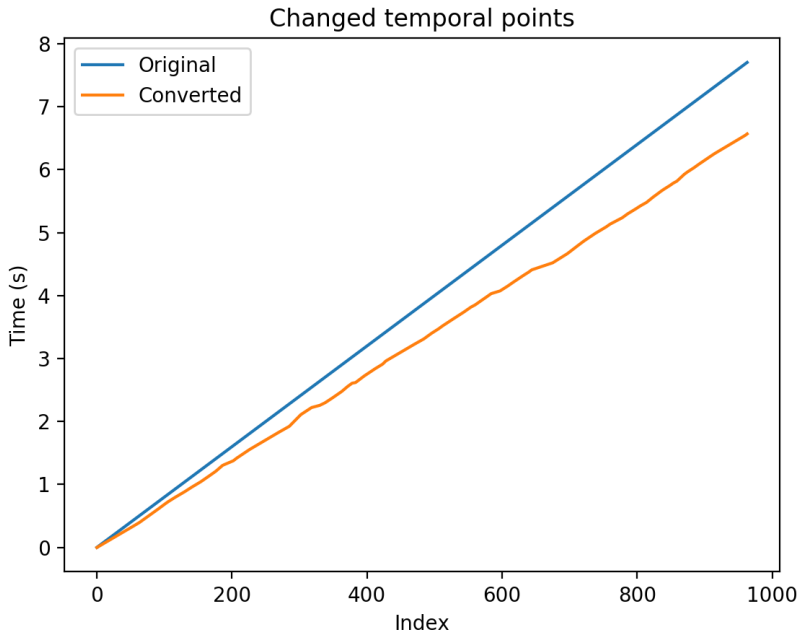


Figure 8.2: Converted temporal points show that converted speech is shorter (6.569s) than the original source speaker speech (7.704s)

Chapter 9

Implementation

Out conversion system consists of multiple systems, mainly feature extraction and synthesis system (STRAIGHT) and training and conversion system. Due to this inconvenience there are multiple programing languages used. Different systems also have to be individually executed by user.

9.1 STRAIGHT

STRAIGHT framework was developed in Matlab code. We have successfully used STRAIGHT source codes without modifications using GNU-Octave. It was required to use signal package to execute all functions used by STRAIGHT. STRAIGHT is very time consuming framework for repeatedly analyzing training data, therefore we save intermediate results on the disk in Matlab format, version 6. The voice conversion system was developed on Linux operating system however used programing languages are used and the voice conversion system should be able to run on different platforms.

9.2 PhnRec

Phoneme recognizer was provided both as a C++ source code and Windows executable. Source code compilation was unsuccessful, therefore Windows executable was used in Linux environment using Wine API translator.

9.3 Conversion Techniques

DTW function was provided by Jan Černocký [1], written for in Matlab code, successfully executed in GNU-Octave. DTW is also time consuming, therefore we save intermediate results on the disk. The finishing steps of DTW (spectrogram smoothing) are implemented in Python3.6. The rest of the spectrogram processing pipeline is also implemented in Python3.6. Very often used are Python libraries for scientific computing: Numpy, SciPy and Matplotlib.

[1] `http://www.fit.vutbr.cz/study/courses/ZRE/public/labs/05_dtw_hmm/`

9.4 Neural Network Implementation

The Neural Network was implemented in Python3.6 using popular library Keras. Keras is high-level wrapper for lower-level computation libraries. The low-level computational library of choice for the task of machine learning is TensorFlow. TensorFlow provides options to use GPU as a computation unit, however due to the unavailability of hardware that's up to date means we only utilized CPU.

Chapter 10

Experiments and Results

The testing data were also obtained from VCC 2016 database [13]. Testing data contain about 1.5 minutes of the source speaker speech.

Lots of effort was put into finding optimal setup for the training system. Also, different techniques that the neural network were attempted, mainly linear regression using transformation matrix $\mathbf{X}$ that is obtained as least-squares solution to a linear matrix equation $\mathbf{SX} = \mathbf{T}$ that is converting source spectrogram $\mathbf{S}$ to target spectral $\mathbf{T}$. This approach turned out not to adapt well to the training data, even such a small data-set as we use.

Thanks to the simplicity of fundamental frequency conversion and phoneme length change, those techniques achieve anticipated results as it was demonstrated in their chapters (7, 8).

10.1 Spectrogram Conversion Results

The main struggle with the neural network was to get good results with training and especially validation loss. We were unable to decrease validation loss below certain point. This results in inaccurate conversion. While overall frequential characteristic of the converted spectrogram reminds that of the target spectrogram, it looses lots of detail. The most important detail that is lost are formants. Formants are result of the acoustic resonance during voice production. Such spectrogram when synthesized sounds on average as the target speaker should, the speech sounds dull and sometimes hardly understandable. Converted samples in .wav format are available on attached DVD in folder */samples/converted*. Those samples were converted from evaluation samples in */samples/eval*. There are samples of the source and target speaker to get the sense of their voices in folders */samples/source* and */samples/target*.

Chapter 11

Conclusions and Future Work

11.1 Spectrogram Conversion Conclusion

Spectrogram conversion doesn't achieve satisfactory results. There are multiple solutions for current voice conversion system to consider to improve spectrogram conversion:

- Acquiring larger training data-set.
- Redefining loss function to represent formants better in the loss
- Adding new parameter to the neural network input to decorrelate parts of source spectrogram that look same, might to be aligned with parts of the target spectrogram that look different
- Use of different representation of the spectrum such as Mel cepstral coefficients

11.2 Future Work

Based on experience gained during this thesis, the main areas of voice conversion that require attention are matching of the training data and using suitable spectrogram conversion techniques.

11.2.1 Nonparallel Training

Nonparallel training allows the training system to use training speech that doesn't contain same sentences from the speakers. Nonparallel training would unlock possibility to use any training speech of the speakers, therefore being able to use more training data and optimizing the data cross-referencing. In parallel training, we are restricted by the parallel data that has to be aligned. Aligned data might contain little defects that spoil DTW.

11.2.2 Fundamental Frequency Discussion

We were unable to collect meaningful information for the fundamental frequency conversion by analyzing fundamental frequency only. Therefore we have to look at other features affecting it such as entire spectral envelope. Another features to look at are linguistic and sentence context. Speakers usually modulate fundamental frequency in such way as there are frequency spikes throughout middle of the sentences and frequency decreases at the end of the sentence.

Bibliography

[1] Tomoki Toda, Ling-Hui Chen, Daisuke Saito, Fernando Villavicencio, Mirjam Wester, and Zhizheng Wu andJunichi Yamagishi. The voice conversion challenge 2016. *Information Technology Center, Nagoya University, Japan University of Science and Technology of China, China The University of Tokyo, Japan National Institute of Informatics, Japan The Centre for Speech Technology Research, The University of Edinburgh, UK*, pages 1632–1636, Sep 2016. `https://www.isca-speech.org/archive/Interspeech_2016/pdfs/1066.PDF` [Accessed 2019-5-11].

[2] J. Yamagishi M. Wester, Z. Wu. Multidimensional scaling of systems in the voice conversion challenge 2016. *The Centre for Speech Technology Research, The University of Edinburgh, UK National Institute of Informatics, Japan*, pages 40–45, 2016. `http://www.vc-challenge.org/vcc2016/papers/SSW9_VCC2016_Results.pdf` [Accessed 2019-5-11].

[3] Y. Stylianov. Voice transformation. In Benesty Jacob, M. M. Sondhi, and Huang Yiteng (Eds.), editors, *Springer Handbook of Speech Processing*, chapter 24, pages 489–503. Springer, New York City, 2008. `http://habla.dc.uba.ar/gravano/ith-2014/presentaciones/Stylianou_2008.pdf`.

[4] Hideki Kawahara, Toru Takahashi, Masanori Morise, and Hideki Banno. Development of exploratory research tools based on tandem-straight. *APSIPA ASC 2009 - Asia-Pacific Signal and Information Processing Association 2009 Annual Summit and Conference*, 01 2009. `https://www.researchgate.net/publication/39999861_Development_of_exploratory_research_tools_based_on_TANDEM-STRAIGHT`.

[5] Prof. Don Colton. Ipa, worldbet, and ogibet english broad phonetic labels. *Center for Spoken Language Understanding - Oregon Graduate Institute of Science and Technology*, 2004. `http://byuh.doncolton.com/courses/cs441/9504.refbet.pdf` [Accessed 2019.5.11].

[6] SCHWARZ Petr, MATĚJKA Pavel, and ČERNOCKÝ Jan. Towards lower error rates in phoneme recognition. *Lecture Notes in Computer Science*, 2004(3206):8, 2004. `http://www.fit.vutbr.cz/research/view_pub.php?id=7647`.

[7] MATĚJKA Pavel SCHWARZ Petr and ČERNOCKÝ Jan. Hierarchical structures of neural networks for phoneme recognition. In *Proceedings of ICASSP 2006*, pages 325–328, 2006. `http://www.fit.vutbr.cz/research/view_pub.php?id=8134` [Accessed 2019-5-11].

[8] Jan Černocký. Rozpoznávání řeči – úvod a dtw. *ÚPGM FIT VUT Brno*, Jan 2018. http://www.fit.vutbr.cz/study/courses/ZRE/public/pred/08_rozp_dtw/08_rozp_dtw.pdf [Accessed on 1/13/2019].

[9] Jan Černocký. Předzpracování řeči, tvorba řeči, cepstrum. *ÚPGM FIT VUT Brno*, Jan 2018. http://www.fit.vutbr.cz/study/courses/ZRE/public/pred/03_prepro_model_ceps/03_prepro_model_ceps.pdf [Accessed 2019-5-11].

[10] Electroacoustics – sound level meters – part 1: Specification. Standard, International Electrotechnical Commission, Dublin, Sep 2013.

[11] Andrew N. Rimell, Neil J. Mansfield, and Gurmail S. Paddan. Design of digital filters for frequency weightings (a and c) required for risk assessments of workers exposed to noise. *Ind Health*, 53(1), Jan 2015. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4331191.

[12] Ondřej Novotný, Oldřich Plchot, Ondřej Glembek, Jan Černocký, and Lukáš Burget. Analysis of dnn speech signal enhancement for robust speaker recognition. *Brno University of Technology, Speech@FIT and IT4I Center of Excellence, Božeťechova 2, 612 66 Brno, Czech Republic*, 11 2018. https://arxiv.org/pdf/1811.07629v1.pdf.

[13] Lorenzo-Trueba Jaime, Yamagishi Junichi, Toda Tomoki, Saito Daisuke, Villavicencio Fernando, Kinnunen Tomi, and Ling Zhenhua. The voice conversion challenge 2018: database and results. *ÚPGM FIT VUT Brno*, 2018. https://datashare.is.ed.ac.uk/handle/10283/3061 [Accessed 2019-5-11].

[14] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conferenc Track Proceedings*, 2015. http://arxiv.org/abs/1412.6980 [Accessed 2019-5-11].

Chapter 12

Appendices

The attached DVD contains following content:

- PDF version of this thesis
- Latex sources for this thesis
- source-codes for the voice conversion system
- voice samples of: source speaker, target speaker, test/evaluation speaker, converted voice
- README.md that contains detailed description of the DVD content and manual for the voice conversion system

12.1 Manual

This section provides manual for using the voice conversion system that is stored in `src` folder.

12.1.1 Creation of Intermediate Results of Speech Analysis and DTW

STRAIGHT is quite time consuming framework. Folders `train_source`, `train_target` and `eval` store training and evaluation speech in `.wav` format, sampled at 22050Hz with 16-bit signed integer PCM encoding. The results of speech analysis of those files are stored in folders `mat_source`, `mat_target` and `mat_eval`. Analysis is done by executing `save_features.m` Matlab script. If executed in GNU-Octave, then signal package is required. Note that analysis takes 1-2 hours and intermediate results take about 2.5GB on the hard-drive. Shortest path calculation is executed using `save_dtw.m` and takes about 1 hour.

12.1.2 Phoneme Recognition

Phoneme recognition on Linux uses Wine program to run PhnRec `executable` and scripts in `scripts` are written for Linux `bash`. In case of not being able to run these steps, phonemes recognition is already pre-computed in folders `rec_*`. Phoneme recognition is done in two steps:

1. Converting .wav files to .raw files used by `PhnRec` using `make_raw.sh`. The .raw files are stored in `raw_*` folders.

2. Recognition itself done by `save_phonemes.sh`. The `.rec` are stored in `rec_*` folders.

12.1.3 Training and Conversion

Training is implemented using python scripts. Required packages are:

- NumPy
- matplotlib
- Keras
- SciPy
- TensorFlow
- Pillow

Training of the neural network and fundamental frequency is done by executing two scripts: `train_nn.py` and `train_f0.py`.

Results of training are stored in `models` and `normalization` folders.

After that conversion can be executed by `convert.py`. Intermediate results of conversion are stored in `mat_converted` folder. Intermediate results are synthesized by executed Matlab script `convert_mat.m`. After that results of the voice conversion are available in `converted` folder as `.wav` files sampled at 22050Hz with 16-bit signed integer PCM encoding.