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

ROOM IMPULSE RESPONSE ESTIMATION FROM SPEECH SIGNAL

ODHAD IMPULSNÍ ODEZVY MÍSTNOSTI Z ŘEČOVÉHO SIGNÁLU

BACHELOR’S THESIS
BAKALÁŘSKÁ PRÁCE

AUTHOR ADAM GREGOR
AUTOR PRÁCE

SUPERVISOR JAN ČERNOCKÝ,
VEDOUCÍ PRÁCE

BRNO 2020

Brno University of Technology
Faculty of Information Technology

Department of Computer Graphics and Multimedia (DCGM) — Academic year 2019/2020

Bachelor's Thesis Specification

21947

Student:	**Gregor Adam**
Programme:	Information Technology
Title:	**Room Impulse Response Estimation from Speech Signal**
Category:	Speech and Natural Language Processing

Assignment:

1. Get acquainted with classical techniques for room impulse response (RIR) estimation.
2. Get acquainted with BUT ReverbDB data-set and replicate the estimation of RIRs from measured data.
3. Suggest an estimation of RIR directly from speech signal (for example based on isolated pulse-like sounds, or machine learning).
4. Implement the proposed technique(s), suggest, how to numerically evaluate the quality of resulting RIRs, compare the results.
5. For selected signals, verify the results by informal listening tests.
6. Create a short video and/or poster on presenting your work.

Recommended literature:

- SZŐKE Igor, SKÁCEL Miroslav, MOŠNER Ladislav, PALIESEK Jakub and ČERNOCKÝ Jan. Building and Evaluation of a Real Room Impulse Response Dataset. IEEE Journal of Selected Topics in Signal Processing, vol. 13, no. 4, pp. 863-876. ISSN 1932-4553.
- then based on supervisor's recommendation

Requirements for the first semester:

- Items 1-3, elaboration of item 4.

Detailed formal requirements can be found at https://www.fit.vut.cz/study/theses/

Supervisor:	**Černocký Jan, doc. Dr. Ing.**
Head of Department:	Černocký Jan, doc. Dr. Ing.
Beginning of work:	November 1, 2019
Submission deadline:	May 14, 2020
Approval date:	April 15, 2020

Abstract

When travelling in a room, any sound is distorted by a room impulse response (RIR). Determining RIR has always been an important task in acoustics, but nowadays, it is even more important, as RIR can be used to augment data for training automatic speech recognition (ASR) systems. Classically, a RIR is estimated from a pair of clean and reverberated sound signals. This is however not practical for real scenarios (such as personal assistants, smart homes, etc.), as the clean signal is not available. The aim of the bachelor thesis is to investigate ”blind” RIR estimation only from a reverberated speech signal. We have used the BUT ReverbDB data set and first, re-implemented techniques for classical clean-reverberated signals estimation of RIRs. Then, we investigated two techniques for RIR estimation only from a reverberated signal. The first technique uses reverberated impulse-like phonemes in speech which are expected to resemble RIR. Averaging and deconvolution of these phonemes were tested to improve the quality and robustness of the estimation. The second technique makes use of a regression neural networks trained to produce the RIR from a speech input. Although none of the techniques reaches the quality of classical measurement, the estimated RIRs have the potential to help in augmenting data for ASR system training.

Abstrakt

Jakýkoliv zvuk šířící se místností je zkreslen impulsní odezvou této místnosti. Měření těchto impulsních odezev bylo vždy důležitou úlohou akustiky, která v dnešní době ještě nabyla na důležitosti, díky možnosti požití impulsních odezev při augmentaci dat pro účely trénování automatických rozpoznávačů řeči. Standardně je impulsní odezva místnosti měřena za pomoci čisté a zkreslené formy zvukového signálu. To je však v praxi nepraktické (například u domácích asistentů či chytrých domů), neboť zde je k dispozici jen zkreslený signál. Tato bakalářská práce se zabývá odhadem impulsní odezvy „naslepo“, pouze pomocí zkresleného řečového signálu. Nejdříve jsme za použití datasetu BUT ReverbDB re-implementovali standardní techniky pro měření impulsní odezvy z čistého/zkresleného signálu. Poté jsme testovali dvě techniky odhadující impulsní odezvu místnosti pouze ze zkreslené řeči. První technika k tomu používá impulsní fonémy ve zkreslené řeči, u kterých se předpokládá, že se podobají impulsním odezvám místností. Bylo testováno průměrování a dekonvoluce těchto fonémů za účelem zvýšení kvality a robustnosti odhadu. Druhá technika využívá regresní neuronové sítě generující impulsní odezvy místností z řeči na vstupu. Ačkoliv žádná z navrhovaných technik nedosahuje odhadů na úrovni standardních měření, mají tyto odhady potenciál při augmentaci dat pro trénování automatických rozpoznávačů řeči.

Keywords

Room Impulse Response Estimation, Speech signal

Klíčová slova

Odhad Impulsní odezvy místnosti, Řečový signál

Reference

GREGOR, Adam. *Room Impulse Response Estimation from Speech Signal.* Brno, 2020. Bachelor’s thesis. Brno University of Technology, Faculty of Information Technology. Supervisor Jan Černocký,

Room Impulse Response Estimation from Speech Signal

Declaration

I declare that this thesis is my own work supervised by Jan Černocký. All sources have been acknowledged by references.

. . . .
Adam Gregor
May 27, 2020

Acknowledgements

I would like to thank my supervisor Jan Černocký for his guidance, always optimistic attitude and all the support he had provided me with. I would also like to thank Ondřej Glembek for useful tips when dealing with neural networks and Igor Szőke for helping me with ReverbDB results replication. And of course I want to thank to my family, BL and all the people who encouraged me.

Contents

Chapter 1

Introduction

Automatic speech recognition (ASR) is a science discipline focusing on transcribing recorded human spoken language into a text. Just as most of computer science fields, it also has made a huge progression over the last decades. What firstly was a transcription of several carefully pronounced English words in the close surrounding of a microphone became a technique able to correctly transcribe whole sentences spoken in various languages that were recorded in rather noisy areas. Just as automatic speech recognition accuracy has dramatically increased so has its role in our lives as it is a part of home assistants or in-car systems which are used by millions of people every day.

However, this usage of automatic speech recognition in the real world introduced new problems that are not present in small artificial tasks. Often, the location of a speaker (especially in a room, such as an office, living room etc.) is far away from the microphone, causing increased noise and reverberation of the recorded speech, which causes ASR's accuracy to drop. In such cases, a possible solution is training the speech recognizer on a clean speech signal augmented by a room impulse response of the room it is located in. Room impulse response describes, how an impulsive sound is reverberated between two points in the room. Such augmented speech then sounds as if it was recorded in the room, where the impulse response was measured. An ASR trained on these augmented data is expected to perform better, in the particular room, than a system trained on clean speech data only.

Most of the methods for room impulse response measurement are based on emission of a known signal and its comparison with its recorded form. Another used approach is estimation of room impulse response from the room's parameters. As the emission of the signal takes a huge amount of time (or can not be done at all) and parameters of the room are usually unknown, those methods are not suitable for portable devices using ASR. For such devices, a method estimating room impulse **only from a speech signal** would be desirable, as it is the most common type of signal these devices come into contact with.

The goals of this thesis are to overview commonly used methods for measurement of room impulse response (chapter 2), implement metrics for comparison of impulse responses (chapter 3), introduce ReverbDB database and replicate some of its results (chapter 4) and design and test methods estimating room impulse response from the speech signal only. This core of this bachelor thesis is dealt with in chapter 5, where impulse responses are estimated analytically and chapter 6 where impulse responses are estimated using neural networks.

Chapter 2

Reverberation and Standard Techniques for Room Impulse Response Measurement

When trying to characterize acoustics of a location, we want to describe, how a sound signal is changed between its emission and recording. To characterize this change, a room impulse response (RIR) is used to describe how a short, impulsive sound (in theory a Dirac pulse) is reverberated between two points of the room.

To capture this response, several methods can be used which can be divided into two categories. Methods of the first category measure real sound signals in the examined environment, while methods of the second one estimate the room impulse response using a simulation of sound propagation in a model based on the examined environment.

2.1 Reverberation

When recording a sound signal in a room, the recorded signal is not only the emitted one, delayed by a time which corresponds to distance between the speaker and the recording device, but, as can be seen in Figure 2.1, also contains noise and reflections of the original sound.

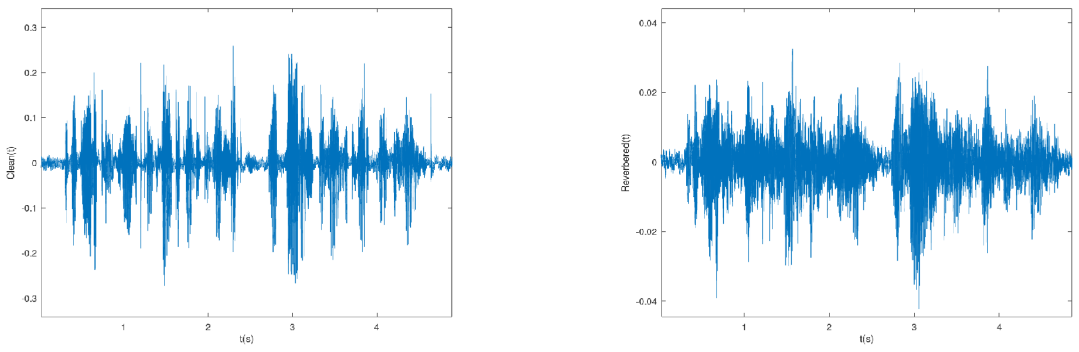


Figure 2.1: Comparison of an original form of signal (left) with its reverberated form (right). Notice that impulses are ”smeared” within reverberation of previous part of signal and noise of room.

The relationship between reverberated signal $y(t)$ and its original form $x(t)$ can be expressed as

$$y(t) = x(t) * h(t) + n(t) \tag{2.1}$$

where $h(t)$ denotes room impulse response, $n(t)$ additive noise of a room and $*$ convolution operator.

Room impulse response (shown in Figure 2.2) describes response of a room to an impulsive sound with a flat frequency spectrum. It contains the initial impulse together with its delayed, attenuated reflections. The initial impulse at the beginning of the RIR corresponds to the shortest propagation path of the impulsive signal and usually possesses the highest amount of energy of all received reflections. Reflections in the latter parts of RIR are created by different propagation paths with greater length than the one of the initial signal (which is the reason for their increased delay). As time grows, so does the number of reflections from different surfaces before reaching the microphone. As every surface absorbs a certain amount of the signal's energy, based on its material (compare a rugged and a plain wall), later reflections are usually less distinct than the early ones.

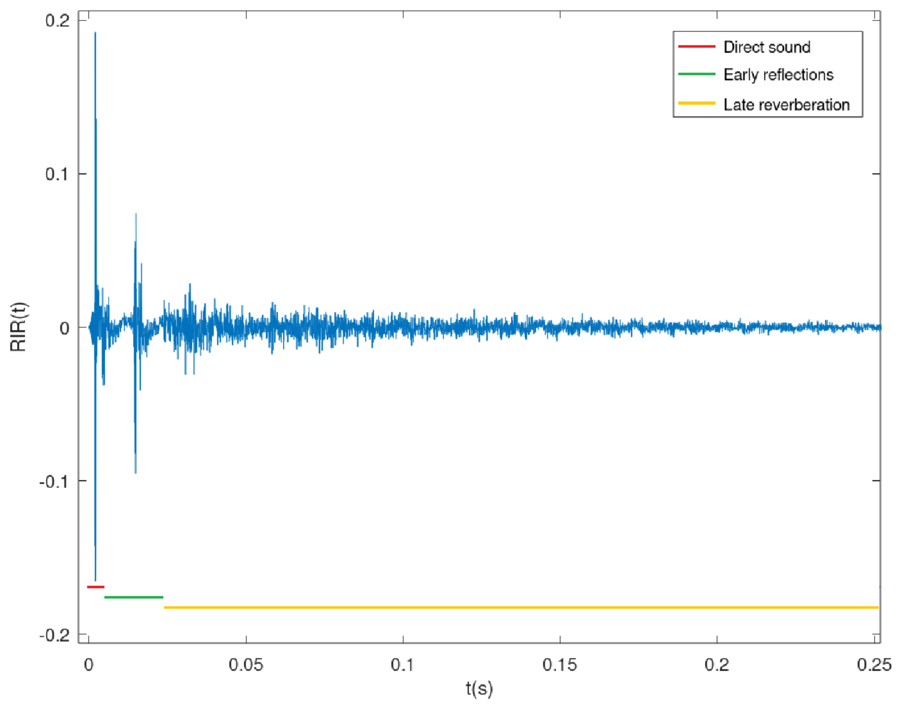


Figure 2.2: A look onto the impulse response with lines, representing parts of the response as described by Yoshioka [18].

Note that every two points of the room have at least slightly different RIRs, as propagation paths of a sound between every two points are different. The propagation and reflection of a sound signal is shown in Figure 2.3.

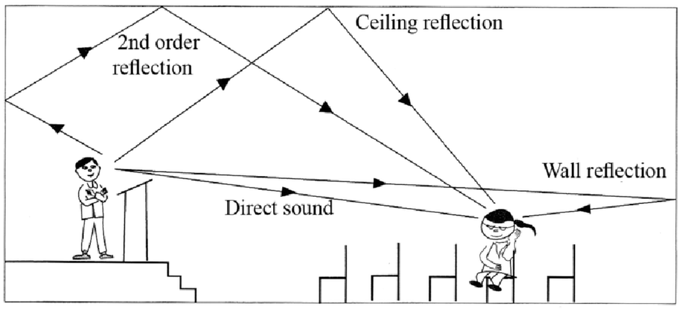


Figure 2.3: Visualization of how sound signal is distributed. Sound waves may be reflected several times before they reach the receiver. Reprinted from Svedström [16].

Reverberated sound (including RIR) can be, according to Yoshioka [18], divided into three parts:

- **Direct sound** – This part of the signal is generated by the shortest propagation path and therefore has the lowest delay.
- **Early reflections** – This part strongly depends on speaker and microphone location and is located within 50 ms after the Direct sound.
- **Late reverberation** – The last part of the reverberated sound is composed of reflections captured 50 ms after the Direct sound and is independent on position of the speaker and the microphone.

Energy ratio of the direct sound and early reflections to the late reverberation has a high influence on performance of an ASR system. The stronger the reverberation is, the higher word error rate can be expected from the recognizer. This ratio mainly depends on the distance between the speaker and the microphone.

In practice, $T60$ is used to measure the precise length of reverberation. It is defined as a time it takes for a sound pressure to decrease by 60 dB after the original signal is stopped:

$$L = 10 \log_{10} \left(\frac{p_2}{p_1} \right), \tag{2.2}$$

where p_2 and p_1 are powers of recorded and original signals. Converted to the linear domain, 60 dB means that the signal has to become million times weaker. $T60$ may be expressed by a single number if the original signal is wideband or as a set of numbers, each standing for different frequency range, when several signals are used. According to NTI-Audio [8], when $T60$ is difficult to measure and the signal decay is supposed to be linear, measuring drop of 30 dB and multiplying the time by 2 or measuring drop of 20 dB and multiplying the time by 3 is sufficient. These methods are then called $T30$ and $T20$.

2.2 Maximum Length Sequence Method

Maximum Length Sequence (MLS) method, as proposed by Schroeder [13], is a RIR measurement method, based on periodical emission of a pseudo-random noise-like signal. The most commonly used noise, the one discussed here, is a binary one, containing two values. Example of such signal can be seen in Figure 2.4. In MLS, the number of samples in one period of the noise-like signal is $L = 2^m - 1$, where m denotes the order of the MLS signal.

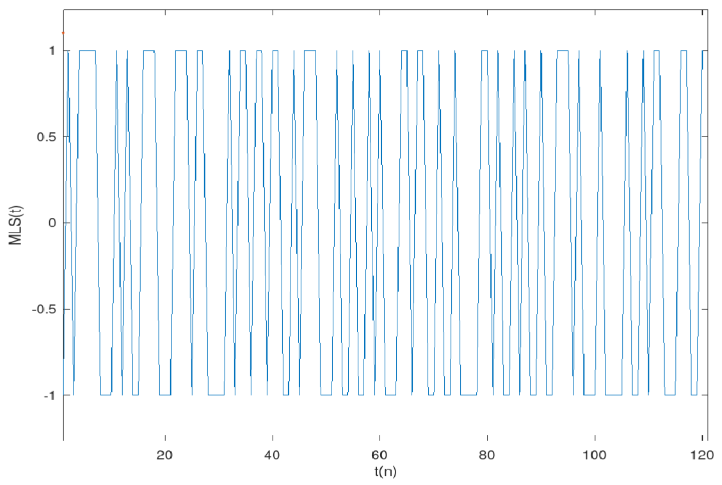


Figure 2.4: Zoom onto an MLS signal of order $m = 18$, containing total of 262143 samples.

The noise-like signal can be generated using m-stage shift registers with arbitrarily set of initial values, except all units set to one (considering values of the binary signal to be ± 1), which would generate only values of one. Digits of output signal are, for each sample, retrieved from the last register while the first register is set to value which equals to values of the last and last-but-one registers after the XOR operation ($\oplus$), as shown in Figure 2.5.

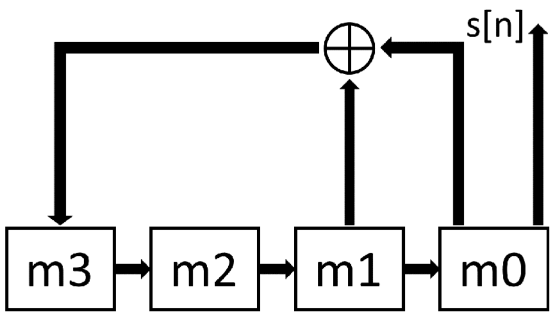


Figure 2.5: Scheme for generating noise-like signal using 4-stage shift register. $\oplus$ denotes XOR operation, $s[n]$ the noise-like signal.

After the output signal reaches length L, registers are set to their initial value and any further computation results in repeating of the original output signal of length L. The signal generation can be mathematically expressed as

$$\begin{aligned} &\sum_{n=0}^{m-1} s[n] \neq m, && \text{for } 0 \leq n \leq m-1, \\ &s[n] = s[n-m] \oplus s[n-m+1], && \text{for } m \leq n \leq 2^m - 1, \end{aligned} \tag{2.3}$$

where m denotes the order and $s[n]$ the noise-like binary signal.

This noise-like signal is then emitted from one point of a room and recorded in another. As the signal is reproduced periodically, recorded periods of the signal can be averaged to partly eliminate the noise of the measured room. The noise does not have to be only that of a silent room, as Schroeder [13] stated: "(MLS) looks promising for making measurements in halls during actual performances!". Experiments performed by Stan [15] did show that MLS is the best method for RIR measurement in occupied rooms.

After the recording is done, the averaged signal is circularly cross-correlated with the initial sound to obtain the impulse response of the room. This act of retrieving the RIR from the reverberated signal is often called deconvolution. According to Hamici [5], circular cross-correlation is based on a linear cross-correlation given by:

$$l_{xy}[n] = \sum_{k=0}^{M+N-2} x[n]\ y[n-k] \tag{2.4}$$

where $x[n]$ and $y[n]$ are compared signals and M and N are numbers of samples of these signals.

Circular cross-correlation then has its signals circular, which means that indices out of range are set back to the valid samples as if modulo operator was used:

$$c_{xy}[n] = \sum_{k=0}^{M-1} x[n]\ y[(n+k) \bmod M]. \tag{2.5}$$

Circular cross-correlation requires signals to be of the same length. Therefore, before circular cross-correlating signals of different length, length alignment has to be performed. The output of the circular cross-correlation has the same length as compared signals.

The pseudo-random noise-like signal is used as in its Fourier transformation, frequency components (except the DC one) have the same magnitude and therefore, it has the same power spectrum as a single impulse. The system (room) hence responds evenly to every frequency.

MLS method assumes the room to be a Linear, Time-Invariant (LTI) system. Non-linearities of the system are present as distortion peaks in the impulse response and have no connection to the real acoustics of the measured room. These distortion peaks can be reduced by thorough calibration of speaker/microphone devices.

Further suppression of distortion peaks can be achieved by using a signal of length $2L$, derived from MLS signal as:

$$x[n] = \begin{cases} s[n], \text{ if n is even,} & \text{for } 0 \leq n < 2L \\ -s[n], \text{ if n is odd,} & \text{for } 0 \leq n < 2L \end{cases} \tag{2.6}$$

where $s[n]$ denotes the original, noise-like signal.

Except for emitting and recording a different signal, the procedure stays the same. With this change, the method is called Inverse Repeat Sequence (IRS).

This method also expects input/output sampling clock synchronization, however, clock asynchronicity can be compensated, as proposed by Szőke [17], by applying cross-correlation on the first and last recorded period of recorded signal and using the time shift to resample recorded signal to match the original one.

2.3 Time Stretched Pulse Method

Time Stretched Pulse (TSP) method, as described in 1981 by Aoshima [1], is based on an impulse signal with a flat power spectrum for RIR measurement. This signal is expanded before emission and compressed after reception. Unlike any other method mentioned in this thesis, TSP signal is created primarily in the frequency domain, from which is then transferred into the time domain.

The basic impulse is generated in the frequency domain. It's spectrum is defined as:

$$\begin{aligned}
X[k] &= 999\ \exp\left(-\left[\frac{k-900}{800}\right]^{12}\right) && \text{for } 0 \leq k \leq 2047 \\
X[k] &= X[4097-k] && \text{for } 2049 \leq k \leq 4095 \\
X[2048] &= 0, && \qquad (2.7)
\end{aligned}$$

where exp() denotes exponential function and $X[k]$ are coefficients of Discrete Fourier Transform (DFT).

As can be seen in Figure 2.6, this complex function is conjugate symmetric which means that its Inverse DFT produces a real signal. This signal is also shown in Figure 2.6. As the transformed complex signal had its DC component located at its centre, high magnitudes are located near its leftmost and rightmost edges.

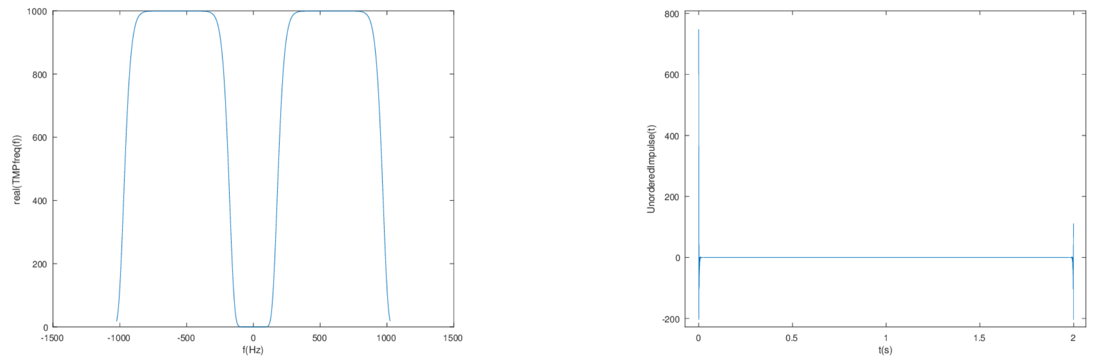


Figure 2.6: In the left panel is a real part of the complex spectrum given by Equation 2.7. Imaginary part of this function is zero. In the right panel is the signal after application of inverse DFT in which amplitudes are located on left and right edges.

When the resulting signal is rearranged chronologically (using Octave or MATLAB command `fftshift`, for example), the signal takes a form of an impulse (see Figure 2.7) which can be further used for the purpose of obtaining impulse response.

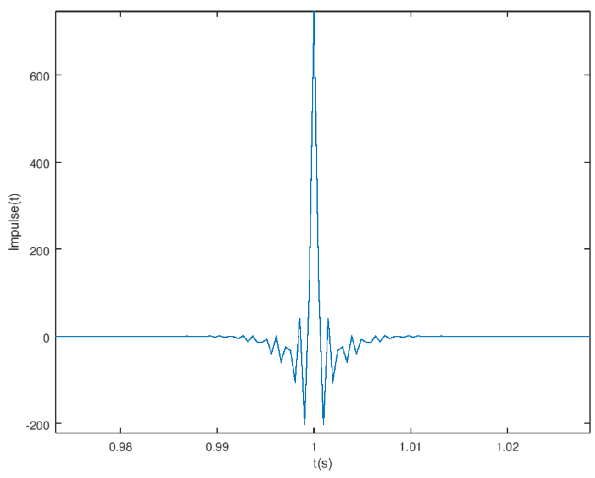


Figure 2.7: Zoom onto the signal from Fig. 2.6 chronologically rearranged in the time domain. Peaks which were located on left and right sides now meet in the middle.

The basic impulse response measurement with such a pulse signal is a very straightforward task since the emitted signal is an impulse itself and hence the recorded signal would be the room impulse response. This approach, unfortunately, suffers from a low signal-to-noise ratio (SNR) because of the low amount of energy emitted[1].

For this reason, the signal from Equation 2.7 is stretched in time by a filter with frequency response proposed by Aoshima [1] as

$$H[k] = \exp\left[j\left(\frac{12k^2}{10000}\right)\right]. \tag{2.8}$$

This results in an increase of signal's energy (compare Figures 2.7 and 2.8) and thereby in improvement of SNR without increasing magnitude of the input signal, which might cause distortion in the measured impulse response. After the reverberated time-stretched pulse is recorded in the measured environment, the impulse response can be obtained by inverse-filtering it. In the case of filter given in Equation 2.8, the inverse-filter frequency response is:

$$H^{-1}[k] = \exp\left[-j\left(\frac{12k^2}{10000}\right)\right]. \tag{2.9}$$

[1]We have encountered the same problem when estimating the RIR from "T" sounds in speech, see chapter 5.

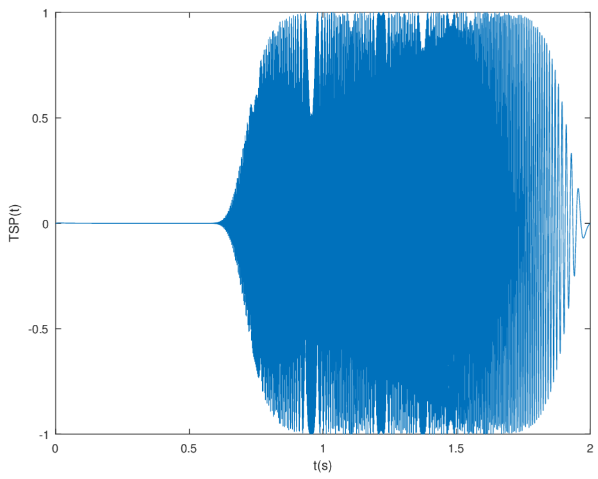


Figure 2.8: Time Stretched Impulse created by stretching the simple impulse signal in time.

Filtering of the impulse signal can be substituted with creation of a new signal. A spectrum of signal that would match the signal created by Equation 2.7 filtered with the filter from Equation 2.8 is:

$$\begin{aligned}
&X[k] = 999\ \exp\left(-\left[\frac{k-900}{800}\right]^{12}\right)\cos\left(\frac{12k^2}{10000}\right) && \text{for } 0 \le k \le 2047\\
&X[k] = X[4097-k] && \text{for } 2049 \le n \le 4095\\
&X[2048] = 0\\
\\
&Y[k] = 999\ \exp\left(-\left[\frac{k-900}{800}\right]^{12}\right)\sin\left(\frac{12k^2}{10000}\right) && \text{for } 0 \le k \le 2047\\
&Y[k] = -Y[4097-k] && \text{for } 2049 \le n \le 4095\\
&Y[2048] = 0, && (2.10)
\end{aligned}$$

where $X[k]$ represents the real part of the spectrum and $Y[k]$ its imaginary part.

TSP method assumes the measured system to be LTI. Non-linearities of the measured system may result in distortion peaks in the impulse response which produce cracking sound when convolved with a clean sound. Generally, as the distortion error grows with the amplitude of the emitted signal, lowering the excitation level of the signal decreases the distortion error. This, however, also leads to an increase of noise error. The best performance is achieved when error contribution from distortion and noise are equal. This ideal excitation level is system-specific and has to be set individually for every measured system based on its properties.

Furthermore, as stated by Dunn [2]: Averaging the results of N measurements will reduce the noise contribution by factor of $\sqrt{N}$ while the deterministic error due to distortion will remain constant. Thus, averaging also reduces the optimum driving signal amplitude.

2.4 Exponential Sine Sweep Method

The current state-of-the-art technique for the measurement of room acoustics is Exponential Sine Sweep (ESS) method as proposed by Farina [3]. This method not only is capable of obtaining a room's impulse response but also can measure harmonic distortion. The ESS is, unlike MLS and TSP, to some point vulnerable to time-variance of the system under test and mismatch of an input/output sampling clock.

This method emits sine sweep with exponentially increased frequency, according to

$$x(t) = \sin\left[\frac{\omega_1 T}{\ln\left(\frac{\omega_2}{\omega_1}\right)}\left(e^{\frac{t}{T}ln\left(\frac{\omega_2}{\omega_1}\right)} - 1\right)\right], \tag{2.11}$$

where ω_1 denotes starting angular frequency, ω_2 ending angular frequency and T is the total duration in seconds. This signal is shown in Figure 2.9.

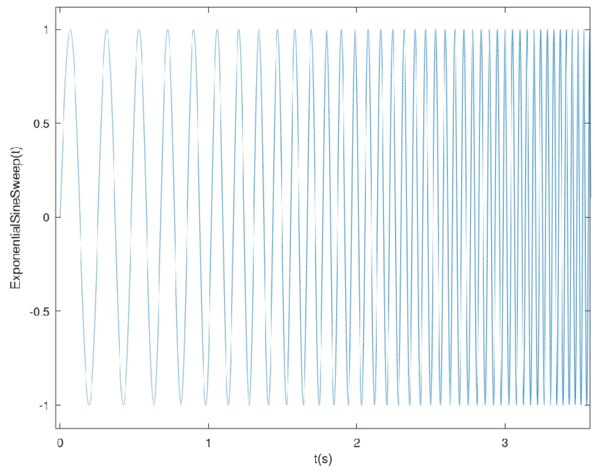


Figure 2.9: A close look on exponential sine sweep signal, starting at frequency $f_1 = 5Hz$.

The ESS method uses linear deconvolution instead of circular deconvolution (discussed in Section 2.2). This linear deconvolution has the property that when the time analysis window has the same length as the sine sweep and is shorter than the impulse response measured, the tail of the impulse is cut instead of making its way to the beginning of the impulse response and therefore the time-aliasing error is avoided. To prevent the tail-cutting, several seconds of silence are added to the end of the emitted sine sweep.

The linear deconvolution is done by using an inverse filter of the original sine sweep (Figure 2.10). As described by Stan [15], the inverse filter of an exponential sine sweep is generated in two steps:

1. The original sine sweep is reversed and delayed, so it does not begin in time before zero. This reversion of the signal leads to its inversion in its phase spectrum. Convolving the original signal with its inverse version leads to squared magnitude spectrum.
2. To compensate for the different amount of energy emitted at lower and higher frequencies, the magnitude spectrum is divided with regard to the current frequency.

Complete equation for calculation of inverse filter $I(t)$ for an exponential sine sweep signal $x(t)$ is given as

$$I(t) = \mathrm{rev}\left(x\left(t\right)\right)\ e^{\frac{-t\ \ln\left(\frac{\omega_2}{\omega_1}\right)}{T}}, \tag{2.12}$$

where rev() denotes a function reversing its input, T total duration of the sine sweep in seconds and ω_1, ω_2 denote starting/ending angular frequency.

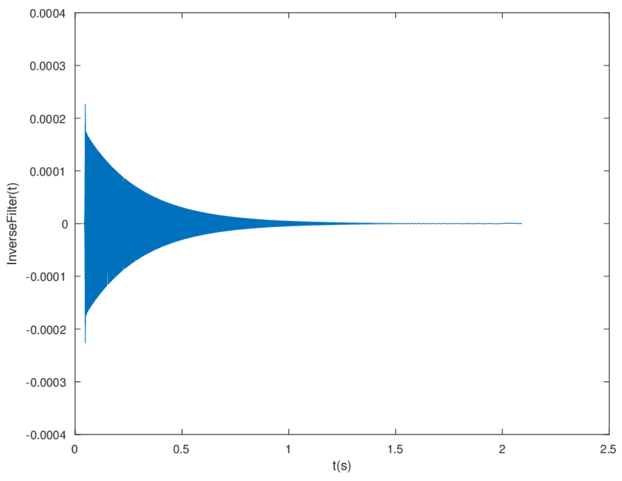


Figure 2.10: An inverse filter for exponential sine sweep.

The original sweep $x(t)$ convolved with its inverse filter $I(t)$ generates delayed Dirac's impulse $\delta(t)$ (Figure 2.11):

$$\delta(t) \approxeq x(t) * I(t) \tag{2.13}$$

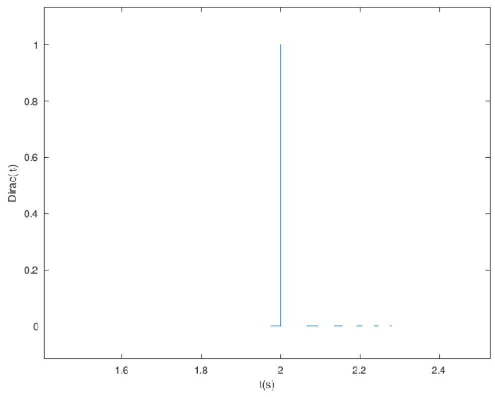


Figure 2.11: Delayed Dirac's impulse resulting from convolution of exponential sine sweep signal with it's inverse filter.

The impulse response $h(t)$ can be obtained by convolving recorded reverberated signal $y(t)$ with inverse filter $I(t)$:

$$h(t) = y(t) * I(t) \tag{2.14}$$

$h(t)$ not only contains the impulse response but also, if the sweep is slow enough, includes measured distortion located prior to the impulse response. RIR calculated using the ESS method with 15 s long sine sweep can be seen in Fig. 2.12. Notice that the impulse response begins at time t = 15 s. Up to that time, measured distortions are present. They have a form of smaller impulses where every impulse corresponds to one order of distortion. Orders of distortion are thoroughly discussed in Farina [3].

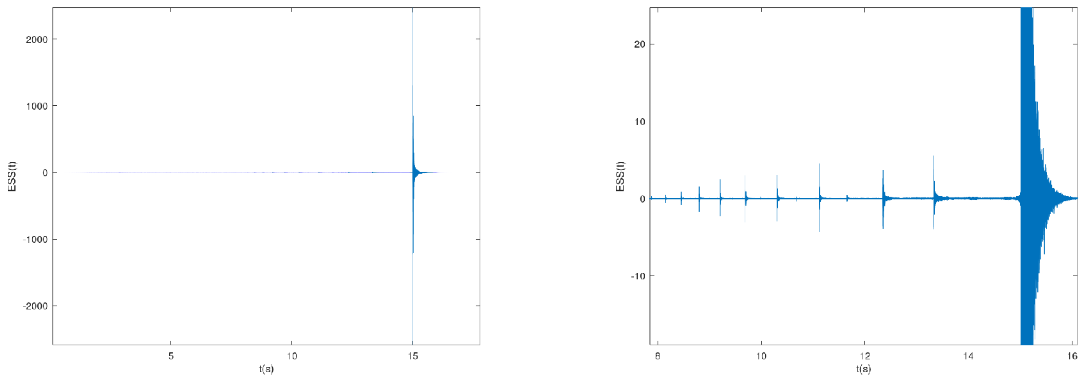


Figure 2.12: Impulse response given by ESS method calculated from 15 seconds long sine sweep.

When using an exponential sine sweep signal, non-linear behaviour (distortion) of the room results in creating clearly visible cracking peaks in the recorded reverberated sine sweep signal. In case of linear sine sweep signal, these non-linearities become noise in the reverberated signal which is correlated with the original signal and therefore, cannot be averaged as a regular noise. If standard circular deconvolution was to be used for retrieval of the RIR from the reverberated signal, these artefacts would appear within the impulse response and corrupt it. However, when using linear deconvolution, these distortion peaks or noise (depends on used sine sweep signal) move prior to the impulse response. As advantages of exponential sine sweep are obvious when, apart from impulse response, distortion is to be measured, exponential sine sweep also provides an improvement upon linear sine sweep in form of better signal-to-noise ratio in lower frequencies.

As with MLS and TSP, it is possible to average several reverberated sweeps to improve signal-to-noise ratio. In time-varying systems, however, Farina [3] suggests to rather use one very long exponential sine sweep, as this prevents harmonic distortion from appearing within the impulse response and the response is not affected by the time variation as only single measurement was realized. Signal-to-noise ratio is also satisfying as a lot of energy is emitted over a long time and is compressed to a short impulse response afterwards.

2.5 Methods Estimating Room Impulse Response from Simulation

While all the above-discussed methods were based on signal measurement in a real environment, there is another group of methods which estimates room impulse response from a mathematical model of the measured room. This estimation just from a model may come in use when signals in the room can not be measured, for example in a hazardous environment or when measuring non-existing areas (i.e. in a video game). This group of methods can be further divided into two groups.

The first group are ray-based methods where the sound is represented by rays which are propagated in free space. These rays can be seen as Dirac's impulses which are propagated only in one direction. Members of this group are Ray-Tracing method and Image Source method (ISM).

In the Ray-Tracing method, a finite number of sound rays are cast into the room at a single time. Every time a ray collides with a surface, the new trajectory of the ray, together with its new energy is computed. The resulting room impulse response is then calculated from rays captured by a microphone, located somewhere in the room.

The ISM method creates a new virtual room every time a ray should be reflected from a wall (illustrated in Figure 2.13). Every time a ray should be reflected from a surface, a new virtual room is created, with a new sound source which has the same distance from the surface as the previous one. This sound source then casts a new sound ray from its virtual room back to the ”real” room. This new ray gets into the original room in the place where the previous sound ray reached the surface and so the new ray acts as reflected original ray, with its energy reduced according to the surface, the ray was ”reflected” from. The final room impulse response is a summation of rays coming from virtual rooms into a microphone.

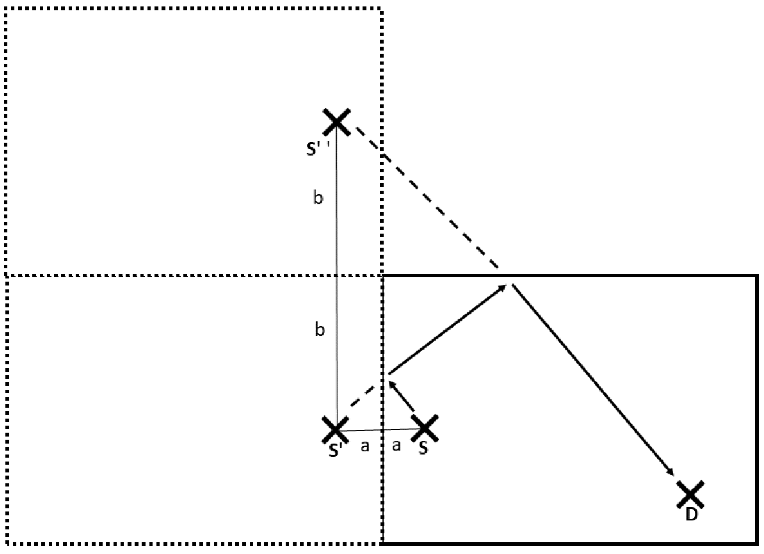


Figure 2.13: Room unfolding performed by the Image Source Method.

The second group are wave-based methods which are designed to solve the wave equation to estimate the response of the room. These methods provide more realistic results than ray-based ones but at the cost of higher computational expense. Two members of this group are the Boundary Element method and Finite Element method which discretize surface or volume to elements behaving as described by the wave equation.

Room impulse responses given from these ”simulation” methods are not very suitable for data augmentation as Ravanelli [12] noticed that usage of a room impulse response given by methods using real sounds provides a more significant improvement on word-error-rate than methods based on simulation even in a case when precise parameters of the room are used.

2.6 Summary

In this chapter, information on reverberation and room impulse response were provided. Also, the most commonly used techniques for measuring RIRs were discussed. As it was shown, not every method is based on measuring of a signal in a real room and some methods are more appropriate for some environments than others. Maximum Length Sequence (which can be easily modified into the Inverse Repeat Sequence) is more suitable for measuring noisy environment but also expects the measured room to be LTI, otherwise, the captured room impulse response may be corrupted. Exponential Sweep Method is appropriate when measuring distortion together with room impulse response.

In chapter 4, pieces of information presented here are used as we try to replicate some RIRs in the BUT ReverbDB database.

What is more, this list of methods showed us that **there is currently no method for estimation of a room impulse response estimation only from a recorded signal**.

Chapter 3

Comparison of Room Impulse Responses

As it is shown later in Section 4.3, when room impulse responses are to be compared with each other, a different metric than sample-by-sample comparison has to be implemented to obtain a result which would objectively express how similar these responses are.

In this chapter, two methodologies are presented. The first, the Time-Ratio method, compares responses in the time domain, while the second, the Frequency-Distance method, does the comparison in the frequency domain.

3.1 Time-Ratio Method for Comparison of RIRs

The Time-Ratio method proposed by the author is based on a comparison of energy in corresponding parts of aligned, normalised impulse responses in the time domain. It returns one value, describing the similarity of compared signals. The returned value starts at zero, meaning the same energy trend in signals over their duration. With increasing deviation of signals, the value returned increases as well. An implementation of this method in Octave is a part of this thesis.

The Time-Ratio method uses 2 inputs: $r[n]$ (standing for `reference`) and $t[n]$ (standing for `test`) which are the compared signals. The returned value is independent on their order.

The Time-Ratio method can be split into nine steps:

1. In the first step, a length of a segment is chosen. These segments represent parts of signal in further steps. Setting length of the segment too large will result in over-generalized signals where similarity is found even for non-similar signals. On the other hand, setting the length too low will result in intolerance for any deviations (e.g. caused by noise). For purpose of sanity check and further experiments, the

author uses length calculated as

$$N_{seg} = 0.0025 \times F_s, \tag{3.1}$$

where F_s denotes sampling frequency of the compared RIRs. This length (2.5 ms) is used as this is the time for which two room impulse responses have the same initial progression as seen in Figure 3.1.

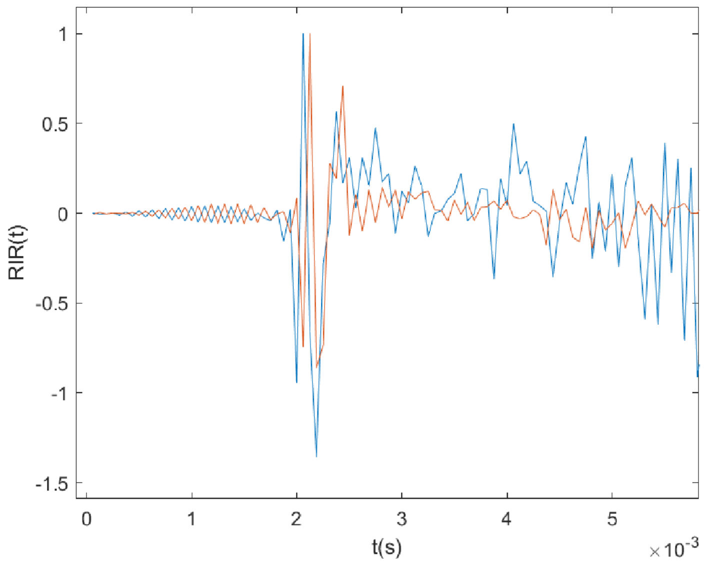


Figure 3.1: Two different impulse responses have similar progression over the first 2.5 ms.

2. In the second step, both signals are normalized so the maximal value of their absolute values equals to 1. This is happening to ensure that signals fit into a $< -1; +1 >$ range. Even when signals do already fit into the range (e.g. they fit into a $< -0.5; +0.5 >$ range), the normalization takes place anyway.

$$\begin{aligned} r[n] &= r/\max\left(|r[n]|\right) \\ t[n] &= t/\max\left(|t[n]|\right), \end{aligned} \tag{3.2}$$

where max() is a function returning maximal value of its input vector.

3. In the third step, each sample of signals is squared, as we will need energies in further computation.

$$\begin{aligned} r[n] &= r^2[n], && \text{for } 0 \leq n < R \\ t[n] &= t^2[n], && \text{for } 0 \leq n < T, \end{aligned} \tag{3.3}$$

where R denotes length of $r[n]$ and T denotes length of $t[n]$

4. In the fourth step, samples are aligned using cross-correlation. The strongest coefficient of linear cross-correlation is found and one of the signals is cut according to a lag of the coefficient so the signals have maximal correlation coefficient. This is happening to assure that the strongest samples of compared RIRs (presumably ones representing the direct sound) are in the same compared segment in further steps. An omission of this step could potentially damage correctness of result.

```
[corr, lag] = xcorr(r, t);
index = find(abs(corr) == max(abs(corr)));
shift = lag(index);

if shift > 0
  r = r(shift : end);
else
  t = t(abs(shift) : end);
endif
```

   where `xcorr` is a function taking two signals as parameters and returning `[R, LAG]`, where `R` is a vector of correlation estimates between the signals and `LAG` is a vector of correlation lags.

5. In the fifth step, signals are length-aligned. The maximal number of segments which can be filled is found. Afterwards, both signals are cut so that their lengths correspond to the length of this number of segments.

$$r[n] = \begin{cases} r[0 : N_{seg} \times \text{floor}(T/N_{seg}) - 1], & \text{if } R > T \\ r[0 : N_{seg} \times \text{floor}(R/N_{seg}) - 1], & \text{otherwise} \end{cases} \tag{3.4}$$

$$t[n] = \begin{cases} t[0 : N_{seg} \times \text{floor}(T/N_{seg}) - 1], & \text{if } R > T \\ t[0 : N_{seg} \times \text{floor}(R/N_{seg}) - 1], & \text{otherwise} \end{cases} \tag{3.5}$$

   where floor() denotes floor function and $s[i : j]$ operation returning signal made by values of s on indexes $< i; j >$.

6. In the sixth step, signals are divided into segments and for each segment, its energy is computed.

$$\begin{aligned} E_1[1] &= \sum_{n=0}^{N_{seg}-1} r[n] \\ E_2[1] &= \sum_{n=0}^{N_{seg}-1} t[n] \\ E_1[k] &= \sum_{n=(k-1)\times N_{seg}}^{k\times N_{seg}-1} r[n] && \text{for } 1 < k \leq R/N_{seg} \\ E_2[k] &= \sum_{n=(k-1)\times N_{seg}}^{k\times N_{seg}-1} t[n] && \text{for } 1 < k \leq T/N_{seg} \end{aligned} \tag{3.6}$$

7. In the seventh step, the energy values of segments of one signal, created in the sixth step contained in vectors $\mathbf{e_1}$ and $\mathbf{e_2}$, are divided by corresponding energy values of segments of the second signal. Which signal is divided by which is decided from the mean value of the resulting vector of results. The result with higher mean value is further used. This is done to assure independence of result on the order of signals (parameters):

$$\mathbf{ratios} = \begin{cases} \mathbf{e_1} \ ./ \ \mathbf{e_2}, & \text{if } \mathrm{mean}(\mathbf{e_1} \ ./ \ \mathbf{e_2}) > \mathrm{mean}(\mathbf{e_2} \ ./ \ \mathbf{e_1}) \\ \mathbf{e_2} \ ./ \ \mathbf{e_1}, & \text{otherwise} \end{cases} \tag{3.7}$$

   where ./ denotes element-wise division of vectors and mean() function returning mean value of vector on input.

8. In the eighth step, from each element of the vector, created in the seventh step, a value of one is subtracted and the absolute value is computed. Due to this, groups of the same energy have a value of zero instead of one. This is required since the method returns values from zero, meaning the same movement of energy in signals.

$$\mathbf{ratios} = |\mathbf{ratios} - 1| \tag{3.9}$$

9. An output of the method is then calculated as a mean value of the vector created in the eighth step.

$$Result = \mathrm{mean}\left(\mathbf{ratios}\right). \tag{3.10}$$

3.2 Sanity Check of the Time-Ratio Method

A sanity check was performed to prove the correctness of the Time-Ratio method. Four rooms with measured RIRs were chosen from the ReverbDB (more about the data set in Section 4.2) and for each, 5 room impulse responses were picked – 3 measurements of the same microphone – speaker pair and 2 randomly selected different responses. A visual comparison of same and different RIRs can be seen in Figure 3.2.

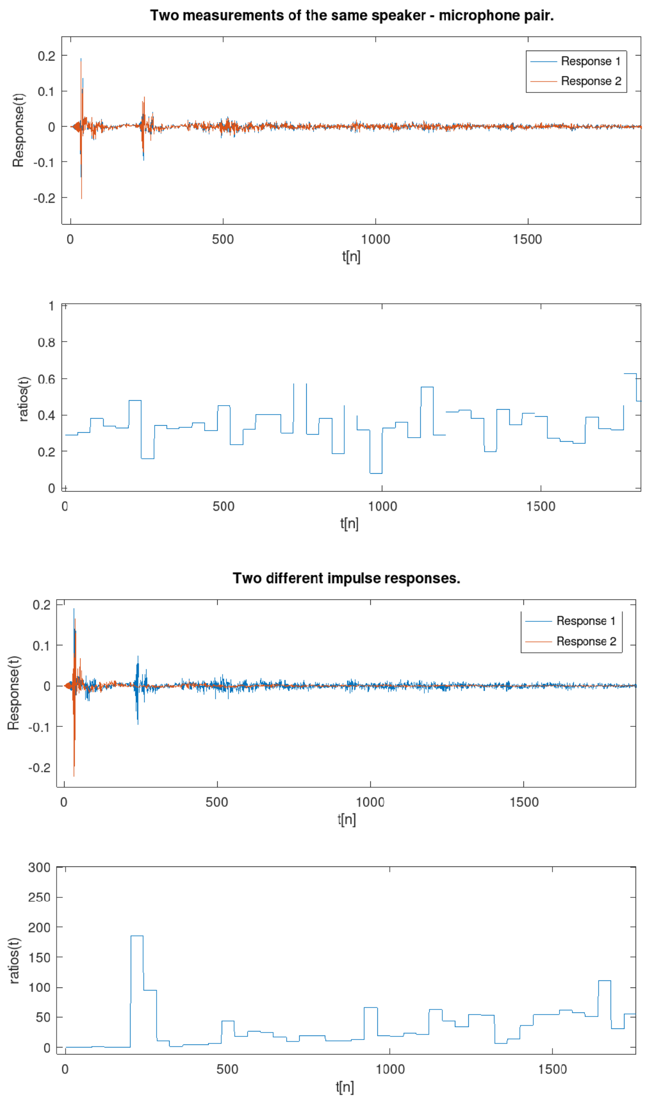


Figure 3.2: Comparison of same (upper) and different (lower) pairs of signals. As can be seen, the two measurements of the same signal copy each other almost sample-wise perfectly. The ratios(n), displays values of vector **ratios** calculated in the 8th step stretched in time so its values can be compared with signals in time.

The results of the sanity check are presented in Table 3.1. When a room was compared with itself, the comparison was made among the 3 measurements corresponding to the the same impulse response. When different rooms were compared, a sample from each room was chosen randomly. Results clearly show that the Time-Ratio method returns a value under 1 when given two measurements of the same RIR. When given two different RIRs, the method returns a large variety of values above 1, depending on a number of variables. Discovering how much such value depends on which variable is out of the scope of this thesis.

	D105	L207	Q301	L227
D105	0.373	6.068	91.059	5649.800
	0.929	6.166	203.560	7502.500
	0.453	7.332	33.225	9823.800
L207		0.399	126.110	45092.522
		0.280	12.291	3829.200
		0.454	89.981	45660.363
Q301			0.277	245132.415
			0.333	116835.356
			0.259	241227.120
L227				0.156
				0.221
				0.242

Table 3.1: Results of sanity check of Time-Ratio method. Values on the diagonal correspond to comparisons of different measurements of the same RIR. The non-diagonal values correspond to comparisons of RIRs of different rooms.

3.3 Frequency-Distance Method for comparison of RIRs

The Frequency-Distance method is based on calculating an average Euclidian distance between magnitude spectrograms of framed signals. As the returned value corresponds to the average distance between spectrograms, it is always positive. This value starts at 0, indicating that spectrograms of compared responses have the same magnitudes in all compared frames. As the deviation between compared signals grows, so does the returned value of the method. An implementation of this method written in Octave is a part of this thesis.

Inputs of this method are impulse responses $r[n]$(standing for `reference`) and $t[n]$ (standing for `test`) or matrices $\mathbf{R}$ and $\mathbf{T}$, representing their magnitude spectrograms. Rows of these matrices represent frequency bands, with elements ordered from 0 (DC component) to $N_{FFT}/2$ (half of sampling frequency). Columns represent time frames. As a distance between two points is independent on their order, so is the Frequency-Distance method independent on its inputs.

The Frequency-Distance method has 8 steps when $r[n]$ and $t[n]$ are inputs. When inputs are $\mathbf{R}$ and $\mathbf{T}$, steps 1 - 5 are omitted:

1. In the first step, signals are time aligned using linear cross-correlation of their squared values. One of the signals is zero-padded according to a lag of the strongest coefficient of the cross-correlation output. This is done to make sure that the strongest samples of $r[n]$ and $t[n]$ will be located in corresponding frames of spectrograms:

```
[corr, LAG] = xcorr(r .^2, t .^2);
index = find(corr == max(corr));
Shift = LAG(index);

if Shift < 0
  r = [zeros(abs(Shift),1); r];
elseif Shift > 0
  t = [zeros(abs(Shift),1); t];
endif
```

2. Time aligned signals are then cut to the length of the shorter one. This is happening to ensure the same length of their spectrograms.

3. In the third step, both signals are divided into frames with 50% overlap and each frame is multiplied by Hann's function of the same size as the size of frames is. The Hann's function equation is given as

$$w[n] = 0.5\left(1 - \cos\left(2\pi \frac{n}{H}\right)\right), \qquad \text{for } 0 \leq n \leq H-1. \tag{3.11}$$

   where the length of the function is H. Length of frames was set to 256 samples. This value was selected as spectrograms generated using this length of the frame have 129 frequency bins, which is the same amount as outputs of our neural networks in chapter 6 do have. This allows us to easily apply this method for all investigated techniques of RIR estimation.

4. Multiplied frames are transferred into the frequency domain using Discrete Fourier Transform and clipped from the length of N_{FFT} to $N_{FFT}/2+1$ as the upper half is the complex conjugate of the lower one. These transformed, clipped frames are arranged into matrices **R** and **T** with M rows (frequency bins) and N columns (frames).

   Steps 3 and 4 are illustrated mathematically and with code in step 6 of Section 5.4.

5. In the fifth step, absolute values of matrices **R** and **T** are calculated. This is to get rid of their phases. **R** and **T** are now magnitude spectrograms.

6. In the sixth step, matrices are normalized to unit total power using square root of summation of all elements in each matrix squared (squaring each element produces a power spectrum). This normalization allows us to compare the shape of RIRs regardless of their total powers or energies.

$$\begin{aligned}
E_R &= \sum_{i=0}^{M-1}\sum_{j=0}^{N-1} \mathbf{R_{ij}}^2 \\
E_T &= \sum_{i=0}^{M-1}\sum_{j=0}^{N-1} \mathbf{T_{ij}}^2 \\
\mathbf{R} &= \frac{\mathbf{R}}{\sqrt{E_R}} \\
\mathbf{T} &= \frac{\mathbf{T}}{\sqrt{E_T}}
\end{aligned} \tag{3.12}$$

7. In the seventh step, Euclidian distances between each two corresponding time frames are calculated:

$$d[j] = \sqrt{\sum_{i=0}^{M} (\mathbf{R_{ij}} - \mathbf{T_{ij}})^2}, \qquad \text{for } 0 \leq j \leq N-1. \tag{3.13}$$

8. In the last step, the mean value is calculated from the vector of distances between single frames. Calculation of mean value rather than summation is important as comparing longer responses would lead to a higher number of frames and therefore to biased results.

$$\mathit{Result} = \text{mean}\,(\mathbf{d}) \tag{3.14}$$

3.4 Sanity Check of the Frequency-Distance Method

A sanity check was performed to check the correctness of the Frequency-Distance method. The test set used was the same as for the Time-Ratio method in section 3.2. Results of the sanity check are shown in Table 3.2.

From the results, it obvious, which samples are two measurements of one RIR, and which are two measurements of two different rooms. The measurements of the same room do, in most cases, fit under the threshold of 0.01, while measurements of different rooms start at 0.024. All results fit in a range of $< 0; 1 >$.

	D105	L207	Q301	L227
D105	0.006	0.036	0.032	0.036
	0.006	0.028	0.036	0.039
	0.003	0.039	0.036	0.042
L207		0.001	0.025	0.046
		0.001	0.024	0.048
		0.001	0.025	0.045
Q301			0.003	0.057
			0.003	0.047
			0.003	0.053
L227				0.020
				0.017
				0.006

Table 3.2: Results of the sanity check of Frequency-Distance method. Values on the diagonal correspond to comparisons of different measurements of the same RIR. Non-diagonal values correspond to comparisons of RIRs of different rooms.

3.5 Summary

In this chapter, two methods were introduced for the purpose of comparing room impulse responses. The first one, the Time-Ratio method operates in the time domain, while the second, the Frequency-Distance method operates in the frequency domain.

Both methods return one positive value describing how much different the two compared responses are, starting at 0, which indicates equivalence. Both methods have a different threshold of returned value after which the two responses do not match. We have experimentally found that for the Time-Ratio method, this threshold is located in interval $< 0.929; 6.068 >$ (as the highest returned value for same responses was 0.929 and the lowest value returned for different responses was 6.068). For the same reasons, this interval is set to $< 0.020; 0.024 >$ for the Frequency-Distance method.

These methods are used further in this thesis as similarity between author's results and ReverbDB measurements are tested. They are also used when RIR is estimated from a speech signal.

Chapter 4

Experimental Evaluation of Standard RIR Measurements

As it was shown in Chapter 2, there is a number of methods for room impulse response measurement. In this chapter, the LibriSpeech and the ReverbDB data sets are introduced, as they are used in further experiments throughout this thesis. Then the ESS method, discussed in Section 2.4 is used in practice. Firstly, we compare its implementation done by the author with the one used by the ReverbDB and secondly, we try to replicate some of the ReverbDB results.

4.1 LibriSpeech

LibriSpeech is an English corpus publicly available[1] under the Creative Commons Attribution 4.0 International Licence. It is derived from audiobooks which are part of the LibriVox project. It contains 1000 hours of captured speech signal sampled at 16 kHz. Apart from this corpus, language model training data and pre-built language models[2] are also part of the LibriSpeech to allow easy reproduction of tests discussed in Panayotov [9].

Speech signals of the LibriSpeech are divided into parts stored in folders corresponding to voices of actors who read them. Each of these actor-representing folders is further divided into subfolders based on a chapter of the book which is read. Speech signals located in these chapter-representing folders are not stored in form of one continuous audio file but are cut into, up to 35 seconds long, tracks. Apart from the tracks themself, a text file containing transcription of these tracks is located in each subfolder. These text files are organized so that every line corresponds to one track of the subfolder. In total, the LibriSpeech is made by 2485 voice actors and 1568 books divided into 5831 chapters.

As the total size of the corpus extends 50GB, it was split into 7 subsets. The speakers were labelled as 'clean' or 'other' based on performance on ASR trained on Wall Street Journal corpus [10], where speakers with lower WER were tagged as 'clean' and speakers with higher WER were tagged as 'other'. Based on their label and sex, speakers were moved into LibriSpeech's subsets as can be seen in Table 4.1.

[1] http://www.openslr.org/12/
[2] http://www.openslr.org/11/

subset	hours	per-spkr minutes	female spkrs	male spkrs	total spkrs
dev-clean	5.4	8	20	20	40
test-clean	5.4	8	20	20	40
dev-other	5.3	10	16	17	33
test-other	5.1	10	17	16	33
train-clean-100	100.6	25	125	126	251
train-clean-360	363.6	25	439	482	921
train-other-500	496.7	30	564	602	1166

Table 4.1: Data subsets in LibriSpeech. Table taken from Panayotov [9].

4.2 ReverbDB

ReverbDB, with its full name Brno University of Technology Speech@FIT Reverb Database, is a product of BUT research group Speech@FIT.

Its main components are a set of RIRs measured for every used speaker-microphone pair, background noise of every speaker-microphone pair, detailed metadata and recordings of reverberated LibriSpeech test-clean, 2000 HUB5 English evaluation and part of NIST Speaker Recognition Evaluation 2010 data sets. ReverbDB is accompanied by a journal paper by Szőke et al. [17] with a thorough description of the set and of ASR experiments which were performed on the data set. The last part of the ReverbDB is a Kaldi-based recipe on close-talk data augmentation so everyone can replicate experiments from the journal. A part of the ReverbDB can be freely obtained[3] under the Creative Common 4.0 Licence.

The BUT ReverbDB contributes to the DRAPAK[4] project, sponsored by Czech Ministry of Interior, which focuses on speech data mining in the security domain. This includes eavesdropping devices such as ”bugs” which is the reason why some of speaker - microphone pairs in the data set take place in rather unusual locations with no clear visibility between each other.

The ReverbDB project is constantly being developed and at the time of writing this thesis, it contains reverberated data from 9 rooms (3 office rooms, 1 stairway, 2 conference rooms, 2 lecture rooms and a meeting room). For each room, the database contains 31 microphone positions and usually 5 speaker locations, creating over 150 unique speaker-microphone pairs per room.

When ReverbDB was designed, other available data sets (such as ACE Corpus and Sweet-Home sets) were studied and a conclusion was made that there is no available data set which would include retransmitted freely accessible speech data set in a large variety of rooms. It was observed that existing data sets focus rather on variety of microphone locations than on diversity of rooms and their amount. This type of variety is less useful as Ravanelli [11] pointed out that for an ASR performance, the variety of rooms is more important. Furthermore, it was discovered that all the observed data sets provide only recordings captured in ideal conditions where microphones were located within 4 meters from a speaker and with direct visibility. This placement is insufficient not only when intelligence ”gadgets” are developed but just as well when personal assistants are built as many users do not stand in a direct line of sight when interacting with them. Metadata

[3]https://speech.fit.vutbr.cz/software/but-speech-fit-reverb-databa
[4]https://www.fit.vut.cz/research/project/1009/.en

in other data sets were problematic as well since there was often no information on precise coordinates and directions of microphones and speakers.

As already stated, the above mentioned shortcomings were taken in mind when ReverbDB was created: it contains retransmitted freely available speech data set (the LibriSpeech test-clean subset), variety of rooms with different acoustics, microphones located in remote places as well as in close, visible positions and detailed metadata describing measured rooms and placing of speakers and microphones.

4.3 Comparison of Author's and ReverbDB's Approach to the Exponential Sine Sweep Method

In this section, a comparison of the Exponential Sine Sweep method used by the author and ReverbDB is provided. A link[5] to the Matlab code for calculating room impulse response used in ReverbDB (further called FSC (Free Source Code)), was shared in Szőke et al. [17] and it is publicly available. Author's implementation of the ESS method is a part of this thesis.

In the first part, exponential sine sweep and its inverse filter generation, both the author and FSC use Equation 2.11 to generate the sine sweep signal. FSC then, unlike the author, sets the value of samples located after the last intersection of the resulting sine sweep with the x-axis to zero (Figure 4.1).

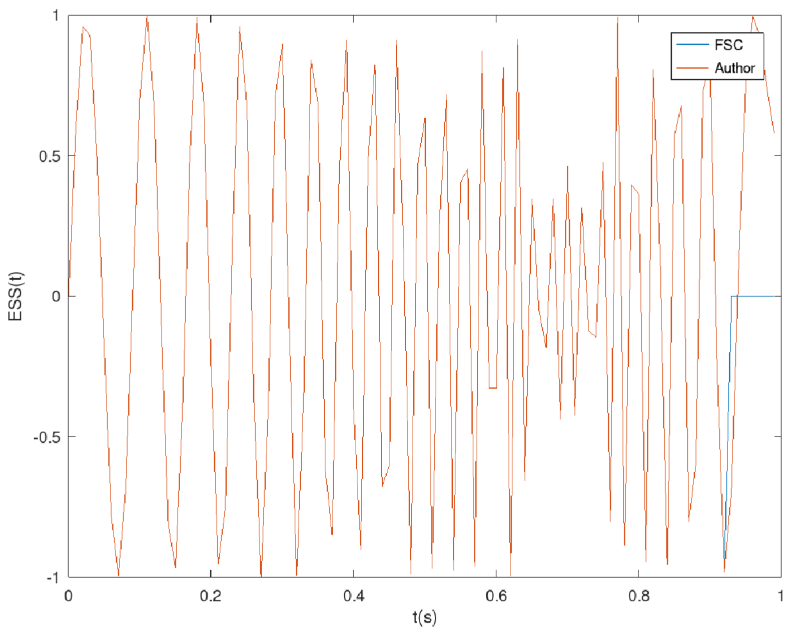


Figure 4.1: Comparison of authors's and FSC's sine sweep. Values are the same until the last intersection of signal with x-axis. FCS's values are further on set to zero.

[5] http://freesourcecode.net/matlabprojects/69639/exp.sweep-and-impulse-response-in-matlab

When generating the inverse filter, the author and FSC use Equation 2.12. In the FSC approach, the resulting signal is then normalized as

$$I(t) = I(t)/\max\left(\mathrm{abs}\left(I(t)\right)\right), \tag{4.1}$$

where $I(t)$ is the inverse filter. Since the maximum of $\mathrm{abs}\left(\sin\left(x\right)\right)$ is 1 and Equation 2.12 decreases signal in time, this normalization has no other effect than to "beautify" the resulting signal by setting its maximum absolute value to 1.

In the second part, extraction of RIR from reverberated signal, author's Octave code retrieves the room impulse response as:

```
1  function RIR = calculateRIR (rev,f1,f2,T,fs)
2    inv = inverseFilter(f1,f2,T,fs);
3
4    RIR = cconv(rev, inv);
5    RIR = RIR(round(length(RIR)/2): round(length(RIR)/2)+fs);
6    RIR = RIR / max(abs(RIR));
```

where `rev` denotes reverberated recorded sine sweep, `f1` starting frequency, `f2` ending frequency, `T` length of sine sweep in seconds, and `fs` sampling frequency.

On line 2, the `inverseFilter` function is used to generate an inverse filter of the original sine sweep in author's approach.

On line 4, the ESS method output is computed using a `cconv` function. The `cconv` function convolves signals on its input. This ESS method output, apart from the impulse response, also contains measured distortion, as it was discussed in Section 2.4.

On line 5, one second long part is cut from the impulse response computed on line 4, originating in the middle of the signal. This new RIR now contains only the desired impulse response.

On line 6, this impulse response is normalized. This is because Octave/MATLAB represent sound signals as arrays of values in range <-1;+1> and the RIR before its normalization may possess values out of this range.

The core of FSC's MATLAB code for retrieving room impulse response is:

```
1  function h =sweepIR(rec, Nimp,T,f1,f2,offset,fs)
2    [temp,filt]=expsweep(T,f1,f2,0,fs);
3
4    t_end=find(20*log10(abs(rec(end:-1:1)))>-20,1,'first');
5    t=length(rec)-t_end-T*fs-0.5*fs;
6    rec=rec(t:end,:);
7    filt=[filt;zeros(round((length(filt)+length(rec))/2),1)];
8    h_full=fftfilt(rec,filt);
9    h_full=h_full./max(max(abs(h_full)));
10
11   t_start=find(20*log10(abs(h_full))>-20,1,'first')-offset;
12   h=h_full(t_start+1:t_start+Nimp,:);
```

where `rec` denotes recorded sine sweep, `T` length of sine sweep in seconds, `f1` starting frequency, `f2` ending frequency, `offset` silence before the sweep in samples and `fs` sampling frequency. Argument `Nimp` is nowhere documented or mentioned, the author assumes it to be the required length of the returned room impulse response.

On line 2, the `expsweep` function is used to generate an inverse filter of the original sine sweep in FCS's approach.

On lines 4 to 6, a clipping of the recorded signal can be seen, based on its volume. This part, however, raised an index error every time the author tried to run it (line 5 always returns a negative number) and therefore these lines of code had to be removed to make the code executable.

On line 7, the extension of the inverse filter is performed in order to ensure sufficient length of the resulting room impulse response.

On line 8, `fftfilt` function is used instead of function `cconv`. The `fftfilt` returns the same values as function `filt`. The difference is that while `filt` multiplies each element of the filter with each element of signal while `fftfilt` uses an FFT-based method of overlap-adding, making it more suitable for longer signals (i.e. sound signals). The `filt` function differs from the `cconv` function in the length of the returned array. While `filt` returns an array of length N, where the N is the length of the filtered signal, `cconv` returns an array of length $M + N - 1$, where the M denotes length of the filter. Since the first N samples of `cconv` are the same as those of `filt` and only a part of the resulting output is required, both functions are suitable.

Comparison of results of author's and FSC's implementation is shown in Figure 4.2. As it was shown in this section, author's and FSC's approach differ, when calculating room impulse response using the Exponential Sine Sweep method. Using different signals and different ways of handling them makes it impossible to compare outputs sample-by-sample.

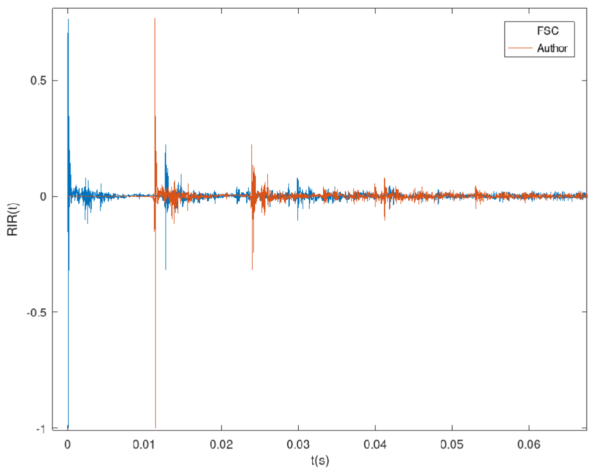


Figure 4.2: Comparison of room impulse responses generated by author and FSC. Notice their similarity, except of the shift presented in author's response.

4.4 Comparison of Replicated Room Impulse Responses with ReverbDB Values

Using the Time-Ratio method, it is possible to compare results of the author's ESS method implementation with the values of the ReverbDB (calculated using FSC's implementation).

To test, how author's room impulse responses correspond to responses contained in the ReverbDB, from each room contained in the data set, 3 impulse responses were randomly picked. These responses were then replicated, using the author's ESS implementation and compared using the Time-Ratio method. As the sampling frequency of reverberated sine sweeps is 48 kHz, the results have this sampling frequency. Since the sampling frequency of impulse responses contained in the ReverbDB is 16 kHz, the 48 kHz results were downsampled using Octave/MATLAB command `resample`. Reverberated sine sweeps were provided by Igor Szőke.

Results are shown in Table 4.2. All results are set under or close to the value of 1 and the range of $< 0.929; 6.068 >$ was not exceeded for a single value. The average value of deviations between corresponding results is 0.392, which is a value similar to results of Table 3.1 when the same room impulse response was measured several times. This proves the validity of author's implementation.

	Sample #1	Sample #2	Sample #3	**AVG**
C236	0.088	0.045	0.142	0.092
Conference Room 2	0.104	0.235	1.319	0.553
D105	0.041	0.070	0.313	0.141
E112	0.382	0.101	0.481	0.321
L207	0.992	0.389	0.463	0.615
L212	0.957	0.414	1.220	0.864
L227	0.238	0.075	0.072	0.385
Q301	0.184	0.139	0.041	0.121
Room 112	0.254	0.078	0.976	0.436
				0.392

Table 4.2: Results of comparison between replicated room impulse responses and original data measured using the Time-Ratio method.

4.5 Summary

In this chapter, audio data sets were introduced and the Exponential Sine Sweep method was used in practice. In the first section, the LibriSpeech dataset was briefly introduced, as it is used in the ReverbDB. The ReverbDB itself was discussed in the second section, where it was stated which data it contains, what is the motivation behind it and who the authors are. Next, a comparison between the author‘s implementation of the ESS method and implementation used by ReverbDB creators was made. This comparison revealed that impulse responses computed by author and ReverbDB cannot be compared sample-by-sample. For this purpose, the Time-Ratio method presented in Chapter 3 was used for a comparison of results. In the fourth section, the two implementations of the ESS method were compared and their correctness was proven.

The data sets discussed in this chapter are further used when we try to estimate room impulse response only from reverberated speech signal as the ReverbDB not only contains various recorded voice signals but also contains impulse responses describing their reverberation. This will allow us to check the validity of results.

Chapter 5

Analytical Approach to RIR Estimation from a Speech Signal

As we have seen in previous chapters, calculation of a room impulse response is a straightforward task, when both clean and reverberated forms of a signal are provided. This however changes, when only a reverberated form of a speech signal is available. In this chapter, we try to blindly estimate room impulse response from impulse-like phonemes found in speech signal and we check the correctness of this estimation against the measured impulse response from ReverbDB using numerical metrics and listening evaluation.

5.1 Principles

We know from signal processing, that to recover an impulse response of a system, we need to excite it with a Dirac pulse $\delta(t)$, that, in discrete time, becomes a unit pulse of $\delta[n]$

When we try to estimate the impulse response of a room using only a speech signal, we do not have any signal strongly resembling the Dirac's impulse as a handclap or a gunshot. Still, some phonemes (units of speech), such as 'P', 'K' or 'T' do, to some point, resemble an impulse as they emit energy over a short time. Their reverberated forms can then be seen as approximations of the room impulse response we are trying to estimate (see Figure 5.1). In this chapter, we try to estimate RIR using these impulse-like phonemes and their reverberation.

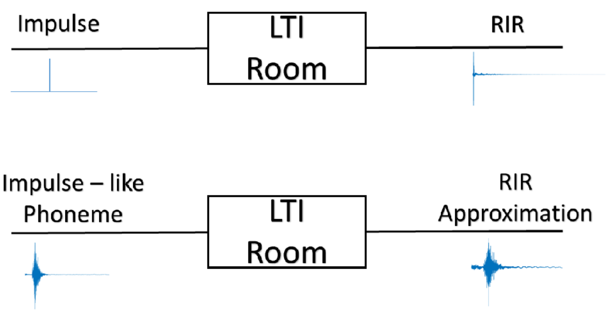


Figure 5.1: Reverberated impulse-like phoneme can be seen as an approximation of a room impulse response.

Extraction of these phonemes can be done in several ways based on additional information provided (such as transcription or clean form of speech). A scheme showing these ways and their accuracy can be seen in Figure 5.2. In practice, the fourth possibility would have to be handled (estimation using only reverberated speech), in this thesis a functionality of proposed methods is tested using phonemes extracted using all additional information (a clean form of speech and text transcript). In case of proving the methods functional, extraction of phonemes using less additional information would have to be dealt with (i.e. using ASR or windowing a signal and analyzing the histogram of its squared values).

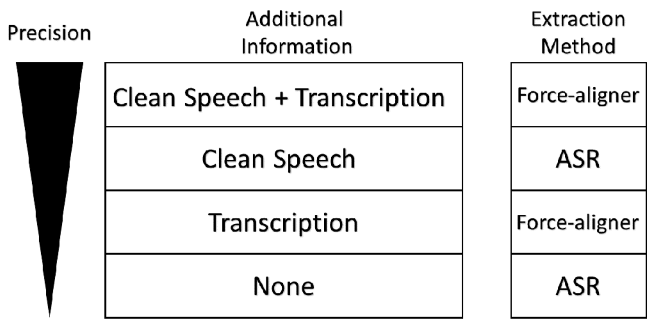


Figure 5.2: Additional information provided, its effect on precision of phoneme extraction and possible approach to the phoneme extraction. We assume that we always have the target (reverberated) speech available.

5.2 Phoneme Selection

As it was stated, when we try to estimate an impulse response using a phoneme, it is reasonable to search for a phoneme, that itself resembles an impulse. This means a phoneme, which contains a short burst of energy followed by silence. A 'T' phoneme did show such property, as a voiceless plosive[1], and therefore is used in further experiments. For illustration, a comparison between the 'T' phoneme and a 'K' phoneme is shown in Figure 5.3.

However, detection of every 'T' phoneme in a speech signal would not be sufficient. As the duration of reverberation may last hundreds of milliseconds or more, it is desired to have the longest possible silence **before** the 'T' phoneme as a reverberation of previous speech could damage the reverberation of the 'T' phoneme.

Also, an appearance of any speech signal located shortly **after** the 'T' phoneme would corrupt the reverberation of the 'T' phoneme and therefore it is important to have silence located after the phoneme as well.

Requirements of having silence present before and after the phoneme are not realistic as single 'T' phonemes do not occur in standard speech signals. For this reason, the requirement for silence before the phoneme was changed to a requirement for silence or a noise-like phoneme (e.g. 'HH', 'V') before the impulse-like phoneme. As these noise-like phonemes do not possess a high amount of energy, their reverberation should not affect the reverberation of the 'T' phoneme.

[1]https://simple.wikipedia.org/wiki/Stop_consonant/

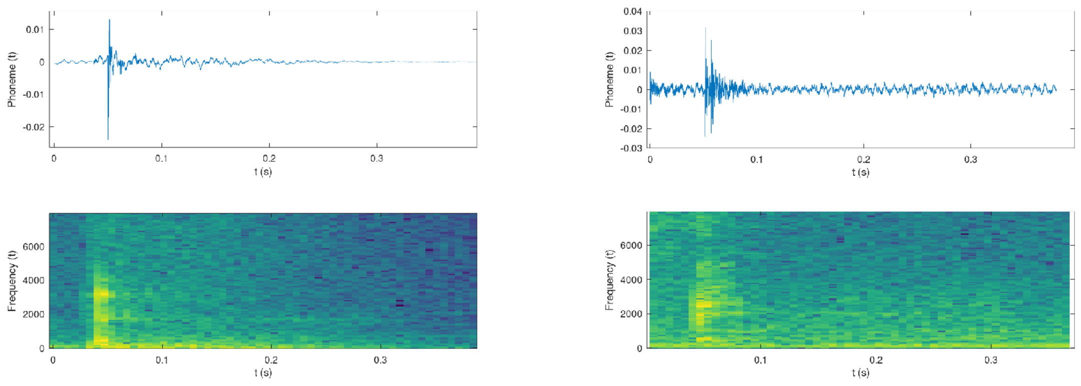


Figure 5.3: Comparison of a 'T' phoneme (on the left) and a 'K' phoneme (on the right). Notice that 'K' is longer and slightly voiced.

The lengths of silent segments were chosen to be at least 100 ms before the phoneme and 400 ms after the phoneme. Such times were picked as they should be suitable for further analysis and it is not uncommon to find a 'T' phoneme with such conditions.

Another important property of the phoneme is its level of noise against the sound of the phoneme. As obtaining a silence of the measured room is not complicated, SNR can be calculated as

$$10\log_{10}\left(\frac{E_p}{E_{sil}}\right), \tag{5.1}$$

where E_p is energy of the 'T' phoneme signal followed by silence and E_{sil} is energy of background noise signal of the measured room. A threshold for phoneme usability was selected as 6 dB of phoneme over the silence of the room. Low SNR ratio is usually caused by a too silent sample of 'T' phoneme, which is exposed to background noise and therefore useless.

5.3 Phoneme Extraction

To extract phonemes from the reverberated speech signal, we use every additional information possible, as was already stated. This can be done since we use the ReverbDB for the testing which uses publicly available data set LibriSpeech that includes a transcription of its speech signals. Therefore, a force-aligner can be used to detect locations of phonemes in signal.

To match the transcription with speech signal, freely available Montreal Force-Aligner[2] is used. For sake of precision, non-reverberated (the original LibriSpeech) speech signals are used as inputs of the force-aligner.

The force-aligner takes as input speech signals and their transcripts of the same name (i.e. *file1.wav* and *file1.txt*) and transforms them into a TextGrid format (i.e. *file1.TextGrid*) containing a transcript where every phoneme of each word of the speech has its time of beginning and time of the end. TextGrid files are then searched for 'T' phonemes with noise-like phonemes or silence in their surroundings by a `GetPhonemes.py` script (which is

[2] https://montreal-forced-aligner.readthedocs.io/en/latest/

a part of this thesis). An output of this script (i.e *Phonemes.txt*) is a text file containing, on each line, information about suitable phoneme found. This text file is then, together with reverberated forms of speech signals, used as input for a `SeparateSound.py` script (also a part of this thesis) which extracts phonemes from the speech signals into .wav files (*0.wav*, *1.wav*. . . *N.wav*). As extracted phonemes are several hundreds of milliseconds long, the few samples long shift between clean and reverberated form of speech can be neglected.

5.4 Phoneme Combination-based Method

An analytical method based on a phoneme combination is presented in this section. This method is based on an assumption that, as it was stated above, reverberated forms of impulse-like phonemes resemble impulse response of the room. However, these reverberated phonemes differ speaker to speaker and are noised by, at least, background noise. Therefore, using only one phoneme extracted from speech would not be sufficient. As we know from signal processing, combining several estimations leads to better results. For this reason, in this method, several phonemes are combined into one. This is illustrated in Fig. 5.4. This combination has to be done by averaging magnitude spectrograms of phonemes, as averaging the phonemes in time or combining their spectrograms containing their phase would lead to their mutual attenuation and the result would be close to zero for all values. Phases of the longest phoneme used are applied onto resulting magnitude spectrogram. This combination of phonemes should eliminate non-correlated noise and speaker dependency. An Octave implementation of this method is a part of this thesis.

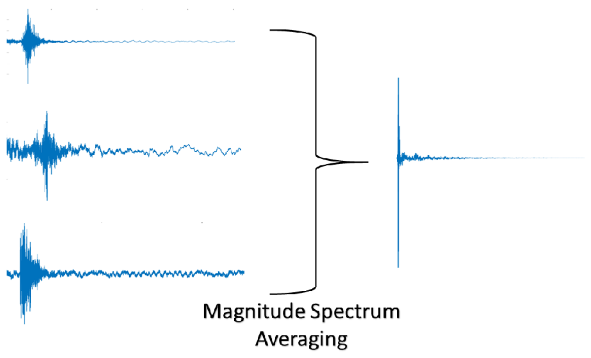


Figure 5.4: In Phoneme Combination-based method, several impulse-like phonemes are combined into one, using their magnitude spectrogram averaging, to obtain an estimation of a room impulse response.

Input for this method is a matrix $\mathbf{X}$ (called `Matrix` in code) containing one extracted phoneme on every line. These phonemes are zero-padded on their right side, so the number of columns of the matrix corresponds to the length of the longest phoneme extracted. This matrix has M rows (phonemes) and N columns.

The method is split into the following steps:

1. In the first step, the power of each phoneme is calculated as

$$P_i = \frac{1}{L}\sum_{n=0}^{L-1} x_i[n]^2, \tag{5.2}$$

   where $x_i[n]$ is the i^{th} phoneme and L is the phoneme's length without padding. The phoneme with the biggest power is moved to the first row of $\mathbf{X}$ and will be used as a **reference phoneme** from now on.

2. Next, all the phonemes are aligned with the referential phoneme. Length of the shift for each, except the reference, phoneme is calculated using cross-correlation of its squared, normalized to unit size of maximal absolute value, form.

```
for sample=2:rows
  Matrix(sample,:) = CorrAlign(Matrix(1,:), Matrix(sample,:));
endfor
```

   where `CorrAlign` is a function taking 2 signals as parameters and returning the second signal aligned to the first one. This is done using zero-padding or cutting, based on the result of cross-correlation of the two signals. See similar signal alignment done in Section 3.3.

3. In the third step, an index of the longest phoneme is found. Its phases will be used in further steps. At the same time, a vector $\mathbf{v}$ with length N is created. Each of its elements is set to a number of non-zero values in the corresponding column of $\mathbf{X}$. This creates a vector, indicating the number of phonemes present in each column of the $\mathbf{X}$. Visualization of this vector is in Figure 5.5.

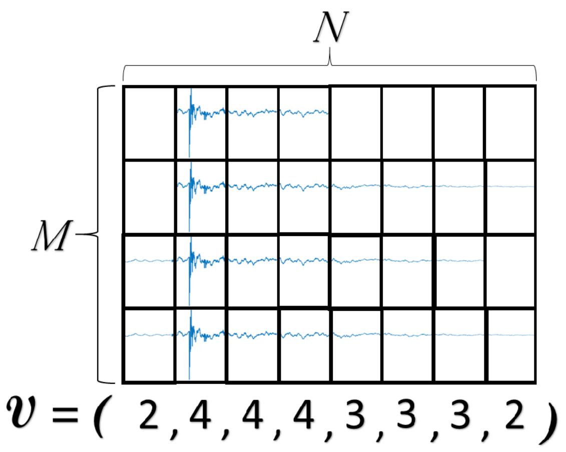


Figure 5.5: $\mathbf{v}$ vector contains number of signals for each column of $\mathbf{X}$.

```
v = zeros(colsOfMatrix, 1);
longestSignal = 0;
longestSignalLength = 0;
for signal = 1:rows
      starts = find(Matrix(signal,:), 1, "first");
      ends = find(Matrix(signal,:), 1, "last");
      v = v +
      [zeros(1, starts-1), ones(ends-starts+1,1), zeros(1, colsOfMatrix-ends)];

      if(ends-starts > longestSignalLen)
        longestSignal = signal;
        longestSignalLen = ends - starts;
      endif
endfor
```

4. Afterwards, each phoneme is normalized to zero mean value and unit standard deviation:

$$\mathbf{x_i} = (\mathbf{x_i} - \text{mean}(\mathbf{x_i}))/\text{std}(\mathbf{x_i}), \qquad \text{for } 0 \leq i < M \tag{5.3}$$

   where $\mathbf{x_i}$ denotes vector containing i^{th} signal of $\mathbf{X}$ and std() function calculating the standard deviation of vector on its input.

5. In the fifth step, phonemes are divided into frames, with 50% overlap. Their mean value is again subtracted in each frame and each frame is multiplied by a square root of Hann's function. Square root of the Hann function is especially suitable for spectrogram manipulations as it ensures smoothing of transitions when re-synthesizing the signal. Standard frame length, when dealing with speech, is about 25 ms. This roughly corresponds to 512 samples at a sampling rate of 16 kHz. This length of window is therefore used, however various frame lengths are tested in Section 5.5.

6. Next, every frame from the previous step is transformed into the frequency domain using a Discrete Fourier Transformation. These transformed frames are clipped from length of N_{FFT} to $N_{FFT}/2+1$ (e.g. frame of length 512 is clipped to a length of 257). This is due to the redundancy of the upper half, which is the complex conjugate of the lower half of the spectrum.

   The following code includes steps 5 and 6 as their code-separation would require extra space due to needed code-leading constructions.

```
w = hann(WindowSize)'.^(1/2);
Nframes = 1 + floor((colsOfMatrix - WindowSize) / (WindowSize/2));
S = zeros(WindowSize/2+1, Nframes, rows)
nstart = 1;

for ii =1:Nframes
  for signal = 1:rows;
      frame = (Matrix(signal, nstart:nstart+WindowSize-1) -
           mean(Matrix(signal, nstart:nstart+WindowSize-1))).* w;
      FRAME = fft(frame);
      S(:,ii, signal) = FRAME(1,1:WindowSize/2+1)';
  endfor
  nstart = nstart + WindowSize/2;
endfor
```

Each layer of the **S** tensor then represents a spectrogram of one phoneme. Every specrogram has $M_S = \frac{N_{FFT}}{2} + 1$ rows (frequency bins) and N_S columns (frames).

$$\mathbf{S}(:,:,m) = \begin{pmatrix} x_0(1) & x_0(t) & \dots & x_0(N_S) \\ x_k(1) & x_k(t) & \dots & x_k(N_S) \\ \vdots & \vdots & \ddots & \vdots \\ x_{M_S-1}(1) & x_{M_S-1}(t) & \dots & x_{M_S-1}(N_S) \end{pmatrix}, \quad \text{for } 0 \leq m < M \tag{5.4}$$

where $x \in \mathbb{C}$, M denotes number of phonemes in $\mathbf{X}$, k frequency bins and t time in frames. The $\mathbf{S}(x, y, z)$ indexing syntax denotes element of tensor $\mathbf{S}$ on x^{th} row, y^{th} colomn and z^{th} layer. The : character denotes every element on given dimension.

7. In the seventh step, frames are combined. This is done by averaging magnitudes of corresponding frames. This creates a new sample, composed of frames throughout samples. Averaging frames of this resulting sample using only the total number of phonemes used can not be performed, as the number of used samples is different across frames (see step 3). Also as we are combining signals in spectra, we can not locate the exact number of samples from count vector **v**. Therefore we search the vector for the maximal value in the range corresponding to averaged frame to be on the safe side.

$$\mathbf{S_{out}} = \sum_{m=0}^{M-1} |\mathbf{S}(:,:,m)|$$
$$\mathbf{S_{out}}(:,n) = \mathbf{S_{out}}(:,n) / \max(\mathbf{v}((n-1)\frac{N_{FFT}}{2} + 1 : (n+1)\frac{N_{FFT}}{2})), \quad \text{for } 0 \leq n < N_S \tag{5.5}$$

v vector created in step 3, max() function returning maximal value from input vector, and $\mathbf{v}(i : j)$ operation returning vector made by values of **v** on indexes $< i; j >$.

8. In the eighth step, phases are added into this resulting averaged sample which now contains only averaged magnitudes. For this purpose, phases of the longest sample, which was found in step 3 are used. Parts of the sample's phases are copied for every corresponding frame. Using phases of shorter signal could corrupt the resulting combined phoneme in areas beyond the length of phases donator.

$$\mathbf{S_{out}}(m,n) = \mathbf{S_{out}}(m,n)\ \mathrm{e}^{\,j\ \mathrm{angle}(\mathbf{S}(m,n,l))}, \qquad \begin{array}{l}\text{for } 0 \leq m < M_S \\ \text{for } 0 \leq n < N_S\end{array} \tag{5.6}$$

   where l denotes index of the longest phoneme in $\mathbf{X}$ and angle() denotes a function returning the argument of its input in radians.

9. After this, the resulting signal is transformed into the time domain and assembled into one continuous signal. This is achieved by applying the Inverse Fourier Transformation on every frame of the resulting sample. Every transformed frame is then once again multiplied with the square rooted Hann's function and arranged into one continuous signal.

```
Result = zeros(1,cols);
nstart = 1;

for ii = 1:Nframes
  SPEK = SOut(:, ii)';
  FRAME = [SPEK conj(fliplr(SPEK(2:WindowSize/2)))];
  frame = ifft(FRAME);
  Result(nstart:nstart+WindowSize-1) = Result(nstart:nstart+ WindowSize-1)
      + frame .*w;

  nstart = nstart + WindowSize/2;
endfor

Result = real(Result);
```

   Function `fliplr()` returns a copy with order of its columns reversed (i.e. $\mathrm{fliplr}([1,2;3,4]) = [2,1;4,3]$) and is used to complete the spectrum with its upper part before the inverse DFT.

10. Finally, as previous steps were performed in order to suppress noise and speaker dependency, frames estimated from only one signal could not be improved enough and are cut away from the result.

```
start = 1;

for i = 1:Nframes
  AvgSignals = mean(divideVector(start:start + WindowSize-1));
  if(AvgSignals < 1.5)
    Result = Result(1:start);
    break
  endif
  start = start+ WindowSize/2 ;
endfor
```

A comparison of room impulse response estimation with the measured impulse response of the estimated room can be seen in Figure 5.6. The noise in the latter part of the estimation was not completely eliminated. Also, the direct sound is not as significant as in case of the measured impulse response. Notice, however, that both responses are primarily located in the first 0.2 seconds.

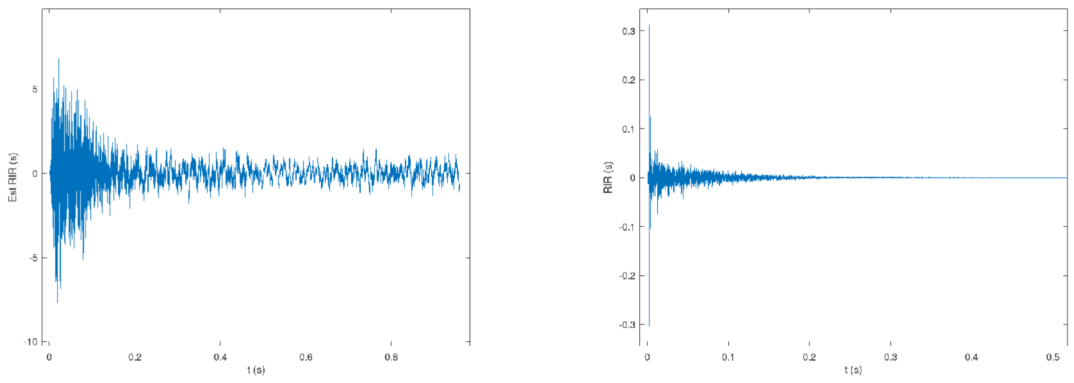


Figure 5.6: Comparison of the estimated impulse response obtained using the phoneme combination-based method (left) and reference RIR (right).

5.5 Testing of Phoneme Combination-based Method

To test the method, ’T’ phonemes from microphone number 1 in rooms L207 (SpkID01), Q301 (SpkID01) and microphone number 2 in room D105 from the ReverbDB data set were chosen. This was due to low reverberation of the microphones and their close distance to speaker. These conditions were found desirable for initial experiments as in this environment, a general efficiency of the method can be tested and in case of success, it can be tested in others, more reverberated or noisy environments.

The similarity between the output of the method and the referential room impulse response is measured using the Frequency-Distance method presented in Section 3.3 as it omits possible deviations in phase, which may be neglected when convolving/deconvolving in the frequency domain using spectrograms. Hann function of length 512 was used. The baseline was obtained as the best RIR estimate using a single ’T’ phoneme. Results of testing are shown in Table 5.1.

	D105	L207	Q301
Number of phonemes	14	9	9
Baseline	**0.071**	**0.074**	**0.074**
Results of method	0.073	0.079	0.093

Table 5.1: Results of testing of Phoneme Combination-based method.

As we can see, the baseline was not overcome for any microphone. Also, the distance between the estimation and measured room impulse did not get into or below the range of

$< 0.020; 0.024 >$ which, as was presented in Section 3.5, empirically makes the difference between the room impulse response corresponding to the measured room and an impulse response of a different room.

Another experiment was performed with increased/decreased size of the Hann's function to 1024/256 samples. The same phonemes were used as for the experiment in Table 5.1. Results can be seen in Table 5.2. Even with better performance achieved by lowering the frequency resolution, baseline values were not met or overcome. For further experiments, however, length of Hann's function's is set to 256 samples as it brings the best results in most cases.

	D105	L207	Q301
Baseline	0.071	**0.074**	**0.074**
Hann 256	0.074	0.078	0.082
Hann 512	0.073	0.079	0.093
Hann 1024	**0.070**	0.084	0.087

Table 5.2: Results as function of Hann's window length.

5.6 Clean Phoneme Deconvolution Method

Another tested analytical approach for RIR estimation is based on an assumption that the difference between speaker specifics of 'T' phonemes can be neglected and their deconvolution with a clean form of the phoneme, prepared in advance, returns a room impulse response similar to the referential impulse response.

The Clean Phoneme Deconvolution method deconvolves (using circular cross-correlation discussed in Section 2.2) extracted reverberated phonemes with a preloaded clean form of these phonemes as

$$e[n] = r[n] \circ c[n], \tag{5.7}$$

where the estimation is calculated using reverberated phoneme $r[n]$ (standing for `reverberated`) and a clean form of different phoneme $c[n]$ (standing for `clean`). The $\circ$ denotes circular cross-correlation operator.

An Octave implementation of the method is a part of this thesis. The `XCORR` function used within the code was taken from MATLAB File Exchange[3].

As an input, the method takes a signal of a reverberated phoneme `r` and a signal of a clean phoneme `c`.

The method follows these steps:

1. First, the needed size of signal padding is calculated to prevent time aliasing from occurring. The needed length of circular cross-correlation is at least

$$R + C + 1, \tag{5.8}$$

where R denotes length of signal $r[n]$ and C length of signal $c[n]$. Each phoneme is then zero-padded according to the calculated value.

```
L = length(r)+length(c)+10;
c = [c; zeros(L-length(c),1)];
r = [r; zeros(L-length(r),1)];
```

[3]https://www.mathworks.com/matlabcentral/fileexchange/4810-circular-cross-correlation

2. In the second step, `r` and `c` are aligned using cross-correlation, as in step 2 in the Phoneme Combination-based method (Section 5.4).

   ```
   r = CorrAlign(c, r);
   ```

3. Next, aligned signals are circularly cross-correlated according to Equation 5.7.

   ```
   RIR = CXCORR(r, c);
   ```

4. In the fourth step, correlation coefficients are cut after the first appearance of zero-valued coefficient as this indicates the correlation between zero-padding of signals, after which a 'tail' of circular cross-correlation is located.

   ```
   ends = find(RIR == 0, 1, "first");
   RIR = RIR(1:ends);
   ```

5. Finally, possible coefficients of the correlation which are located before the maximal coefficient (representing the direct sound) are cut as the initial direct sound part of room impulse response posses the maximal coefficient.

   ```
   starts = find(RIR == max(RIR));
   RIR = RIR(starts:end);
   ```

A comparison of resulting room impulse response estimation with the reference impulse response of the estimated room can be seen in Figure 5.7. We can observe that differences between input phonemes caused the resulting estimation to have greater reverberation than the measured room impulse response has.

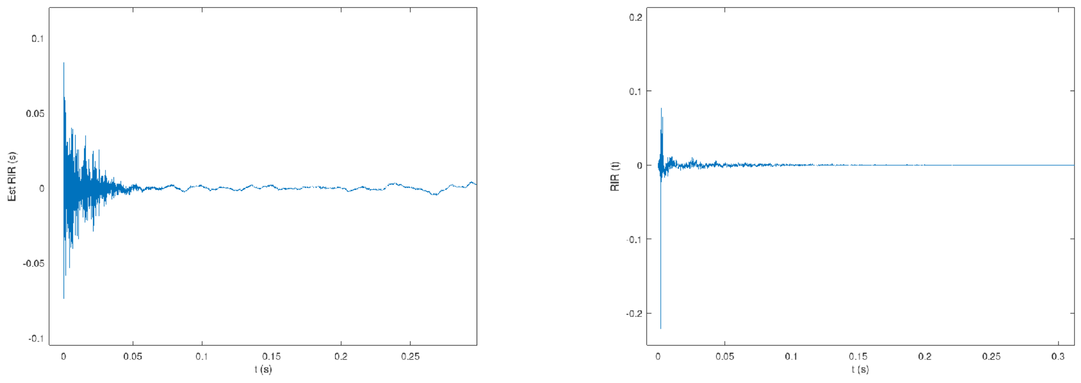


Figure 5.7: Comparison of the estimated impulse response obtained using the Clean Phoneme Deconvolution method (left) and reference RIR (right).

5.7 Testing of the Clean Phoneme Deconvolution Method

For testing of the Clean Phoneme Deconvolution method, 3 'T' phonemes from LibriSpeech were chosen as `c`. For each microphone, they were applied onto 3 different reverberated 'T' phonemes with the highest SNR. If the phoneme was about to be deconvolved with itself, 'T' phoneme with 4^{th} highest SNR was picked instead.

Tested rooms and microphones were chosen the same as for the testing of the Phoneme Combination-based method.

For the measurement of similarity between the result and measured impulse response the Frequency-Distance method is used again as it allows us to compare results of both methods presented. The baseline was obtained as the best RIR estimate using single 'T' phonemes.

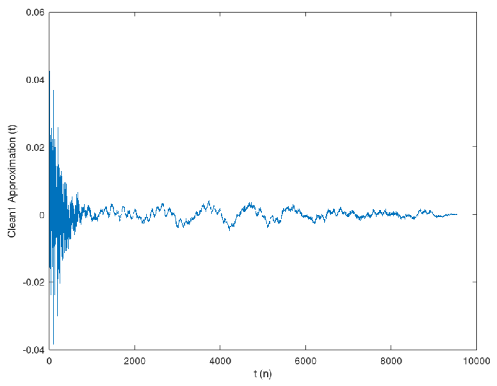
In Table 5.3, results of the testing are presented. As can be seen, the method overcame the baseline[4] for every room by applying the $Clean2$ or $Clean3$ phoneme onto the reverberated phoneme with the highest SNR. This means that the results of the method are numerically closer to the measured impulse response than reverberated phonemes. Also, results of the Phoneme Combination-based method were surpassed for each room.

	D105			L207			Q301		
	rev1	rev2	rev3	rev1	rev2	rev3	rev1	rev2	rev3
Baseline	0.084			0.089			0.087		
Clean1	0.080	0.093	0.067	0.077	0.085	0.078	0.090	0.093	0.093
Clean2	**0.060**	0.068	0.069	0.071	**0.067**	0.069	0.080	0.077	0.066
Clean3	0.067	0.065	0.064	**0.067**	0.087	0.089	0.074	0.078	**0.062**

Table 5.3: Results of the Clean Phoneme Deconvolution method.

Despite the success in overcoming the baseline, as results did not reach the range of $< 0.020; 0.024 >$, they are still not numerically suitable for usage in favour of others, measured impulse responses.

Even though the best numerical results were obtained by using phonemes $Clean2$ and $Clean3$, in further experiments $Clean1$ phoneme is used instead, as its results are visually and acoustically closer to referential RIR (see Figure 5.8).



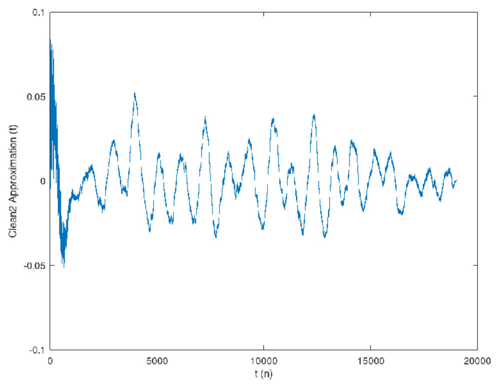


Figure 5.8: A comparison of results obtained when using $Clean1$ phoneme (left) and $Clean2$ phoneme (right).

[4]The baseline values are in table 5.3 slightly different from those in tables 5.1 and 5.2 since some of phonemes resembling the reference RIR the most had to be removed from data set since their clean forms were used as $c[n]$.

5.8 Testing of Combinations of Clean Phoneme Deconvolution Method and Phoneme Combination-based Method

Finally, we are going to use both earlier introduced methods at once and compare results with those of earlier tests. Two types of combination are performed. In the first one, the Clean Phoneme Deconvolution method is applied to every phoneme, leading to several estimated impulse responses, which are then combined using the Phoneme Combination-based method. In the second one, phonemes are combined using Phoneme Combination-based method and afterwards, the Clean Phoneme Deconvolution method is applied onto the result. The reverberated form of the $Clean1$ sample from Section 5.7 was removed from test sets.

For the testing, the same phonemes are used as in testing the phoneme combination-based method (Section 5.5). Frequency-Distance method is used for similarity measurement.

Results are presented in Table 5.4. In the table, we can see that results are numerically similar to results of the Phoneme Combination-based method. In most of cases, the variant of combination where the Phoneme Combination-based method was followed by the Clean Phoneme Deconvolution method resulted in numerically better results. No result reached the
$< 0.020; 0.024 >$ range.

	D105	L207	Q301
Baseline	0.071	**0.076**	**0.073**
Combination Deconvolution	**0.066**	0.094	0.082
Deconvolution Combination	0.076	0.081	0.087

Table 5.4: Resuls of combination of methods presented earlier in this thesis.

5.9 Listening Evaluation

To analyze outputs of methods presented above by listening, for each room eight estimates of room impulse responses and eight augmentations of a clean speech signal were generated. The augmentation was done using convolution and was done to allow comparison of speech, instead of impulse responses[5]. These estimated impulse responses consist of 3 estimates generated by the phoneme combination-based method using three lengths of Hann's function, 3 estimations generated using different clean phonemes in the clean phoneme deconvolution method and 2 estimates obtained by application of the two methods in a different order. For rooms D105 and L207, one extra augmentation was generated, obtained by clipping the impulse response calculated by using Phoneme Combination-based method using Hann's function of length 256 samples as the latter part of the RIR contained only noise.

For the Phoneme Combination-based method, impulse responses generated are impulsive sounds, however, followed by a strong noise (as it was shown in Figure 5.6). The impulses themselves also do not resemble the sound of impulse responses very much as they sound very artificial. When convolved with a clean signal, the augmented version contains

[5]As an impulse response is a "clap"-like sound, their comparison is not doable by listening.

very long reverberation time and artefacts. When the noise is cut away from the estimated response, the augmentation starts to resemble the original reverberation, by the length of the reverberation time, however, the sound itself is not very clear. Of all the used Hann's function lengths, the 256 samples long sounds the best as it creates the smoothest sound.

In case of the Clean Phoneme Deconvolution method, impulse responses generated using the $Clean1$ phoneme resemble an impulse, followed by an echo. Augmentations generated by this clean phoneme resemble the original signal the most for all cases, although in case of Q301, where the reverberation time was presumably shorter than the phoneme, the augmented version possess longer reverberation time than original. Also, in the case of D105, the augmented version sounds as if the signal was captured underwater. The $Clean2$ and $Clean3$ do not have impulse character and augmentations created using them resemble recording the original signal underwater with no reverberation resembling the room. None of the results has the clarity of the original reverberated signal.

Lastly, when combining the methods, the resulting estimates again sound like impulses followed by an echo. When methods are combined so the Clean Phoneme Deconvolution is applied first, the impulse end contains an artificially-sounding buzz. The speech augmented using these estimates also sound less clear than when the Phoneme Combination-based method is used first. When a signal is augmented using the estimate based on a combination of methods, where the Phoneme Combination-based method was used first, augmented version of signal in a room with short reverberation time (room Q301) contains deep humming sound, throughout the signal.

In the author's opinion, the best results were achieved, when using the $Clean1$ phoneme in the Clean Phoneme Deconvolution method and when a combination of methods was used, with Phoneme Combination-based method performed first as they have a similar level of reverberation and possess the clearest sound.

5.10 Summary

In this chapter, analytical procedures which would allow us to transform speech, particularly impulse-like phonemes, into impulse response were subject of our interest. Two possible approaches were suggested.

The first one, the Phoneme Combination-based method, uses averaging of magnitude spectrogram of aligned phonemes to eliminate noise and specifics of the speaker while keeping the phoneme's reverberation, which can be used as a room impulse response. Testing of this method, however, did not show its usability, as results did not for most of cases match or surpass the baseline – the highest measured resemblance between a reverberated phoneme, used in the combination process, and the measured room impulse response.

The second approach, the Clean Phoneme Deconvolution method, neglects specifics of speakers and deconvolves recorded phonemes using a different, prepared, clean form of the phoneme. Testing of the method did overcome the baseline, again given by the highest resemblance between reverberated phoneme and measured impulse response, however, its results were not sufficient to use this estimation instead of different measured impulse response as the resemblance did not reach the range of $< 0.020; 0.024 >$, separating responses of different rooms from impulse responses of the same room, with little deviations.

The third attempt to estimate the room impulse response was made using a combination of the two methods in both possible orders. The $< 0.020; 0.024 >$ range was still not met or surpassed.

Reasons for the inability to estimate room impulse response well enough to be on par with measured RIRs are possibly too large deviation of the impulse-like phoneme from a real impulse and too large diversity between forms of the phoneme, based on speaker and context. Also, the reverberated phonemes possess relatively low energy and are easily corrupted by background noise. This corruption apparently can not be sufficiently suppressed by combining several phonemes. For this low-energy reason, standard measurement techniques, discussed in Chapter 2, use long-lasting, highly energetic, signals which, when compressed into a short signal of a room impulse response avoid this background noise corruption.

These numerically huge deviations between the original impulse response and its estimations, however, do not necessarily mean that the estimations could not be used to improve ASR training to bridge the gap between clean and target data. Usage of these estimated impulse responses (possibly further modified by e.i. a denoiser) was, however, beyond the scope of this thesis.

Further tests in harsher environments with stronger reverberation and a higher level of noise were not performed as results of tests where the microphone was located close to the speaker (and low reverberation was present) were not sufficient and no ASR system was trained yet to show usability of the estimations.

Chapter 6

Machine Learning Approach to RIR Estimation from Speech

As it was shown in the previous chapter, combining phonemes to receive room impulse response produces results which resemble the real room impulse response to some point, but their similarity to reference ones, computed using the Frequency-Distance method, is still at levels corresponding to impulse responses of different rooms. Also in listening tests, some results sound similar to the original recordings but still are easily distinguishable. In this chapter, we try to blindly estimate room impulse response using deep neural networks. As no architecture for impulse response estimation is known to the author, several architectures are tried and their results are compared.

6.1 Principles

One of the reasons why methods in the previous chapter did not generate very precise results may be the limitation in input data used. From all the speech signals where every spoken word contains information about the reverberation of the room, only several carefully selected phonemes were extracted and used for the impulse response estimation. This situation changes when deep neural networks are used. When using DNNs, several seconds of continuous speech (or even whole signals) can be used without a need for any phoneme detection and measurement of its SNR. As we want to transform values representing speech into different values representing room impulse response, we will be using regression neural networks (see Figure 6.1).

Speech
Regression DNN
RIR

Figure 6.1: For transformation speech signal into a room impulse response, regression DNN has to be used.

Both input (speech) and output (RIR) can be represented in several forms. They can be represented in the time domain as arays of samples or in the frequency domain as spectrograms. As dealing with phases-containing spectrograms would be troublesome, magnitude, power and logarithmic spectrograms are considered as suitable.

Also, as speech and RIR signals tend to have high dynamics, its suppression should be considered, using normalization, to achieve better results.

6.2 System Setup

To create, train and test neural networks, Python3 framework PyTorch was used. PyTorch is an open-source machine learning framework allowing GPU acceleration[1]. Features of PyTorch's sub-package TorchAudio, as spectrogram generation or audio resampling, were used during the data sets generation[2].

An input used for all architectures tested in this chapter is a power spectrogram received from initial 4.016 seconds of an audio signal sampled at a frequency of 8000 kHz. This part of the audio signal is then transformed into a spectral domain by operating on 256 samples weighted by Hann function. This results in spectrogram with 129 bands and 251 frames. 0^{th} band, containing the DC component, is then removed as it is unnecessary for the impulse response estimation. The length of 4.016 s was chosen as it fills 251 frames of the spectrogram.

For training of models, Adam optimizer was used. Adam, firstly introduced in Diederik et al. [6], is an adaptive learning rate method, meaning that it computes individual learning rates for different parameters. The name itself stands for ADAptive Moment estimation. The initial learning rate is in all cases is set to 0.001 and ReduceLROnPlateau learning rate scheduler is used with *factor* set to 0.1 and *patience* set to 3. Every model training takes 30 epochs. The objective function used is the Mean Squared Error. A size of batch is set to 8.

[1]https://pytorch.org

[2]https://pytorch.org/audio/

6.3 Outputs and their Normalization

When values are presented to models, target impulse responses are normalized, based on their domain to suppress their high dynamics.

6.3.1 Signals

If a room impulse response signal in the time domain is the target of a model, it is first normalized by a result of an exponential regression calculated over averaged absolute values of all impulse responses contained in data set (see Section 6.4). The normalization is calculated as:

$$x[n] = \frac{h[n]}{y[n]}, \tag{6.1}$$

where $h[n]$ is the first 0.496 s of the RIR, and $y[n]$ is the result of the exponential regression. The result of exponential regression has form of

$$y[n] = ae^{bn}, \tag{6.2}$$

where a and b are parameters returned by the exponential regression. Result of such regression can be seen in Figure 6.2.

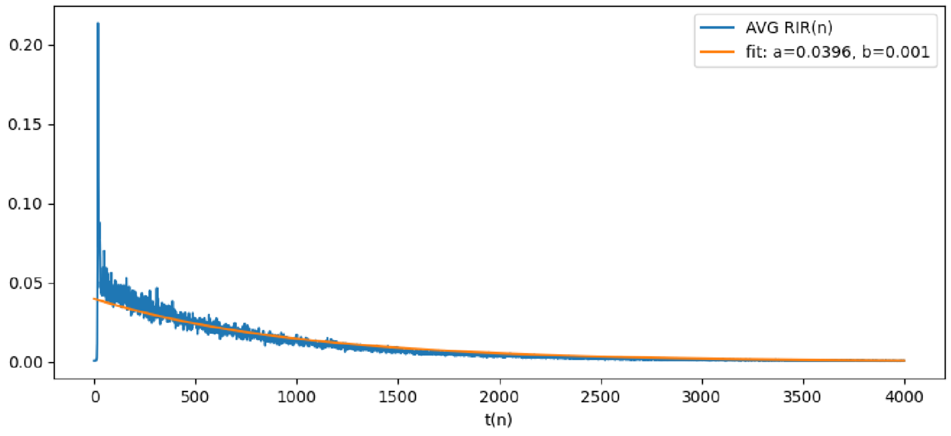


Figure 6.2: Averaged absolute values of room impulse responses and a result of an exponential regression.

The regression was done using a combination of script *RIRNormalization.py*, which searches for impulse responses in specified directory, averages their absolute values and exports them into a single *RIRavg.scv* file, which is then imported into Excel where the regression itself is performed using its scatter graph functions. Excel was used as *curve_fit* function contained in *scipy.optimize* Python3 package did not return sufficient results. Impulse response normalized this way in the time domain is displayed in Figure 6.3.

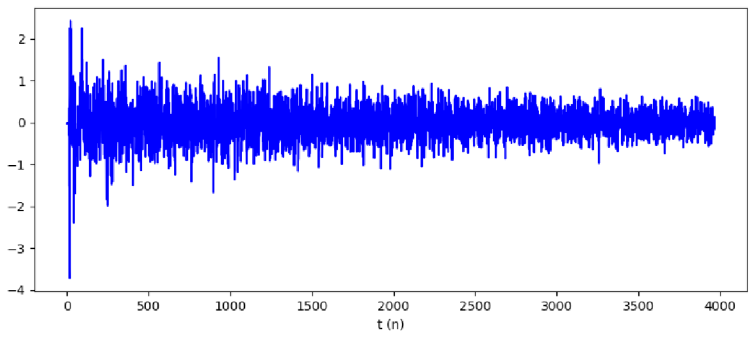


Figure 6.3: Room impulse response normalized using the exponential regression.

6.3.2 Spectrogram

When impulse response in the frequency domain is the target of a model in form of a spectrogram (of any kind, see Table 6.5), target spectrograms are normalized using mean and standard deviation values calculated from all impulse responses contained in data set. The normalization is computed as:

$$S[m,n] = (S[m,n] - MEAN_{RIRS})/STD_{RIRS}, \tag{6.3}$$

where $MEAN_{RIRS}$ and STD_{RIRS} denote mean and standard deviation values calculated from initial 0.496 s of all RIRs contained in data set. These RIRs were in forms of spectrograms according to the output form of the model. This means that i.e. power spectrograms have different $MEAN_{RIRS}$ than magnitude spectrograms.

The $MEAN_{RIRS}$ and STD_{RIRS} values were calculated using the *RIRNormalization.py* script, as it returns these values as well as averaged absolute values of RIRs.

6.3.3 Reconstruction of RIRs

When outputs of models are tested, they are un-normalized again. If a signal in the time domain is the output of a model, its restoration is calculated using the result of exponential regression:

$$h[n] = x[n] \times y[n]. \tag{6.4}$$

This will return the original dynamic to the signal. Similarly, when any kind of spectrogram is the output of a model, its dynamics is restored by re-applying mean and standard deviation values of RIRs in given form of spectrogram:

$$S[m,n] = S[m,n] \times STD_{RIRS} + MEAN_{RIRS}. \tag{6.5}$$

When the output of a model is in form of any spectrogram, the impulse response in the time domain, used for purpose of listening evaluation, is calculated using the Griffin-Lim [4] algorithm, which's implementation is a part of the TorchAudio library. The Griffin-Lim algorithm iteratively estimates phases for a magnitude spectrogram by minimalizing the mean squared error between Short-time Fourier transform of the estimation of signal and the provided magnitude spectrogram. Result of the Griffin-Lim algorithm achieved with 64 iterations can be seen in Figure 6.4.

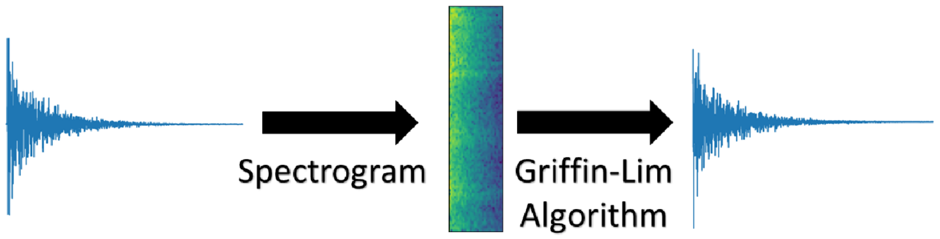


Figure 6.4: Restoration of phases from a magnitude spectrogram using the Griffin-Lim algorithm produces results similar to the original signal.

6.4 Data Sets And Their Generation

As a source of data for networks, the ReverbDB database was used. For training, validating and testing, microphones were selected manually, excluding ones which were partly or fully boxed and those with long reverberation times. After this exclusion, 105 microphones were left, each containing its room impulse response and its recording of LibriSpeech test-clean set. These microphones were then split into 3 non-overlapping groups: TRAINING, VALIDATING and TESTING with sizes of 83, 10 and 12 microphones. The composition of groups is shown in Table 6.1.

	TRAINING	VALIDATING	TESTING
D105	**03, 04, 05, 09, 10, 13, 19** 01, 07, 08, 11, 14, 15, 29	02, 06, 16	26
L207 SpkID_01	**01, 06, 07, 09, 12, 19, 22, 24** 02, 03, 04, 08, 18, 20, 21, 25	/	05, 23, 31
L207 SpkID_06	**03, 04, 06, 07, 08, 19, 20, 22** 01, 05, 09, 18, 21, 23, 25	02	31
L212 SpkID_01	**04, 06, 08, 13, 15, 19** 03, 05, 07, 10, 11, 14, 16, 24	01	02, 26, 31
L212 SpkID_02	**03, 05, 07, 08, 09, 15, 22, 24, 28** 01, 02, 06, 11, 14, 26, 31	04, 16	13
Q301	**04, 24** 06, 07, 09, 16, 19, 20	01, 02, 03	05, 08, 26

Table 6.1: Distribution of microphones into groups. The bold microphones stay in TRAIN group when test 6.6.2 is performed.

Data sets were then generated by randomly picking files from group corresponding to generated data set (i.e. TESTING group corresponds to test data set), checking whether the file's length is sufficient, resampling the file to 8 kHz, cutting off the part after the 4.016 seconds, transforming the signal into a spectrogram and removing the 0^{th} coefficient. The room impulse response of the microphone the file belongs to was then added to a different dimension of the tensor in which the spectrogram was located (so in [0, :, :] the room impulse response was located and in [1, :, :] the spectrogram was located). This tensor was then saved into a folder dedicated to the currently generated data set. After the whole data set was generated, an entry was added into a text descriptor, containing a path to the data set, its length and its mean value.

A data set generator used in this chapter is a part of this thesis. Data sets used in this chapter have a total length of 50000 samples if no other value is stated with 40500 samples being at training set, 4500 being at validation set and 5000 being at test set. As there are 105 microphones across data sets, one microphone was used to generate about 476 samples.

6.5 Models

Four architectures of neural networks were designed.

The first architecture is inspired by Zeinali's et al. [19] implementation of Snyder's et al. [14] architecture originally used for speaker verification, in which several changes were made and discussed, due to uncertainty of how the original architecture was implemented in Kaldi. This architecture is described in Table 6.2. The BandNorm layer type normalizes every band of input using mean and standard deviation value. StatisticsPooling calculates mean and standard deviation value of every input layer, creating linear output of length $2 \times$ number of input's layers. After every Hidden Layer, except the 1^{st} and the 7^{th}, a ReLu non-linearity is located. After non-linearities of Hidden Layers #2 #3 #4 #5 and #6, BatchNormalizations are located.

	Architecture 1
Layer Name	**Layer Type**
Input Layer	(128,251)
Hidden Layer #1	BandNorm(128)
Hidden Layer #2	1dConv(128, 512, 5)
Hidden Layer #3	1dConv(512, 512, 5)
Hidden Layer #4	1dConv(512, 512, 7)
Hidden Layer #5	1dConv(512, 512, 1)
Hidden Layer #6	1dConv(512, 1536, 1)
Hidden Layer #7	StatisticsPooling(1536, 3072)
Hidden Layer #8	Linear(3072, 3967)
Hidden Layer #9	Linear(3967, 3967)
Hidden Layer #10	Linear(3967, 3967)
Output Layer	See Table 6.5

Table 6.2: Architecture 1 scheme.

The second and third architectures are inspired by an autoencoder described by Novotný et al. [7]. The third model uses wider linear layers in order to test, whether wider layers may lead to better results. Architectures 2 and 3 are described in Table 6.3. After every Hidden Layer, except the 1st, a Tanh non-linearity is located.

	Architecture 2	**Architecture 3**
Layer Name	**Layer Type**	**Layer Type**
Input Layer	(128,251)	(128,251)
Hidden Layer #1	BandNorm(128)	BandNorm(128)
Hidden Layer #2	Linear(128*251, 1500)	Linear(128*251, 4096)
Hidden Layer #3	Linear(1500, 1500)	Linear(4096, 4096)
Hidden Layer #3	Linear(1500, 1500)	Linear(4096, 4096)
Output Layer	See Table 6.5	See Table 6.5

Table 6.3: Architectures 2 and 3 scheme.

The fourth architecture handles the input spectrogram as a figure, unlike previous models by application 2d convolution layers. Architecture 4 is described in Table 6.4. After every Hidden Layer, except 1st, 4th and the 7th, a ReLu non-linearity is located. After non-linearities of Hidden Layers #2 #3 #5 and #6, BatchNormalizations are located.

	Architecture 4
Layer Name	**Layer Type**
Input Layer	(128,251)
Hidden Layer #1	BandNorm(128)
Hidden Layer #2	2dConv(1, 6, 3)
Hidden Layer #3	2dConv(6, 6, 3)
Hidden Layer #4	MaxPool(2)
Hidden Layer #5	2dConv(11, 11, 5)
Hidden Layer #6	2dConv(11, 11, 5)
Hidden Layer #7	MaxPool(2)
Hidden Layer #8	Linear(16929, 8464)
Hidden Layer #9	Linear(8464, 3967)
Hidden Layer #10	Linear(8464, 3967)
Output Layer	See Table 6.5

Table 6.4: Architecture 4. scheme.

The output layer of architectures presented above depends on their required output. A list of possible forms of output and corresponding output layers is located in Table 6.5 As can be seen, when impulse response is generated in the time domain, its length is 3967 samples (0.4959 s). This is a sufficient time as only microphones with short reverberation time were chosen and hence T_{60} is located up to this time. 3967 samples are also enough to fill 31 frames of spectrogram with Hann's function length of 256, which is the size of spectrogram, produced by output layers producing spectrograms.

Output Layers	
Form of impulse response	**Layer Type**
Time Domain	Linear(X, 3967)
Magnitude Spectrogram	Linear(X, 129*31).abs()
Power Spectrogram	Linear(X, 129*31).abs()
Log. Mag. Spectrogram	Linear(X, 129*31)

Table 6.5: A list of output layers. X denotes number of outputs of previous layer and .abs() method calculating absolute value for every element in tensor.

6.6 Experiments

6.6.1 Networks with Different Outputs

In the first test, every network architecture proposed in Section 6.5 is trained for every form of output. An average difference between outputs of networks and the reference room impulse responses is measured using the Frequency-Distance method (FD). When resulting room impulse response is in the time domain, the average difference is also measured using the Time-Ratio method (TR). Also, an average difference between consecutive outputs (DIFF) is measured using the Frequency-Distance method to observe, whether tested model truly returns estimations based on inputs. The DIFF value of target results of testing set had the value of 0.084. As we know that the produced RIRs should be mostly different for consecutive outputs, a value significantly lower than 0.084 will, therefore, raise a suspicion about the distinctiveness of results. Results are displayed in Table 6.6.

	Time Domain			Magnitude Spectrogram		Power Spectrogram		Log. Mag Spectrogram	
	TR	FD	DIFF	FD	DIFF	FD	DIFF	FD	DIFF
Architecture 1	22.640	0.083	0.000	0.113	0.044	0.149	0.000	**0.063**	**0.042**
Architecture 2	**158.924**	**0.096**	**0.044**	0.130	0.000	0.154	0.001	0.084	0.005
Architecture 3	**87.072**	**0.093**	**0.031**	0.145	0.004	0.164	0.005	0.111	0.004
Architecture 4	22.197	0.084	0.000	0.127	0.043	0.146	0.000	**0.064**	**0.013**

Table 6.6: Comparison of models based on different architectures and with different outputs.

As can be seen, most of the models have a problem with overfitting as they return the same output with no or very distinctiveness. In case when outputs have form of a power spectrogram, this is happening for every architecture. This might be, in the author's opinion due to requiring too small values still possessing too high dynamics (despite the output normalization performed). However, the overfitting is not present for all models, which proves that wrongly selected training data are not the reason for this phenomenon.

Architectures 1 and 4-based models proved to be capable of outputting input-dependent results when outputs take form of magnitude or logarithmical magnitude spectrograms. While the magnitude outputs of the architectures do not visually resemble the required outputs very much as their later-time values are often zero, the logarithmical magnitude spectrograms, despite being ”smoother” than correct results, visually do resemble correct outputs. This can be seen in Fig 6.5. The architecture 1-based model outputting logarithmical magnitude spectrograms achieved the best score of all models tested with its average Frequency-Distance score of 0.063. As it did not reach the $< 0.020; 0.024 >$ range, results are still, numerically, at levels corresponding to different impulse responses. This average score is also close to the best result achieved using analytical methods, achieved by the Clean Phoneme Deconvolution method, where the *Clean*2 phoneme was used to deconvolve *rev*1 in room D105. It is, however, important to keep in mind that despite obtaining the best Frequency-Distance result, the listening evaluation showed that results with worse Frequency-Distance score may sound better in practice.

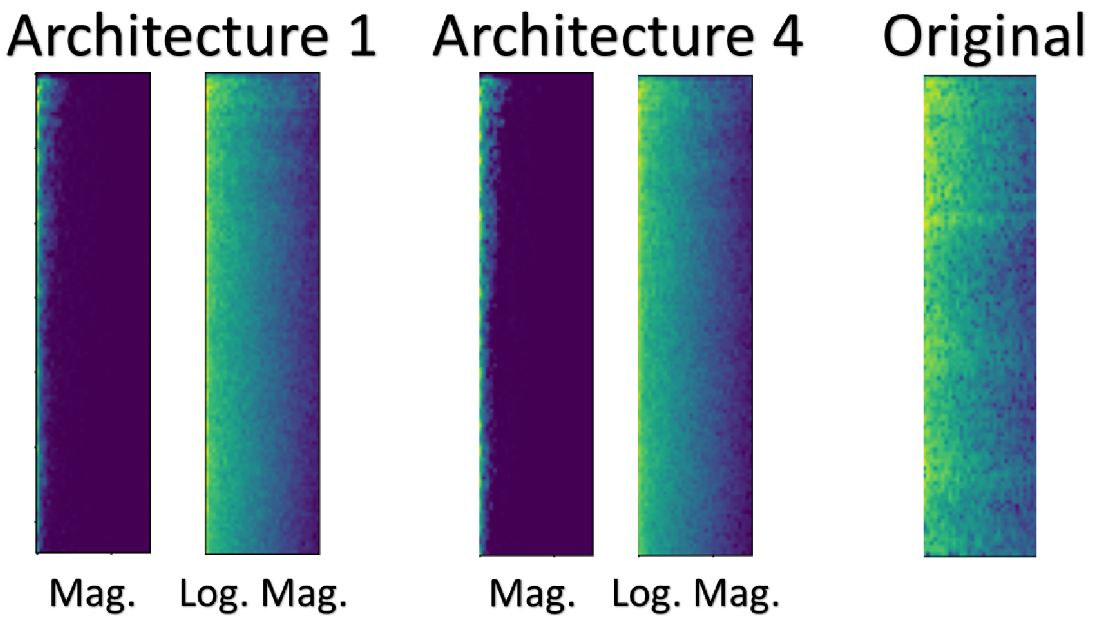


Figure 6.5: Comparison between outputs of architectures 1 and 4 when magnitude and logarithmical magnitude spectrogram is the output form. The Original denotes spectrogram of reference RIR.

Interestingly, as architectures 2 and 3-based models were not capable of producing input-dependent outputs of any form of a spectrogram, they do output input-dependent results in form of signal in the time domain. Spectrograms of these results visually also resemble the correct output, see Fig 6.6. Deviations measured using both TR and FD proved, that using wider linear layers, in this case, leads to results numerically closer to the correct ones. However, they are also less distinctive.

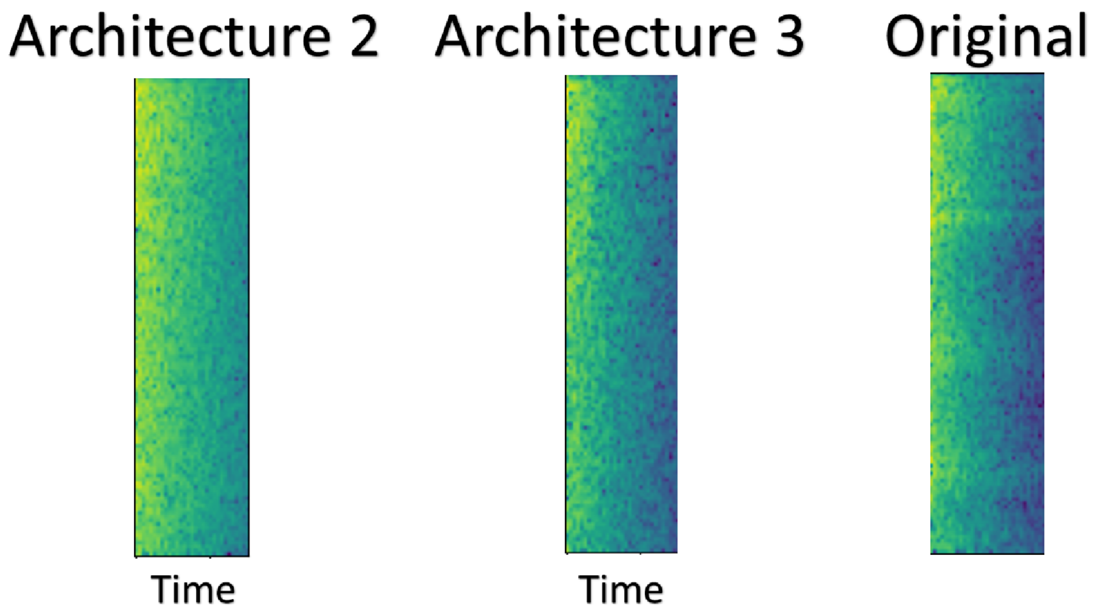


Figure 6.6: Comparison between spectrograms of architectures 2 and 3 when output has a form of RIR signal in the time domain. The Original denotes spectrogram of reference RIR.

This suitability of different architectures for different outputs may, in the author's opinion, be given by the fact, that architectures 1 and 4-based models, which are the most suitable for outputting logarithmic magnitude spectrograms do use convolution layers. These layers can make them more suitable for outputting figure - like outputs, such as spectrograms, especially the logarithmical magnitude ones, where a great range of values is on the output, while restricting them from variety when estimating one-dimensional outputs, such as the impulse response in time.

Architectures 2 and 3-based models, containing only linear layers do not have this restriction and hence might be more suitable for estimation of RIRs in time. This absence of convolution layer may then be the reason for the inability to estimate figure-like outputs.

Another possible reason for this behaviour could be the nonlinearity used, as architectures 1 and 4-based models contain ReLU while architectures 2 and 3-based models contain Tahn non-linearity.

6.6.2 Impact of Training Set Length And Amount of RIRs Onto Networks Performance

In the second test, we observe how results of networks change when the size of training set is increased and when the number of impulse responses located in training set is reduced.

By increasing the size of training set, we observe, whether a larger size of training set, not containing any new impulse responses, can improve results. Training set size is for this experiment increased from the initial size of 40500 to 70000 samples.

By reducing the number of impulse responses used for training, we want to measure how much is the performance of models influenced by the number of impulse responses in training set. Models with higher performance drop are more likely to perform better in cases when more impulse responses are added to the training set. The number of impulse responses is, in this experiment, reduced from the initial 83 microphones to 40. The size of training set remains 40500 samples.

Models used for this experiment are architecture 1-based model outputting magnitude spectrogram, architecture 1-based model outputting logarithmic magnitude spectrogram and architecture 3-based model outputting signal in the time domain. These architectures and their outputs forms were chosen as they provided the best results for given form of output while keeping the difference between its outputs above 0.030 on average. That is also the reason why no power spectrogram-generating model was used.

Results can be seen in Table 6.7. All values were measured using the Frequency-Distance method. A#X -> Mag. / Log. Mag. / Time denotes model based on architecture X with magnitude / logarithmic magnitude / time output.

	Original Result		70k Trainset Result		40 RIRs Result	
	FD	DIFF	FD	DIFF	FD	DIFF
A#1 -> Mag.	0.113	0.044	0.119	0.039	0.120	0.021
A#1 -> Log. Mag.	0.063	0.042	0.063	0.046	0.065	0.037
A#3 -> Time	0.063	0.031	0.093	0.020	0.098	0.029

Table 6.7: Results of increased training set size with no additional RIRs and reduced number of RIRs contained in training set.

It can be observed that use of a larger training set with no additional RIRs not only did not improve the accuracy of models (as in case of architecture 1-based model outputting logarithmical magnitude spectrograms), but it even decreased accuracy for other models. Larger training set also decreased distinctiveness of outputs, except for the architecture 1-based model outputting logarithmical magnitude spectrograms, where small improvement was measured. This may be caused, in the author's opinion, by the large number of inputs corresponding to the same output. Due to this, models may tend to output results heavily inspired by training data. Further research which would search for the optimal number of speech signals used for one provided RIR in training set is out of the scope of this thesis.

In case of reducing the number of RIRs provided in training set from 83 to 40, a decrease of accuracy and distinctiveness of outputs was observed for every model. Despite the biggest accuracy drop was measured in case of architecture 1-based model outputting magnitude spectrograms, the author does not think that provision of more training RIRs would dramatically increase its accuracy, due to its outputs, where later frames consist of zero-valued elements. For this reason, architecture 3-based model outputting the room impulse responses in the time domain has the highest potential for improvement in case of more RIRs were provided.

We have experimentally discovered that the number of impulse responses provided in the training set has a bigger positive impact on the performance of models than provision of more speech signals reverberated using the same set of RIRs. What is more, too large training set containing too few impulse responses may even cause accuracy and distinctiveness of results to drop.

6.7 Listening Evaluation

For purpose of listening evaluation of results, microphone 2 in room D105 and microphone 1 in rooms L207 (SpkID1) and Q301 (SpkID01) were chosen for RIR estimation, due to the previous estimation of their RIR in Section 5.9, which allows us to compare results of estimations for both approaches. Clean speech signal was augmented using the estimated RIR as listening only to 0.5 s long RIR estimation would not be sufficient for comparison.

The listening evaluation was done for models outputting logarithmical magnitude and magnitude spectrograms based on architectures 1 and 4 and models outputting RIR in the time domain, based on architectures 2 and 3. These models were chosen as distinctiveness of their results based on their input was observed in Table 6.6. As input of each model, random speech from the estimated microphone was used.

Results obtained from models outputting logarithmic magnitude spectrograms do resemble the correct RIR and the reverberated speech signal. The reverberation is on the same level as in the original reverberated recording. The model based on architecture 1 provides, in the author's opinion, clearer-sounding results than the model based on architecture 4, especially for the room Q301, where the reverberation time is shorter than in other rooms.

All models outputting magnitude spectrograms do have, in their augmented speech results, artificially-sounding echo, which is not located in the reference reverberated signal. Furthermore, for the architecture 4–based model, all results sound similar. This invariance of results is not present in the architecture 1-based model and also this model does not have the artificial-sounding echo as distinct as the architecture 4-based one.

Lastly, results of models outputting their estimation in the time domain, do not have any artificially-sounding artefacts. Their results are, however, very similar to each other in terms of reverberation time length and sound's cleanness.

Of all the models, the architecture 1-based model, outputting logarithmic magnitude spectrograms has its results the most resembling the original impulse responses. This highest resemblance is not only of all the DNN models but also of all the analytical methods tested in Chapter 5 as well. A comparison of room impulse response generated by this model with correct impulse response can be seen in Figure 6.7 on the next page.

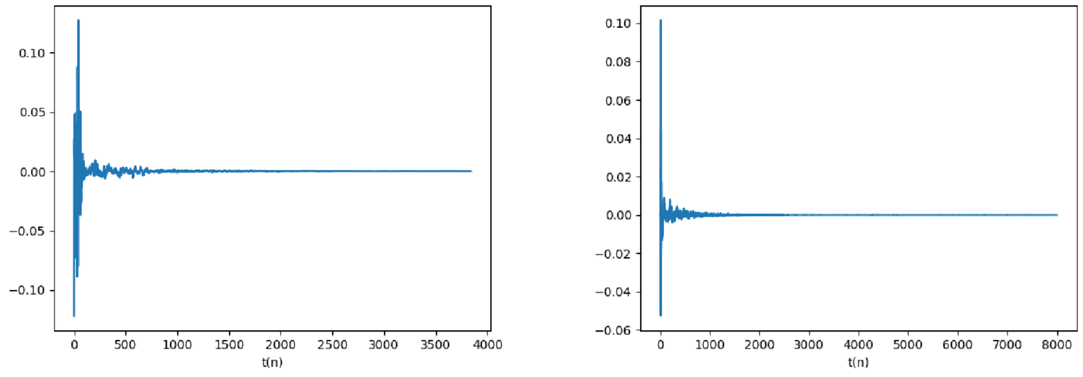


Figure 6.7: Comparison of an estimation of impulse response with short reverberation time (left) with reference RIR (right). The impulse response was estimated using architecture 1-based model outputting logarithmic magnitude spectrograms.

6.8 Summary

In this chapter, we tried to estimate room impulse responses from speech signals using deep neural networks using power spectrograms containing the first four seconds of reverberated speech as input.

Four different neural network architectures were proposed and each was implemented in four variants, with different forms of resulting RIR representation. These outputs are signal in the time domain, magnitude spectrogram, power spectrogram and logarithmic magnitude spectrogram.

Of all networks, the numerically best results were achieved when using architecture inspired by Snyder, outputting logarithmical magnitude spectrogram. Despite its results, it, however, did not match or surpass the $< 0.020; 0.024 >$ range of Frequency-Distance method separating two measurements of the same impulse response and two different impulse responses. During listening evaluation, the model produced estimations closest to original responses. However, as no testing was performed on an ASR, the effect of its results on WER are unknown.

Finally, we have experimentally discovered that increased size of training set, not containing any new impulse responses not only does not improve results of neural networks but even may cause their accuracy to drop and decrease distictiveness of their results. For accuracy and result-distinctiveness improvement, additional room impulse responses in training set are required.

Chapter 7

Conclusion

7.1 Summary

In this thesis, we have introduced the reader to the subject of reverberation, which can be represented as a room impulse response. We also presented methods used for the room impulse response measurement (i.e. Maximum Length Sequence method or Exponential Sine Sweep method). Methods estimating the impulse response only from a mathematical model of the room were mentioned as well (i.e. Ray-Tracing method).

Further, we have introduced two methods designed for comparison of measured impulse responses. The Time-Ratio method, comparing signals in the time domain, and Frequency-Distance method, operating in the frequency domain. The functionality of both methods was successfully checked.

The Time-Ratio method was used in chapter 4, where we have replicated some of results from the ReverbDB database. This was done using the Exponential Sine Sweep method. We have observed differences between the author's implementation of the method and implementation used in the ReverbDB. Using the Time-Ratio method we have proved that both implementations return similar results.

Next, we have tried to estimate the impulse response of a room using methods processing impulse-like phonemes. The first method proposed, the Phoneme Combination-based method, combines magnitude spectrograms of phonemes to get rid of the uncorrelated sound. The second one, the Clean Phoneme Deconvolution method, neglects the speaker-specifics and deconvolves reverberated phonemes with prepared clean phonemes. Combination of the methods was also tested. Despite the results did not numerically succeed (similarity of results with original responses was measured using the Frequency-Distance method), listening test have shown that several results were similar to the original reverberated speech. As the results were not used for training of an ASR, we cannot tell, how they would affect its performance.

Lastly, deep neural networks were used for impulse response estimation as they can use continuous speech instead of extracted phonemes. Four architectures were proposed and trained for several forms of outputs. The best results were achieved with an architecture inspired by Snyder outputting logarithmic magnitude spectrograms. Despite numerically failing to resemble reference RIR, listening tests have revealed similarity of augmented speech to genuine reverberated one. This way, we have proved suitability of neural networks for estimations of room impulse responses.

7.2 Short-Term Perespective

As analytical methods presented in this thesis showed us, estimation of impulse response only from small windows of speech does not bring any huge success in terms of creating estimations of RIRs undistinguishable from the measured ones. In the author's opinion, neural networks are promising to achieve numerically and acoustically accurate RIR estimations. Testing current architectures with different forms of input could lead to better results. Also, models could be trained to estimate larger reverberation times in case of success on short reverberations. New architectures can be tested as well as already proposed architectures used for a different purpose, i.e. embeddings-using architectures. The first thing that should be done, however, is training an ASR system on data augmented by RIRs obtained from methods/models presented in this thesis to objectively evaluate their results.

7.3 Long-Term Perespective

From a broader point of view, as the RIR estimation's main goal is to augment data for an ASR to perform better during far-field speech recognition, experiments could be made which would aim to transfer reverberated speech into its clean form, instead. If a neural network was able to do this, no expensive microphone arrays would be needed for this task. Also, an experimental speech recognizer could be trained, which would use a combination of data from a camera and a microphone to improve its performance.

Bibliography

[1] Aoshima, N. Computer-generated pulse signal applied for sound measurement. *Journal of the Acoustical Society of America*. 1981, vol. 69, no. 5, p. 1484–1488. ISSN NA.

[2] Dunn, C. and Hawksford, M. O. Distortion immunity of MLS-derived impulse response measurements. *AES: Journal of the Audio Engineering Society*. 1993, vol. 41, no. 5, p. 314–335. ISSN 00047554.

[3] Farina, A. Simultaneous Measurement of Impulse Response and Distortion With a Swept- Sine Technique. *Proc. AES 108th conv, Paris, France*. 2000, vol. 5133, I, p. 1–15.

[4] Griffin, D. and Jae Lim. Signal estimation from modified short-time Fourier transform. *IEEE Transactions on Acoustics, Speech, and Signal Processing*. 1984, vol. 32, no. 2, p. 236–243.

[5] Hamici, Z. Real-time pattern recognition using circular cross-correlation: A robot vision system. *International Journal of Robotics and Automation*. 2006, vol. 21, no. 3, p. 174–182. ISSN 08268185.

[6] Kingma, D. P. and Ba, J. L. Adam: A method for stochastic optimization. In: *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*. 2015.

[7] Novotný, O., Plchot, O., Glembek, O., Černocký, J. and Burget, L. Analysis of DNN Speech Signal Enhancement for Robust Speaker Recognition. *Computer Speech and Language*. 2019, vol. 2019, no. 58, p. 403–421. Available at: https://www.fit.vut.cz/research/publication/12039. ISSN 0885-2308.

[8] NTI-Audio. *Reverberation Time RT60 Measurement*. 2020. [Online; accessed 27-March-2020]. Available at: https://www.nti-audio.com/en/applications/room-building-acoustics/reverberation-time-rt60-measurement.

[9] Panayotov, V., Chen, G., Povey, D. and Khudanpur, S. Librispeech: An ASR corpus based on public domain audio books. *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*. 2015, 2015-August, p. 5206–5210. ISSN 15206149.

[10] Paul, D. B. and Baker, J. M. The Design for the Wall Street Journal-Based CSR Corpus. In: *Proceedings of the Workshop on Speech and Natural Language*. USA: Association for Computational Linguistics, 1992, p. 357–362. HLT '91. Available at: https://doi.org/10.3115/1075527.1075614. ISBN 1558602720.

[11] Ravanelli, M. and Omologo, M. On the selection of the impulse responses for distant-speech recognition based on contaminated speech training. *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*. 2014, October, p. 1028–1032. ISSN 19909772.

[12] Ravanelli, M., Svaizer, P. and Omologo, M. Realistic multi-microphone data simulation for distant speech recognition. *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*. 2016, 08-12-September-2016, no. 1, p. 2786–2790. ISSN 19909772.

[13] Schroeder, M. R. Integrated-impulse method measuring sound decay without using impulses. *Journal of the Acoustical Society of America*. 1979, vol. 66, no. 2, p. 497–500.

[14] Snyder, D., Garcia-Romero, D., Sell, G., Povey, D. and Khudanpur, S. X-Vectors: Robust DNN Embeddings for Speaker Recognition. In: *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2018, p. 5329–5333.

[15] Stan, G. B. Comparison of different impulse response measurement techniques. *AES: Journal of the Audio Engineering Society*. 2002, vol. 50, no. 4, p. 249–262. ISSN 00047554.

[16] Svedström, T. *Tuote-1: Ultrasonic Echo Chamber*. Dissertation. Available at: http://urn.fi/URN:NBN:fi:aalto-201605262132.

[17] Szoke, I., Skacel, M., Mosner, L., Paliesek, J. and Cernocky, J. Building and evaluation of a real room impulse response dataset. *IEEE Journal of Selected Topics in Signal Processing*. May 2019, vol. 13, no. 4, p. 863–876. ISSN 1932-4553.

[18] Yoshioka, T., Sehr, A., Delcroix, M., Kinoshita, K., Maas, R. et al. Making machines understand us in reverberant rooms: Robustness against reverberation for automatic speech recognition. *IEEE Signal Processing Magazine*. 2012.

[19] Zeinali, H., Burget, L., Rohdin, A. J., Stafylakis, T. and Černocký, J. How To Improve Your Speaker Embeddings Extractor in Generic Toolkits. In: *Proceedings of ICASSP 2019*. IEEE Signal Processing Society, 2019, p. 6141–6145. Available at: https://www.fit.vut.cz/research/publication/12037. ISBN 978-1-5386-4658-8.