CONCATENATIVE CROWD NOISE SYNTHESIS

- A C++ IMPLEMENTATION USING THE VALENCE-AROUSAL MODEL

Master thesis by Christian Knörzer

[redacted]

Supervision: Henrik von Coler

Reviewer: Prof. Dr. Stefan Weinzierl, Henrik von Coler

Technische Universität Berlin

Fakultät I - Geisteswissenschaften

Fachgebiet Audiokommunikation

STATUTORY DECLARATION

Hiermit erkläre ich an Eides statt gegenüber der Fakultät I der Technischen Universität Berlin, dass die vorliegende, dieser Erklärung angefügte Arbeit selbstständig und nur unter Zuhilfenahme der im Literaturverzeichnis genannten Quellen und Hilfsmittel angefertigt wurde. Alle Stellen der Arbeit, die anderen Werken dem Wortlaut oder dem Sinn nach entnommen wurden, sind kenntlich gemacht. Ich reiche die Arbeit erstmals als Prüfungsleistung ein. Ich versichere, dass diese Arbeit oder wesentliche Teile dieser Arbeit nicht bereits dem Leistungserwerb in einer anderen Lehrveranstaltung zugrunde lagen.

_____ _____

Ort, Datum (Christian Knörzer)

ACKNOWLEDGEMENT

I would like to thank my tutor Henrik von Coler and my professor Prof. Stefan Weinzierl for their subject-specific support, as well as my family and Marion for their moral and mental support.

I. ABSTRACT

The goal of this master thesis is to implement a C++ algorithm that synthesizes the noise of a talking, human crowd and to evaluate the performance of the algorithm for different affective states within a two-dimensional valence-/arousal plane. The algorithm uses a concatenative synthesis with grains from speech recordings in an anechoic environment.

To create the grains automatically, five different syllable segmentation algorithms are implemented as MATLAB functions and compared to a manual syllable segmentation; the algorithm closest to the manual segmentation is chosen for the creation of the corpus of the concatenative synthesis.

Three different models for the handling of the selection and processing of affective states are presented: *Model A1* and *A2* use grains that are mapped to a position within the two-dimensional *affective space*. The mapping is computed manually for *Model A1* and automatically with a combined regression- and classification model for *Model A2*. For both models, a position within the affective space and a range to determine the grains used for the concatenation can be selected. *Model B* selects grains during runtime according to acoustic parameters derived from the selected position within the affective space.

The output streams of the C++ algorithm of the three models implemented as VST3-plugins were convolved with binaural impulse responses and then presented to a group of 50 people who evaluated the position of the perceived affective state within the valence-/arousal plane.

Results show significant[1], medium to high correlations between the (affective) position input values of the plugins and the subjects' evaluations for both affective dimensions of *Model A1* and *Model A2* and for the arousal dimension of *Model B*. *Model B* fails to generate crowd noises with affective states of perceivably different valence values due to the lack of a sufficient correlation of acoustic parameters with the valence.

Several improvements and approaches for future research are suggested.

[1] For a significance level of 0.01

ZUSAMMENFASSUNG

Ziel dieser Masterarbeit ist die Implementierung eines C++-Synthesealgorithmus zur Generierung von (Hintergrund-)Geräuschen einer sprechenden Menschenmenge mit anpassbarem, affektivem Zustand. Die Auswahl des affektiven Zustandes erfolgt über ein zweidimensionales Koordinatensystem mit der Dimension der Valenz (von negativ bis positiv) auf der x-Achse und der Dimension der Erregtheit (von ruhig bis erregt) auf der y-Achse. Die Synthese geschieht mittels einer konkatenativen Synthese mit silbenähnlichen Audioeinheiten, welche mit einem MATLAB-Skript automatisch aus im schalltoten Raum aufgenommenen Sprachaufnahmen generiert werden.
Für die Generierung werden die automatisch generierten Audioeinheiten von fünf verschiedenen MATLAB-Skripten mit manuell erstellten Silbengrenzen verglichen. Der *Silbengenerator*, der der manuellen Segmentierung an nächsten kommt, wird für die Erstellung des sogenannten Synthese-Korpus verwendet.

Drei verschiedene Modelle für die Auswahl und Generierung der affektiven Zustände werden implementiert: Bei *Modell A1* und *A2* wird den Audioeinheiten eine feste Position im gegebenen zweidimensionalen affektiven Raum zugewiesen. Dies geschieht bei *Modell A1* manuell, bei *Modell A2* automatisiert durch die Kombinierung eines Regressions- und Klassifizierungsmodells. Ausgehend von dem gewählten Auswahlbereich im Koordinatensystem verwenden beide Modelle nur Audioeinheiten für die Synthese, die innerhalb des Bereiches liegen. Bei *Modell B* wählt der Algorithmus zur Ausführungszeit Audioeinheiten aus, die aufgrund von akustischen Merkmalen zur derzeit ausgewählten Position im Koordinatensystem passen.

Die Ausgangssignale der VST3-Plugins, die diese Modelle umsetzen, wurden binaural gefaltet und 50 Probanden präsentiert, welche die Stimuli hinsichtlich der wahrgenommenen Position im zweidimensionalen affektiven Raum bewerteten.

Die Ergebnisse zeigen signifikante[2], mittelstarke bis starke Korrelationen zwischen bei der Geräuschgenerierung als Eingangsparameter für den Algorithmus gewählten, affektiven Positionen und wahrgenommenen affektiven Positionen für beide Dimensionen bei *Modell A1* und *A2*, sowie für die Dimension der Erregung für *Modell B*. Da (derzeit noch) keine bekannten, ausreichenden Korrelationen zwischen der Valenz-Dimension und akustischen Parametern existieren, kann der Algorithmus nach *Modell B* keine in der Valenz-Dimension klar unterscheidbare Signale erzeugen.

Eine Vielzahl von Möglichkeiten zur Verbesserung und Erweiterung der beschriebenen Synthese und zur weiteren Erforschung des Themengebietes wird erläutert.

[2] Bei einem Signifikanzniveau von 0,01

II. DEFINITIONS AND ABBREVIATIONS

Affect	"A set of observable manifestations of a subjectively experienced emotion" (Merriam-Webster, 2017a)
Affective position	A position in the affective space, here mostly within the two-dimensional valence-arousal plane
Corpus	The ensemble of the grains together with data describing or classifying the grains to be used in a concatenative synthesis
DAW	Digital Audio Workstation
Emotion	Here: Also used as synonym for affective state
F0, f_0	Fundamental frequency
F1	First formant
FFT	Fast-Fourier-Transformation
Fricative	"a consonant characterized by frictional passage of the expired breath through a narrowing at some point in the vocal tract" (Merriam-Webster, 2017b)
Grain	Small piece of audio data as used in granular synthesis or concatenative synthesis
GUI	Graphical User Interface
Host	Software that provides the opportunity to be extended by plugins and to exchange data between the host and the plugin
HRTF	Head-related transfer function
NaN	Not a number, invalid result
OSC	Open Sound Control
Pitch	Used here mostly as synonym for fundamental frequency
Plugin	Software component that extents the functionalities of another software (of the host). Cannot be executed without the host.
RMS	Root Mean Square
Signal-to-noise ratio (SNR)	Ratio of signal power to noise power
Soundscape	Acoustic analogy to the visual landscape, the entirety of sounds in a certain environment (Wulff, 2012)
Syllable	A C++ object that contains the audio data of one grain and Meta data about that grain
syllable	Not necessarily a syllable in the traditional sense, but comprises also syllable-like grains

Thread	Sequential process in programming “that share[s] memory” (Lee, 2006, p. 2)
TTS	Text-to-speech
Valence-arousal plane	A two-dimensional affective space described by the x-coordinate of valence and the y-coordinate of arousal
VUV	voiced/unvoiced

III. TABLE OF CONTENTS

SYNTHESIS OF AFFECTIVE CROWD NOISE

1. Introduction

When it comes to the simulation of noise of a human crowd, sound designers of games, virtual reality applications or developers of audio installations tend to use long, prerecorded sounds (e.g. the so called *Wallas*[3]) as a basis for their soundscapes.[4]

This implicates several disadvantages, as those sound files have fixed characteristics:

First of all, a prerecorded sound has a fixed length. Any use that exceeds this length leads to the necessity of loops, resulting in audible repetitions. The number of available channels is also determined, restricting the sound designer in his choice of files or leading to the necessity of *up-* or *down-mixing*.

The perhaps most striking disadvantage, however, is the fact, that sound files of human crowds mostly contain the audible room response of the room or environment they were recorded in. These files can therefore only be used in scenarios that take place in rooms or environments whose audio characteristics are sufficiently similar to those of the recording location, as the multiplication of two different room responses can lead to a rather unrealistic sound.

Those complications leave the sound designer the choice between investing time in finding a suitable *sound texture* (for a definition see chapter 2.2.2) file or the more elaborate solution of conducting their own recordings.

But even with custom human crowd recordings, a real-time user interaction can only be implemented to a very limited extent, e.g. with a level control for different sound files. As a consequence of using many sound files (that are sufficiently long to avoid repetitions) to achieve a possible user interaction, more memory space is required.

With sound texture synthesis, a *dry*[5] signal of infinite length can be generated for as many channels as needed using only a limited amount of stored data and enabling a real-time adaption of the texture sound to events like user interaction.

For game sound design or interactive audio installations, a synthesized crowd noise that changes its *mood* or affective state according to the user's actions could intensify the user experience, since humans are able to correctly identify the expressed affective state within speech signals even without understanding the content of the speech itself (Scherer, 1995, p. 237).

The hereby presented master thesis describes and discusses the implementation of a C++-application, which synthesizes the textural noise of a human crowd with a selectable affective state using a concatenative, corpus-based, granular synthesis on the basis of syllables or *syllable-like* grains.

[3] The term *Walla* describes an audio file containing background noise in form of a talking or murmuring human crowd. It is used in movies, radio and television to set a mood (Carlsson, n.d.).
[4] See e.g. the statement in Appendix B
[5] In this context, *dry* states the absence or negligible presence of room reflections

Chapter 2 gives an overview on the topics related to this work and presents previous related work. Chapter 3 describes, how the speech material is *prepared* for the synthesis algorithm and chapter 4 explains, how audio *grains* are concatenated within the synthesis. The audio output of the synthesis is evaluated in a listening test described in chapter 5. Results of the listening test as well as general methods used for the synthesis are discussed in chapter 6.

2. Background

A synthesis of affective crowd noise combines the fields of emotion research with the fields of speech and texture synthesis. The following subchapters are dedicated to providing some background information on each of the fields and offer the basis for the decision to implement the synthesis as a concatenative synthesis with a two-dimensional valence-arousal coordinate system as affective model. The final subchapter gives an overview on previous work that is either directly linked to the synthesis of crowd noise or allows a deduction of methods for the here presented work.

2.1 The valence-arousal model of affect

The possibility to select affective states demands for a predefinition on how the algorithm itself distinguishes affective states and which kind of handles are offered to the user to select an affective state.

Psychologists tended to describe emotion or affect in terms of individual dimensions of affect (as sadness, anger, tension or fear) up until the middle of the last century. While the theory of emotion used in psychiatric and neuroscience research still mostly assigns a limited and discrete number of independent basic emotional dimensions to humans (Posner, Russell, & Peterson, 2005, p. 2), Schlosberg (1952) placed different states of affect based on facial expressions on an oval according to a two-dimensional scale. The longer axis of the oval represented a pleasantness-unpleasantness dimension, the shorter axis an attention-rejection dimension (Schlosberg, 1952). Two years later, he included an activation theory of emotion by Lindsley (1951, as cited in Candland, 2003, p. 174) to his research of facial expressions and suggested a three-dimensional model of affect.

Russell (1980) discarded the Schlosberg's *attention-rejection*-dimension, placing affective states on a circle within a "two-dimensional bipolar space" (Russell, 1980, p. 1162). Affective states are placed according to their degree of valence as horizontal dimension and arousal as vertical dimension. His *Circumplex Model of Affect* with 28 affective states can be seen in Figure 1.

This two-dimensional model of valence and arousal was chosen to serve as a basis for the *affective space* used both for processing the audio source material and as a way for the user to determine the emotional output of the synthesis, since it seemingly offers a promising compromise between affective diversity and practicability.

The term *affective position*, whenever used in the following chapters, implies a position within this two-dimensional *valence-arousal-plane*.

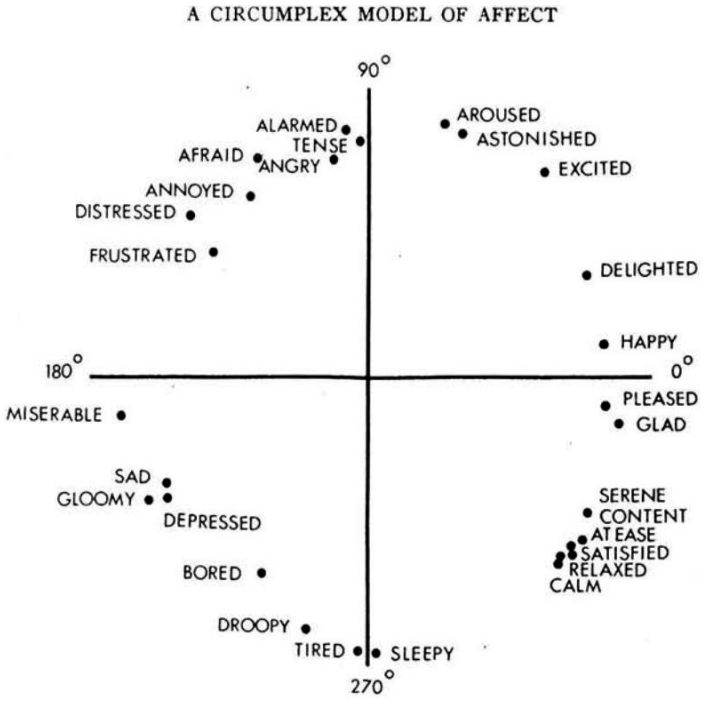


Figure 1: Russell's Circumplex Model of Affect (from Russell, 1980, Figure 2)

2.2 Synthesis

The output of the C++-algorithm can be described as speech texture sounds. The following chapters therefore provide an insight on the topics of speech and sound texture synthesis.

2.2.1 Speech synthesis

Speech synthesis has been a widely explored field within the last decades. Researchers concentrated particularly on the translation of written texts into acoustic speech. Consequently, numerous scientific papers, programs and application extensions dedicated to the topic of the so called TTS-systems (short for *text-to-speech*) are available today.[6]

The first step of transferring a text into speech is the translation of the text into the correct phonetic counterparts. As for background crowd noises the content of what is said is not of interest, this linguistic analysis step will not be explained any further here. A detailed explanation with examples for the German language can be found in Pfister & Kaufmann (2017, Chapter 8), further information on computer linguistics in general can be found in Carstensen et al. (2010, Chapter 3.3.2).

The second step, the translation of the phonetic symbols into speech can roughly be separated into two different synthesis methods: The concatenative synthesis and the parametric synthesis.

2.2.1.1 Concatenative synthesis

The concatenative synthesis or data-driven synthesis is the most frequently used synthesis method today (Pfister & Kaufmann, 2017, p. 246) and is based on the reassembling of beforehand recorded, segmented and labelled *units*. In TTS-systems, one *unit* refers to a speech

[6] An example for a TTS-system, an implementation of the Web Speech API created by Matt West, can be tried out on CodePen: https://codepen.io/matt-west/pen/wGzuJ (as of January 18)

segment and can consist of an allophone[7], a diphone[8], a triphone[9], a syllable, a word, a part of a phrase or a whole phrase, although the last two can also be categorized as mere speech replay systems, in which previously recorded utterances of a speaker are replayed. The essential difference for the distinction would be the fact that a speech replay system is limited to the recorded material of sentences, whereas a TTS synthesis model should be able to transfer every text into an audio speech signal (Pfister & Kaufmann, 2017, p. 27).

The entirety of the speech units together with their acoustical or higher-level descriptors builds the so-called *corpus* of the concatenative synthesis.

The basic concept is to find the speech unit that fits the next phonetic symbol and is the best approximation for a natural prosodic progression. For concatenative synthesis in general, this "selection is performed according to the descriptors of the units, which are characteristics extracted from the source sounds, or higher level descriptors attributed to them" (IRCAM, 2012, sec. Principle). Applied to speech units, the *higher level descriptors* would correspond to the phonetic classification, whereas *characteristics extracted from the source* would be for example pitch and the RMS values.

Depending on the size of the corpus, the selected speech units might have to be adapted in duration, fundamental frequency or loudness to fit to the previous units. Applying these processes to the speech units is computationally much more expensive than it is for the formant synthesis (see 2.2.1.2.1) and it can only be done within certain limits to avoid unnatural results or artefacts. The best case in terms of the naturalness would be if no adjustments were necessary. This can be achieved by storing variances of the same unit, which in turn increases the size of the corpus. The amount of adjustments decreases also with the size of the speech units, since the number of connection points between units decreases. However, bigger speech units imply a bigger corpus as well (Lemmetty, 1999, p. 33). The decision on a unit size is therefore always a trade-off between data size and quality.
Today's TTS-systems usually rely on a combination of small speech units like diphones and bigger units (e.g. for the more frequently needed words). The smaller units are only used when no *good fit* of the bigger units can be found within the corpus, e.g. for personal names (Pfister & Kaufmann, 2017, p. 247).

The limitations in the *parametric control* during the synthesis make it difficult to add effects like different speaking styles or affective states to speech units (Murray, Edgington, Campion, & Lynn, 2000, p. 173). One approach for that is the RP-PSOLA-system, developed by Vine and Sahandi (2000). However, their paper focuses on the examination of perceived distortions of the synthesis system rather than examining the impact of the presented system on the emotion

[7] *Allophones* are the phonetic interpretations of a phoneme (Carstensen et al., 2010, p. 180). Replacing one allophone by another doesn't change the meaning of a word, but the sound. Example: The phoneme /t/ has (amongst others) the allophones [t^h] as in *torch* and [t] as in *stop*.
[8] A *diphone* is the speech unit that contains the speech signal from one *core* of a phone or allophone to the next *core*. The pure concatenation of phones or allophones leads to jumps at the transition points or unnatural transitions between the *cores*. This can be avoided by the use of diphones. Examples for TTS-systems that rely only on diphones can be seen or listened to on the website of the MBROLA project (MBROLA Project Development Team, 1999)
[9] A *triphone* is a speech unit that consists of a *core* sound in the middle and transitions towards other core sounds on the left and right side of the *core*, thereby taking into account the phonetic context of the *core* (Pfister & Kaufmann, 2017, p. 393).

recognition.
For results with a natural sound, it is evidentially better, if the speech units already contain the required speaking style or affective state. If this is the case, the concatenative synthesis can deliver a synthesized speech output with a high level of naturalness and can preserve characteristics of the original speaker of the speech units.

One example for a system that is based on this method is *Talkapillar*, an expressive TTS-system developed by Beller, Hueber, Schwarz & Rodet (2006).

2.2.1.2 Parametric synthesis

A parametric synthesis creates artificial speech through the variation of the parameters of a speech creation model (Eichner & Wolff, 2001, sec. Beschreibung). Two sub classes of the parametric synthesis, namely the *formant synthesis* and the *articulatory synthesis*, will be discussed in the following sections.

2.2.1.2.1Formant Synthesis

The term *formant* describes a distinctive local maximum in the spectrum of a speech signal, which is the result of a resonance in the vocal tract. When speaking, these resonances are adapted by the articulators' positions (e.g. the tongue or the lips) to form the specific sound that people connect – for example – to a vowel. The formant synthesis imitates theses resonances by filters of second order (Pfister & Kaufmann, 2017, p. 14).

As a first step of the formant synthesis, a signal generator, e.g. consisting of an impulse generator for voiced parts and a noise generator for unvoiced parts, creates an excitation signal with a selected fundamental frequency for the voiced part. This signal is then filtered with the formant filters. Most formant synthesizes use five of these filters (Pfister & Kaufmann, 2017, p. 243).

For the most part, a speech signal is a signal undergoing a constant change. Speech can be for few moments almost stationary, but never reaches a real stationary state – real stationary sounds are perceived as technical sounds, like the horn of a car (Pfister & Kaufmann, 2017, p. 243). As a consequence, to synthesize a natural speech sound, the excitation signal and the formant filters have to change constantly, too.

The parameters that are changed can be the fundamental frequency or the ratio between the unvoiced and the voiced part of the excitation signal, the formant frequencies and amplitudes, or the "intensity of a low- and high-frequency region"(Lemmetty, 1999, p. 29).

However, it is extremely difficult to model the parametric changes from one speech sound to another according to the changes of the vocal tract resonances, which differ from language to language and from speaker to speaker. To model the parametric changes in a more natural way, researchers make use of HMMs [10] (Hidden-Markov-Models) or nowadays also *Neural Networks,* which were previously trained on speech recordings. Yet most implementations of the formant synthesis sound understandable but very unnatural in comparison to today's concatenative synthesis implementations.

Nevertheless, the formant synthesis offers some evident advantages: As for the creation of a specific sound only the parametric settings have to be stored, the data consumption of the

[10] For detailed information on HMMs in speech synthesis, see e.g. Tokuda et al. (2013)

formant analysis is very small compared to the concatenative synthesis. The fact that no human speech recordings are involved in the synthesis of the sound (the *analysis* part for the parametrization left aside) makes the formant synthesis also a very flexible solution. The implementation of changes of pitch, *speech rate*[11] or *voice quality*[12] is much less complicated as with a data-driven synthesis, since it implies only the change of one or several parameters.

These benefits could make the formant synthesis a promising approach for the implementation of an affective speech synthesis. "One can control voice quality, pitch, intensity, spectral energy distributions, harmonics-to-noise ratio or articulatory precision which allows modeling many co-articulation effects occurring in emotional speech" (Oudeyer, 2003, p. 161). Yet, the deficits in terms of naturalness are up until now still too big to consider this form of synthesis for the here presented crowd noise synthesis.

An example for a TTS-system based on formant synthesis is *eSpeak*, an open source speech synthesizer created by Duddington, Avison, Dunn & Vitolins (2017).

2.2.1.2.2Articulatory synthesis

"When speaking, the vocal tract muscles cause articulators to move and change shape of the vocal tract which causes different sounds" (Lemmetty, 1999, p. 28).

The *Articulatory Synthesis* is an approach of speech synthesis that generates speech signals by modelling the human articulatory system and its movements (Pfister & Kaufmann, 2017, p. 240).

Kröger & Birkholz (2009, p. 307f) provide a sub classification of this approach.

They present a *Vocal Tract Model*, which calculates the geometries of the complete vocal tract. The position and shape of every vocal tract organ (including e.g. lips, tongue and nasal cavity) are remodeled and changed over time to create speech signals. Parameters for the vocal tract modelling are e.g. derived from MRI- or X-Ray-Scans or from sensors attached to the vocal tract (Pfister & Kaufmann, 2017, p. 240).
Another sub classification presented by Kröger & Birkholz (2009, p. 307f) is the *Acoustic Model*, which models the air flow and the air pressure distribution within and around the human body. A subglottal air flow in combination with a specific lung pressure is altered over time by a *tube model*, which consists of several *tube sections* imitating the trachea, pharynx, nasal, and oral cavities. Those tube sections possess distinctive aerodynamic and acoustic characteristics described by the tube model.

According to Lemmetty (1999, p. 28) the *Articulatory Synthesis* theoretically has the highest potential to deliver high-quality synthetic speech. However, the computation of this model is both computationally expensive and much more complicated than other methods. The movements of the tongue alone are highly complicated and therefore almost impossible to model (Lemmetty, 1999, p. 29). Furthermore, the retrieval of data for the parametrization of the models through MRI-scans or sensor information is time-consuming, expensive and somewhat unpleasant for the person being examined, which is why implementations of an *Articulatory*

[11] The *speech rate* is a term to describe the *speed* of the speech and denotes usually the number of speech units per time unit
[12] The *voice quality* comprises spectral features of the speech signal, e.g. the *spectral slope*

Synthesis have not been able to deliver results that are comparable to a concatenative synthesis regarding the naturalness of the produced speech signal (Pfister & Kaufmann, 2017, p. 240f).

Given the characteristics of all the different synthesis methods, the concatenative synthesis was chosen as approach for the synthesis of a talking, human crowd.

2.2.2 Texture synthesis

The term *sound texture* represents an audio medium that consists of (altered repetitions of) smaller units. These smaller units are often extracted from an original (recorded or synthesized) sound file and (re-)synthesized "using a sequence of extracted building patterns"(Lu, Wenyin, & Zhang, 2004, p. 156). The sound texture is normally an attempt to create a sound that is perceived as similar to or ideally identical with the original sound (Möhlmann, 2011, p. 16). Sound textures should not contain audible repetitions or artefacts and should sound natural (Strobl, Eckel, Rocchesso, & Le Grazie, 2006, p. 4). Schwarz (2011, p. 221) defines a sound texture by the following five points:

> 1. Sound textures are formed of basic sound elements, or atoms;
> 2. atoms occur according to a higher-level pattern, which can be periodic, random, or both;
> 3. the high-level characteristics must remain the same over long time periods (which implies that there can be no complex message);
> 4. the high-level pattern must be completely exposed within a few seconds ("attention span");
> 5. high-level randomness is also acceptable, as long as there are enough occurrences within the attention span to make a good example of the random properties.
>
> (Schwarz, 2011, p. 221)

Schwarz (2011, p. 221) visualizes the consequences of point 3 and 4 in a graph shown in Figure 2. It illustrates, that unlike speech or music, after a certain amount of time, a sound texture does not provide any new information to the listener. This moment, however, arrives later than it does for pure noise. Thus, the potential information content of a sound texture approaches a constant value which is higher than that of noise.

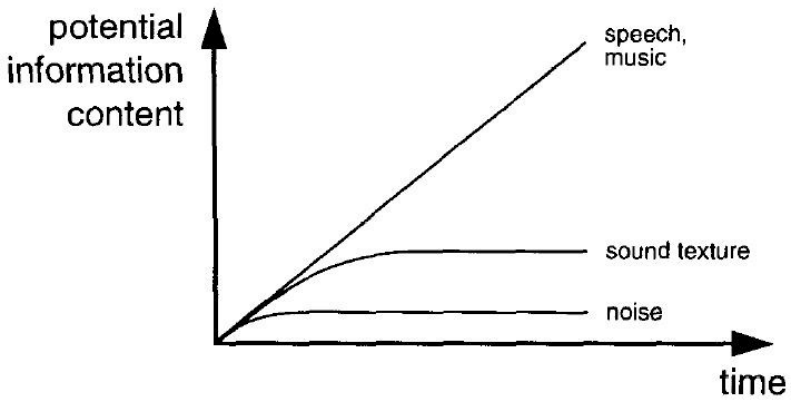


Figure 2: Information content over time for different sound categories (from Schwarz, 2011, p. 221)

There are different ways and models to approach the synthesis of texture sounds. Schwarz (2011, p. 223) distinguishes between signal models and physical models. The *corpus-based*

concatenative synthesis presented in 2.2.1.1 as a speech synthesis method is also a common model to create textural sounds. It can be classified as an extended type of the granular synthesis and is therefore a (physically informed) signal model. An overview over the classification can be seen in Schwarz's (2011) publication.

Regarding *simpler* textural sounds (e.g. fire, waves or rain), sound synthesis is already a common tool for sound design, not so much for movies as for sound installations or virtual reality installations and games, in which the duration of the *soundscape* is not predefined.

2.3 Previous related work

2.3.1 Orchestra of speech

The *Orchestra of Speech* is a research project by Daniel Formo as a part of the Norwegian Artistic Research Program at the Music Department of Norwegian University for Science and Technology in Trondheim (Formo, 2013a). It covers the extraction of musical features or "prosodic traits" (Formo, 2013a) from speech in order to recombine speech segments in a musical way to create electroacoustic music. The main goal of the project is to provide a tool or instrument allowing a musical use of speech in an improvisational manner. The instrument is implemented as a standalone patch in Max/MSP (Formo, 2014). Especially his work on syllable extraction from speech signals is of interest for the here presented work.

2.3.2 CataRT

CataRT is a concatenative software synthesizer by the IRCAM, which creates its corpus from a given audio file and selectable audio features. It is implemented as Max/MSP and is available as a standalone version or as modular system for Max/MSP (Schwarz, 2017).

A loaded audio file is split up into segments of a selected size and a set of features (e.g. audio features as spectral centroid, pitch, loudness, file-related features as position in the original file or higher-level descriptors imported by text files) is extracted from the resulting grains. The grains are then displayed as dots in a window according to two features chosen as dimensions. A third dimension can be displayed with the color of the dots.

Via a graphical user interface and the mouse or an external controller, a position in the two-dimensional space as well as a circular window region around the middle position are selected. According to the selected trigger mode, the playback of a grain within the selected window is triggered on certain events (e.g. a mouse move or a metronome beat) (Schwarz, 2010, sec. Trigger Modes). Several parameters like crossfade, attack, release or transposition can be set by the user. A more detailed overview on the functions of CataRT is provided by Schwarz (2010).

The software is intended to be a tool for *explorative granular synthesis*, *gesture-controlled synthesis*, *audio-controlled synthesis*, as a *data-driven drumbox* or for *expressive speech synthesis* (Schwarz, Beller, Verbrugghe, & Britton, 2006, p. 5). The expressive speech synthesis is, however, seemingly meant in an experimental way and not in a way to add emotions to a natural sounding speech.

2.3.3 MyNoise.net Background Noise Generator

MyNoise.net offers several sound generators as a browser application allowing to mix and play back background soundscapes of different categories. The soundscapes are meant to be used to boost concentration or creativity by masking unwanted background noise[13], to mask a tinnitus, as a relaxation aid, for meditation or for sound therapy. Every soundscape contains ten channels. Every channel contains sounds with a different spectral *focus* (Pigeon, 2013c).[14] The user can thus adapt the spectral distribution of the soundscape via the channels in a way that masks the unwanted background sounds best or to get the mix that suits best his purposes.

Apart from nature sounds, industrial soundscapes or city noise, human crowd noises referred to as *Babble Noise* are also available. According to Pigeon (2013b), these Babble Noises are an ideal way to mask confidential conversations, e.g. in offices. A *Babble Noise* consists of a mix of speech recordings that were treated in a way to mask the words that are spoken. Some words are still understandable, but only if you specifically pay attention to them.

The output of the ten channels is generated in runtime by playing back HTML5 audio files (Pigeon, 2013a). The audio files mainly have a length of several seconds and are played in random order with a dynamically changed panning. The Babble Noise Generator does not make use of anechoic speech samples. The voice recordings contain room information, which is not consistent over the different recordings.

Unfortunately, Pigeon (2013a) does not provide any details on the algorithms used. It is therefore not evident, if the selection of speech segments follows pure randomness or if higher-level descriptors or audio features are involved in the generation.

2.3.4 Parametric crowd noise examination

The project run by Grimaldi, Böhm, Weinzierl & von Coler (2017) in the course of an internship at the Technical University of Berlin examines the perception of a parametric synthesis of human crowd noise in virtual environments. The crowd noise for Grimaldi's studies is created by a Pure-Data-synthesizer concatenating grains of a corpus, which was specifically recorded for this application.

The recordings took place in the anechoic chamber of the TU Berlin to avoid an audible room information within the corpus. Two groups of five students (five male and five female students in total) were placed on seats arranged in a circle facing each other. Five cardioid microphones, each of them pointing at one seat, were placed in the center of the circle. The distance between the students and their respective microphone was approximately 1.5 m to avoid an elevation of low frequencies due to the proximity effect (Grimaldi, 2017, p. 2).
The students were first recorded unsupervised for about 45 minutes, then they were asked to act out scenarios with the intention of provoking speech of different affective states. Five of those scenarios were:

[13] *Noise* is meant here in the colloquial way, representing a sound or several sounds that are perceived as annoying
[14] The channels are not divided up into different frequency regions as a multi-band equalizer would do it. The sounds are distributed amongst the ten channels according to their spectral features from *low sounds* to *high sounds*. It is not specified, if the distribution is based on an acoustic parameter like the spectral centroid or only based on perception of the author.

- There is a fire in your apartment. Call the fire fighters.
- You see a group of young people damaging your bike from the distance. Make them stop.
- The Police stopped you for riding your bike without lights. Try to talk your way out.
- Ask your friends on the phone to help you move, tomorrow.
- Your kid's pet died. How would you tell it?
  (Grimaldi, 2017, p. 2f)

The recordings were split into utterances and sorted by speaker and by the three different excitement classes *calm*, *neutral*, and *excited* (Grimaldi, 2017, p. 3). An RMS and pitch value were also extracted from the utterances, but as the excitement classification was executed manually on the basis of the played scenario, the paper is not clear on how these values were used. Each utterance was segmented into syllable estimates by an algorithm developed by Härmä (2003). Rejecting syllables with a length of less than 80 ms or more than 1000 ms, the remaining syllables form the corpus for the synthesis.

Grimaldi (2017, p. 3) uses a speech model, which concatenates random syllables of one speaker to an audio stream. The random selection, which avoids immediate repetitions of one syllable, as well as the playback of the grains are computed by Pure Data objects (Grimaldi, 2016, p. 27). Grimaldi multiplies each syllable with a Gaussian envelope to get smooth transitions between the syllables. He uses a metronome with a speed of one beat per second as a trigger to deactivate and reactivate the concatenation of syllables for one stream to include *longer* pauses to the streams. The probabilities for the deactivation-decisions as well as the inclusion of smaller pauses (with a length of approximately 250 ms) are not further specified.

Grimaldi (2017, p. 5) places different numbers (16, 32 and 96) of streams (representing different sizes of crowds), which include the three different excitement levels, on varying circle positions around the listener using the *SoundScape Renderer* by Geier, Ahrens & Spors (2008). This way he generates 45 different binaural stimuli, which are presented to 30 subjects in a controlled environment via headphones.

He then compares the effects of the excitement levels and the number of streams on the perceived naturalness of the stimuli.

Grimaldi's results show a significantly lower perceived naturalness for the two higher number of streams compared to the perceived naturalness of the smallest number of streams. For the two larger stream numbers, the perceived naturalness for the excited crowd is significantly higher than for the neutral or the quiet crowd. For the smallest stream number, all three excitement levels are perceived as relatively natural. The estimated size of the crowd is bigger than the actual stream number. This could be due to the fact, that in a real human crowd, not everybody is talking all the time.

Two main artefacts of the synthesis are mentioned: Perceived repetitions of long grains and unnatural jumps of the RMS values. The first artefact is proposed to be solved by the improvement of the syllable segmentation and the creation of a bigger corpus. To diminish the second artefact, it is suggested to consider the RMS values of the previous grain in the selection of the new grain.

The here presented master thesis can be regarded as extension of Grimaldi's (2017) work by adding the valence-arousal model as a handle to set the affective state of the crowd and by implementing the synthesis algorithm as a C++-plugin, which can be used with numerous hosts on several platforms.

3. Corpus-Creation

To enable a real-time concatenation of syllables, the synthesis algorithm needs a pre-prepared *corpus*, which includes the audio data of every syllable as well as descriptive information on the syllable. In the case of the hereby presented work, this descriptive information has to enable the synthesis of a specific, selected, affective state within the valence-arousal coordinates. Chapter 3.1 therefore discusses the way of linking syllables to affective states.

The speech material for the creation of the corpus was taken from the recordings made by Grimaldi et al. (2017) for the project described in section 2.3.4. Syllables or syllable-like grains have to be extracted from these recordings. Chapter 3.2 thus addresses an automated syllable segmentation.

3.1 Emotion detection

Before starting the creation of a corpus or programming the algorithm, a decision on the way syllables are stored and connected to *affective positions* on the valence-arousal plane has to be made. Two options are theoretically conceivable:

Option *A*: The syllables are directly linked to a position within the valence-arousal plane, which means, that each syllable from the corpus can only be used for one *affective position*. Since single syllables can hardly be evaluated regarding their affective state, the evaluation has to be executed with utterances rather than syllables. Syllables within one utterance would be placed on the same position.

Option *B*: The syllables contain certain acoustic features that correlate with the dimensions of the valence-arousal plane and a *syllable-picking algorithm* collects during runtime all the syllables within the corpus that correspond to the acoustic features the selected affective state demands for.

As Schröder (2001, p. 2) states, recognition rates for different synthesized emotions are better, if the whole corpus is directly linked to an affective state rather than picking corresponding units out of one corpus based on prosody rules. This would speak for option *A*.

However, option *B* has the advantage, that one syllable could potentially be used for different affective states, possibly providing a bigger variety of syllables for each affective state. Obviously, option *B* can only be implemented if one or more acoustic features (or a combination of them) of the syllables themselves or the progression of acoustic features within the utterances correlate with the valence- and the arousal-dimension.[15] Such a correlation could also allow for an automated syllable placement on the valence-arousal plane for *Option A*.

Listening to recordings of utterances of different affective states but without meaningful content (e.g. numbers read by actors in different affective ways), humans are able to correctly identify the expressed affective state with an accuracy of about 50 %, which is about four or five times higher than the guessing probability (Scherer, 1995, p. 237). A correlation with affective states (but not necessarily with the affective dimensions of the chosen model) has to exist *somehow*. The following chapter contains a literature review on studies that examined such a correlation.

[15] These can of course be different features or feature combinations for the two dimensions.

3.1.1 State of research

Numerous studies throughout the last decades were dedicated to the topic of emotion recognition. The majority of them focused on the discrimination of distinctive affective states, mostly including *anger*, *happiness* or *joy*, *fear*, *sadness* and *neutrality* or *calmness*.

Scherer (1995, p. 241) summed up common results of studies examining a correlation of acoustic parameters and affective states up until 1995:
According to him, the affective state of *anger* is generally connected to an increase of the mean fundamental frequency f_0, the mean intensity and the high-frequency energy in comparison to neutral speech. The f_0-contours are directed downwards.
Fear is connected to an increased mean f_0, to a higher range of f_0 and to an increased articulation rate.[16] For some studies, *worry* or *anxiety* as "weaker forms of fear" (Scherer, 1995, p. 241) are also connected to an increased mean f_0.
Sadness is connected to a decreased mean f_0 and f_0 range and a decrease in the mean intensity, high frequency energy and articulation rate. For the most part, f_0-contours are directed downwards.
Joy is generally connected to an increased mean f_0, f_0 range, f_0 variability, mean intensity and articulation rate.

Kienast & Sendlmeier (2000) extended the set of examined affective states by *boredom*. They used three phrases spoken by six actors. They extracted parameters describing a segment deletion rate, formant shifts, and a *spectral balance* of fricatives in comparison to fricatives in neutral speech. All fricatives in utterances expressing sadness and boredom had a negative *spectral balance* value compared to the one in neutral utterances, while happiness, fear and anger lead to positive *spectral balance* values. Kienast & Sendlmeier's (2000) examinations showed a positive and high segment deletion rate for fear, boredom and sadness, a low positive rate for happiness and a negative rate (meaning segment insertions) for anger.

Goudbeek & Scherer's (2008) studies on 17 emotions confirm Scherer's (1995) results regarding fundamental frequency and intensity. The examination of their corpus also revealed significant correlations on several parameters connected to spectral characteristics, such as the *Hammarberg Index*[17] or the intensity below 1 kHz.

As the performance of computers increased drastically within the 1990s, researchers started to include machine learning algorithms in the implementation of automated *emotion classifiers*. These studies consist of a parameter extraction from recorded, spontaneously acted out or read speech and the use of the obtained parameters to train one or several machine learning algorithms:
With a "jack-knifing procedure", Scherer (1996, p. 2) explored the performance of subsets of 29 acoustic parameters in the classification of 224 affective vocal cues which were spoken by professional actors. The recordings were assigned to 14 different emotion classes and all 29 parameters were extracted from them. The vocal cues were then automatically classified by calculating the mean *parameter profile* for every class (not using the cue that was being classified) and assigning it to the class, for which the sum of squared differences between the

[16] The term *articulation rate* is used here as a measure for the speed of speech
[17] In this case the „difference between the energy maxima in the 0-2 kHz and 2-5 kHz range" (Goudbeek & Scherer, 2008, p. 2)

cue's profile and the classes profile was lowest. The algorithm changed randomly the subset of parameters keeping parameters that provide a higher percentage of correct classifications, until no further improvement was achieved. With "an overall hit rate of 40.4% (as compared to 7% expected by chance)", Scherer (1996, p. 2) achieves the best score with the following subset of parameters:

> Fundamental frequency: Mean, standard deviation, 25th percentile, 75th percentile; Energy: Mean; Speech rate: duration of articulation periods, Bands in the voiced long term average spectrum: 125-200 Hz, 200-300 Hz, 500-600 Hz, 1000-1600 Hz, 5000-8000 Hz; Hammarberg index; slope of spectral energy above 1000 Hz; proportion of voiced energy up to 1000 Hz. Bands in the unvoiced long term average spectrum: 125-250 Hz, 5000-8000 Hz (Scherer, 1996, p. 2)

The five affective states *fear*, *anger*, *happiness*, *sadness* and *neutrality* are subject to studies made by Petrushin (1999) and McGilloway et al. (2000):
Petrushin (1999) lists 14 features as the strongest of a set of 43 features he extracted from 700 utterances consisting of four sentences:

> F0 maximum, F0 standard deviation, F0 range, F0 mean, BW1 mean, BW2 mean, energy standard deviation, speaking rate, F0 slope, F1 maximum, energy maximum, energy range, F2 range, and F1 range. (Petrushin, 1999, p. 4)

He achieves the best results with an *Ensemble of Neural Networks* classifier with the lowest score for fear (35-53%), the highest score for sadness (73 – 83%) and an average score of 70%.
McGilloway et al. (2000) examined 32 different acoustic features from the ASSESS system[18] of five text passages read by 40 subjects. Apart from the parameter *energy below 250 Hz*, the parameters were all related to the progressions of fundamental frequency and intensity over time. McGilloway et al. (2000, p. 5) achieved the best score on the test set with a *Linear Discriminant* classifier, the recognition rate being 52.3 %.
Oudeyer (2003, Chapter 5) achieved very high recognition rates (up to 96%) training 19 different classifiers with the extracted parameters from a database of 4800 utterances for the four affective states joy/pleasure, sorrow/sadness/grief, anger, normal/neutral. The acoustic features used were "max, min, median, 3rd quartile and 1st quartile of low-passed signal intensity, pitch and minima of unfiltered signal intensity"(Oudeyer, 2003, p. 176).

As Cowie et al. (2001, p. 52) points out in a summary on the topic, there are some parameters (especially the ones related to pitch or intensity), that have repeatedly been validated across studies to have an impact on the perception of affective content of speech. However, other parameters have led to contradictive results: The affective states of *anger*, *fear* and *happiness* have both been assigned to faster and slower speech rates (R. Cowie et al., 2001, p. 52).

Other studies, describing an *analysis-by-synthesis* approach, conducted an implementation of a system to generate affective (speech) signals, which were then evaluated by an *emotion discrimination* by humans:

[18] ASSESS stands for "Automatic Statistical Summary of Elementary Speech Structures" (Roddy Cowie, Douglas-Cowie, & Sawey, 1995) and is a "system for semi-automatic analysis of acoustic speech parameters. […] It generates a simplified core representation of the speech signal based mainly on the F0 and intensity contours." (Schröder, 2005)

Scherer & Oshinsky (1977) synthesized a saw tooth signal, which was modified with different pitches, amplitudes, envelopes, speeds, number of harmonics and cutoff frequencies. All in all they presented 188 *cues* to 48 students who attempted to rate the perceived affective content of the cues. The links between the acoustic parameters and a perceived affective state or dimension which were rated as significant are listed in Table 1. Although connections between auditory stimuli and affect are shown, it does not necessarily mean, that the results can also be directly assigned to speech.

Table 1: Significant affective states or dimensions associated with acoustic parameters as presented by Scherer & Oshinsky (1977, p. 339)

Acoustic parameters of tone sequence	**Direction of effect**	**Emotion rating scales listed in decreasing order of associative strength**
Amplitude variation	Small	Happiness, pleasantness, activity
	Large	Fear
Pitch variation	Small	Disgust, anger, fear, boredom
	Large	Happiness, pleasantness, activity, surprise
Pitch contour	Down	Boredom, pleasantness, sadness
	Up	Fear, surprise, anger, potency
Pitch level	Low	Boredom, pleasantness, sadness
	High	Surprise, potency, anger, fear, activity
Tempo	Slow	Sadness, boredom, disgust
	Fast	Activity, surprise, happiness, pleasantness, potency, fear, anger
Envelope	Round	Disgust, sadness, fear, boredom, potency
	Sharp	Pleasantness, happiness, surprise, activity
Filtration cutoff (number of harmonics)	Intermediate (few)	Pleasantness, boredom, happiness, sadness
Level (number of harmonics)	High (many)	Potency, anger, disgust, fear, activity, surprise

Burkhardt & Sendlmeier (2000) used the parameters *mean pitch*, *pitch range*, *pitch variation*, *pitch contour of the phrase*, *pitch contour of the syllable*, *f_0-flutter*, *intensity of syllables*, *speech rate*, *phonation type*, *vowel precision* and *lip-spreading* to alter a formant synthesis generating affective utterances in German. While they chose an utterance with a neutral meaning,[19] the intended affective states of the generated utterances through parameter modulation were

[19] The phrase was "'An den Wochenenden bin ich jetzt immer nach Hause gefahren und habe Agnes besucht'(At the weekends I always drove home and visited Agnes)" (Burkhardt & Sendlmeier, 2000, p. 2)

hot/cold anger[20]*, happiness/joy*, *crying despair*, *quiet sorrow*, *fear* and *boredom*. In a first experiment, the output of different settings of the first five parameters was assigned to the perceived affective states by 30 native German-speaking participants to obtain the best setting for each affective state. Burkhardt & Sendlmeier then further adapted the resulting settings for each state and presented them in several versions per emotion to 42 subjects, who were again asked to assign the heard utterances to an affective state. The received recognition rates were between 29 % (for hot anger) and 81 % (for joy).

Breazeal (2002) designed for her *sociable robot* both an affective recognition and speech system. Her robot was able to distinguish between *approval*, *prohibition*, *comfort*, *attention* and *neutral* speech and responds (in the same order) pleased, sad, content, interested or calm. The recognition consisted of two stages: A low-level pitch and energy extraction and a high-level prosodic feature extraction. Breazeal achieved very high recognition rates of between 40% and 87% for utterances with a strong affect and with speakers other than those the robot was trained with.

Oudeyer (2003, Chapter 3) generated "cartoon emotional speech" (Oudeyer, 2003, p. 160) expressing *happiness*, *anger*, *sadness*, *comfort* and *calmness* with the freeware speech synthesizer MBROLA.[21] Recognition rates were between 38% and 76%.

Across these studies, subjects were always asked to identify the perceived affective state by assigning an affective state from a list of possibilities to the perceived stimulus. As Schröder (2004, p. 95) points out, this procedure "corresponds rather to a discrimination task than an identification task, especially when the number of categories involved is small." It is evident, that the recognition rates of Scherer (1996) are not only less than half as good as the ones presented by Oudeyer (2003, Chapter 5), because Scherer didn't use the same mathematically advanced machine learning algorithms or different parameters, but also because his algorithm discriminated amongst 14 affective states as opposed to the four examined by Oudeyer.
Another important remark might be, that "a forced choice test provides no information about the quality of the stimulus in terms of naturalness or believability." (Schröder, 2004, p. 95)

Although the examined affective states of the previously mentioned studies can be converted to positions on a valence-arousal scale,[22] the findings of the studies regarding the respective parameter profiles are difficult to translate to the two-dimensional emotion model. The translation would necessarily be based on numerous assumptions.[23] However, to be able to illustrate an affective state corresponding to a position within the valence/arousal-plane, a continuous parameter adaption over both dimension has to be specified.

It is therefore necessary to examine the acoustic parameters regarding a possible direct correlation with the two dimensions themselves.

[20] *Hot* and *cold anger* can be discriminated by a different level of arousal: *Hot anger* or rage can be "characterized by the outward display of anger by means of gestures, facial and verbal (e.g., cursing) expressions, hostile aggressive behaviour" (Biassoni, Balzarotti, Giamporcaro, & Ciceri, 2016, p. 2), *cold anger* such as irritation comprises "milder and more subtle forms of anger" (Biassoni et al., 2016, p. 2).
[21] More information on the MBROLA project on http://tcts fpms.ac.be/synthesis/mbrola html
[22] The corresponding positions are listed e.g. by Schröder et al. (2001, p. 89)
[23] E.g. regarding the selected set of emotion *profiles*, the mere assumption, that in between the placed profiles, parameter values can be interpolated, the way of interpolation or the behaviour of parameter values towards *extreme* areas, where no parameter profile was examined.

Scherer & Oshinsky (1977) set their results in relation to the affective dimensions *activation*, *pleasantness* and *strength*. They observe a "consistent relationship between the auditory cues of pitch, loudness, rate, and timbre and the subjectively rated activity level"(Scherer & Oshinsky, 1977, p. 335), but not for the pleasantness or strength dimension.
Scherer later confirms this statement: "There has been relatively little evidence for the vocal differentiation of individual emotions on other dimensions such as valence" (Scherer, 1995, p. 241).

Pareira (2000) calculates Pearson's linear correlation coefficients for 40 utterances placed on the three-dimensional model of *activation*, *evaluation* and *power*. Her study reveals significant correlations of the parameters *mean of* f_0, f_0 *range* and *mean of RMS* with the activation dimension. For the evaluation dimension, the *mean of* f_0 shows a weak correlation (at a 0.05 level), but only for female speakers.

Cowie et al. (2001, p. 52f) connect positive activation to an increase in *mean* and *range of* f_0 and a *tense* voice quality, negative activation to a decrease in mean and range of f_0.

Schröder (2004, Chapter 11) use the *Belfast Naturalistic Emotion Database* for his analysis. The database contains emotional speech material from 124 (English) speakers recorded from talk shows and "religious programs, as well as interviews recorded in a studio" (Schröder, Cowie, Douglas-Cowie, Westerdijk, & Gielen, 2001, p. 87). The recordings were split up into 5500 "inter-pause stretches" (Schröder et al., 2001, p. 88), which were placed on the same three-dimensional model as used by Pareira (2000) and mentioned by Cowie et al. (2001) by seven subjects. Schröder (2004, Chapter 11) used the ASSESS system to extract acoustic parameters and calculated the correlation and a linear regression for each dimension. He states, "that nearly all of the acoustic variables show substantial correlations with the emotion dimensions" (Schröder, 2004, p. 113). However, his correlation calculation returns significant results even for very small correlations due to the big database used. A *scatter plot* of the strongest correlation he found can be seen in Figure 3.

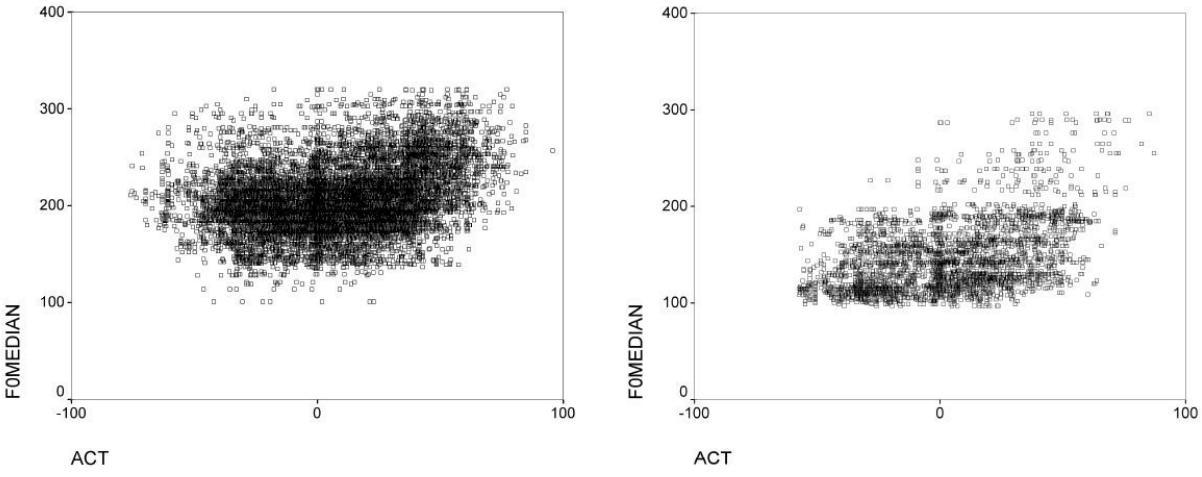


Figure 3: Fundamental frequency of utterances over arousal as presented by Schröder (2004, p. 108) for female (left) and male (right) speakers. Correlation coefficients were 0.313 for female speakers and 0.441 for male speakers (Schröder, 2004, pp. 114–115).

Schröder evaluated correlations with coefficients over 0.25 as very strong, between 0.14 and 0.25 as strong and correlations with coefficients between 0.08 and 0.14 as weak (regarding the absolute values). For the activation dimension, the parameters related to pitch induce the

strongest correlations (coefficients up to 0.44 for *mean* f_0 for male speakers). Regarding parameters that are not related to pitch, the strongest correlation is caused by the *spectral slope* with a coefficient value of 0.265 for female speakers. The strongest correlations within the evaluation dimension are those for *median duration* f_0 *falls* and the *Hammarberg index*[24] *'coarse'* (with coefficiants of 0.17 and -0.17). A correlation related to intensity could not be found by Schröder, which he assumes to be due to the amplitude manipulations that TV show audio tracks undergo (Schröder, 2004, p. 113). Schröder suggests not to use these correlations for models that don't include all three dimensions:

> For example, if the input to be used for calculating acoustic variables specified activation and evaluation, but not power, then it would not be optimal to use the coefficients calculated here. (Schröder, 2004, p. 111)

The mentioned studies seem to have certain drawbacks regarding a generalization or usefulness of the results for the purposes of the present thesis: Most of the studies either use only a handful of sentences to connect acoustic parameters to affective states (Goudbeek & Scherer, 2008, 2008; Kienast & Sendlmeier, 2000; Petrushin, 1999), or do not make use of natural sounding speech (Burkhardt & Sendlmeier, 2000; Oudeyer, 2003), use the same dataset for the training of an algorithm as for its testing (Oudeyer, 2003; Petrushin, 1999) or use a commercial synthesizer (Breazeal, 2002).

Yet, the examined studies generally agree on a positive correlation between pitch and arousal as well as between intensity and arousal. Correlations with the valence dimension are assumed to be parameters describing the voice quality[25], even though a general agreement as for the arousal dimension could not be found.

While an implementation of *Option A* as described in chapter 3.1 might be able to illustrate an affective state without evident acoustic correlations for the affective dimensions,[26] *Option B* depends on a generally valid correlation between at least one acoustic parameter and each affective dimension.

3.1.2 Phrase Analysis

The following chapters 3.1.2.1 and 3.1.2.2 describe two examinations of a correlation between several acoustic parameters and the affective dimensions of valence and arousal to get a brief idea of whether an implementation of *Option B* that can recreate affective states (especially along the valence dimension) based on acoustic features was feasible.

3.1.2.1 Movie and talk show utterance analysis

488 utterances with comparably very low background noise were extracted from movies and talk shows using the *Sonic Visualiser*[27]. They then were placed manually on the valence-arousal

[24] The Hammarberg indices are parameters created by Britta Hammarberg and her co-workers to describe voice quality features by the division of specific maxima of distinctive frequency bands (Schröder, 2004, p. 109).
[25] The *voice quality* as used here does not refer to a quality aspect of a speech recording, but to spectral characteristics of the voice itself.
[26] The essence of the affective state, even if it is not corresponding to acoustic parameters that have been examined, could be preserved in the speech material itself, then be recognized by the human listener and subconsciously translated to an *affective position* on the affective dimension
[27] The *Sonic Visualiser* is a freeware application for the "analysis, visualisation, and annotation of music audio files"(Cannam, Landone, & Sandler, 2010). It is available at http://www.sonicvisualiser.org/.

plane and examined regarding a correlation between their positions on the valence-arousal plane and acoustic features.

A MATLAB-GUI-application was developed to facilitate the manual placement of large numbers of audio files. A screenshot of the user interface is displayed in Figure C-1 in the appendix. Further explanations on the use of the application are given in the appendix on page 81f.

The utterances were chosen to cover the valence-arousal-plane in an approximately even manner. The positions can be seen in Figure 4. A linear correlation test on the two dimensions of the utterances showed no correlation with a Pearson correlation coefficient of −0.013 and a probability value of 0.77.

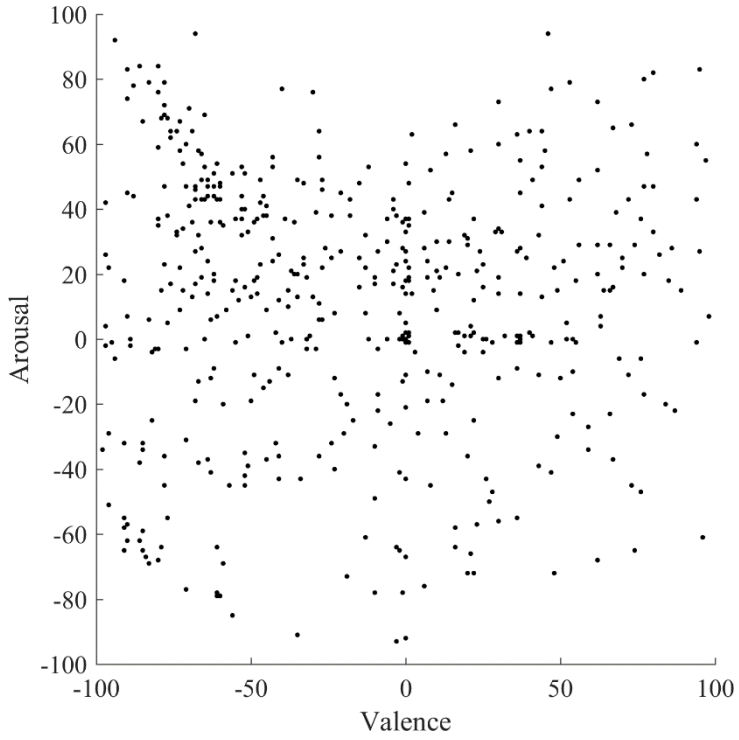


Figure 4: Positions of examined utterances from movies and talk shows across the valence-arousal-plane

46 acoustic parameters were extracted from each utterance covering pitch-related, intensity-related, spectral features as well as some other features. A complete list with explanations can be seen in Table 2.

Table 2: Acoustic Parameters extracted from the utterances and examined regarding a correlation with the affective dimensions

Pitch-related parameters	
▪ mean of f_0 ▪ minimum of f_0 ▪ maximum of f_0 ▪ *spread* of f_0 ▪ standard deviation of f_0	The fundamental frequency was calculated via an autocorrelation applied on the zero-frequency-filtered signal as presented by Murty & Yegnanarayana (2008, p. 1608).[28] The *spread* of f_0 was calculated as the difference of the 90th percentile and the 10th percentile of f_0.

[28] This process is explained in further detail in section III.3.2.2.4

▪ mean of relative pitch changes ▪ minimal relative pitch change ▪ maximal relative pitch change ▪ relative pitch change variance ▪ standard deviation of relative pitch changes	These parameters examine the *relative pitch changes* between successive voiced parts and were included to examine the *inter-syllable* pitch relations within an utterance. A relative pitch change is defined as the absolute difference of the ratio of pitches of successive voiced parts to 1. The voiced/unvoiced discrimination was estimated frame-wise with the number of zero crossings of the signal divided by the signal energy according to Bachu, Kopparthi, Adapa, & Barkana (2010).
▪ Number of pitch rises and falls ▪ Duration of pitch rises and falls ▪ Culminated amount of pitch rises and falls	To calculate the pitch rises and falls, the pitch returned by an autocorrelation function was smoothed by a moving-average filter. Signal parts in between turning points and end points of the resulting pitch curve were counted as pitch falls or rises. An example can be seen in Figure 5.
▪ Maximal *syllable prominence*	The maximal peak prominence of a pitch contour formed by the mean pitch of each syllable.
▪ mean of pitch steps	Absolute differences of the mean pitches of successive syllables.
Intensity related parameters	
▪ mean of RMS of utterance ▪ maximum of RMS of utterance ▪ variance of RMS of utterance	The RMS was calculated over windows of 1024 samples length with an overlap of 512 samples.
▪ RMS of voiced parts	The voiced/unvoiced decision was implemented as described by Bachu et al. (2010): If the quotient of zero crossing rate and energy exceeds a threshold, the examined window is regarded as *unvoiced*. Unvoiced parts were removed from the signal and the RMS was calculated with the remaining signal. The impact of each part on the result depends therefore on the length of the part.
▪ mean of relative RMS changes ▪ minimal relative RMS change ▪ maximal relative RMS change ▪ variance of relative RMS changes ▪ standard deviation of relative RMS changes	Describe the ratio of RMS values between successive voiced parts. As with the pitch, these parameters were included to examine inter-syllable RMS progressions. Differences between successive voiced parts were divided through the first of two syllables.
▪ mean of RMS peaks ▪ mean of distance in between RMS peaks	The peaks were also calculated over frames of 1024 samples length with an overlap of 512 samples. The distance between peaks could be a way to estimate the speed of speech.
Spectrum-related parameters	
▪ Spectral Slope of voiced parts ▪ Spectral Flatness of voiced parts ▪ Spectral Rolloff of voiced parts	Apart from the Spectral Slope, these parameter calculations were done with code provided by Lerch (n.d.) according to formulas descibed in

▪ Spectral Centroid of voiced parts ▪ Spectral Spread of voiced parts ▪ Tonal Power Ratio of voiced parts	Lerch (2012, Chapter 3).[29] The Spectral Slope was retrieved as the linear approximation of the FFT-spectrum from 70 to 8000 Hz.
▪ Hammarberg Vocal Effort ▪ Hammarberg Breathy Voice ▪ Hammarberg Head ▪ Hammarberg Coarse ▪ Hammarberg Unstable	These Hammarberg Indices were implemented as described by Schröder (2004, p. 109) and were only calculated over the voiced parts.
▪ ratio of spectral energy above 400 Hz to under 400 Hz ▪ ratio of spectral energy above 3000 Hz to under 3000 Hz	The cumulated sums of the corresponding frequency bins of FFTs of the utterances as a whole.
Other parameters	
▪ zero crossings per second ▪ mean of aperiodicity (from Yin-algorithm[30]) ▪ proportion of voiced parts ▪ speech rate (as the reciprocal of the length of voiced parts)	

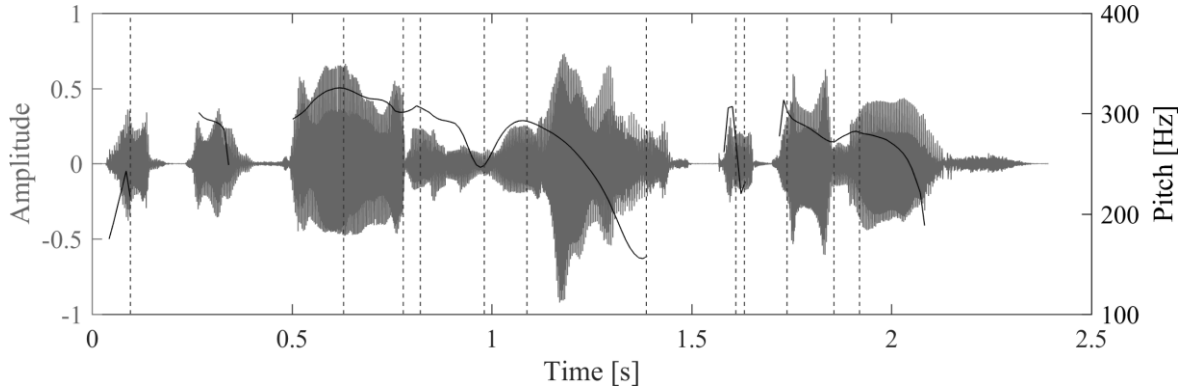


Figure 5: Pitch contour of the shouted phrase "Lass mein Fahrrad in Ruhe, das geht doch nicht!" (roughly translated “Let go off my bike, you can’t do that!”). Turning points are marked as horizontal, dashed lines.

A *Jarque-Bera test* revealed, that the distributions for most variables do not correspond to a normal distribution. As a linear correlation after Pearson requires data that is distributed normally, a rank correlation test after Spearman was computed for all extracted audio features. Results can be seen in Table 3. (A linear correlation test was computed, too, since some variables were normally distributed, and it turned out to show very similar results which can be seen in Table C-1 in Appendix C.) To take in account a possible quadratic relation between acoustic parameters and affective dimensions a coefficient of determination[31] for the curve fitting of a quadratic curve was calculated for both dimensions. Results for the curve fitting can be seen in Table 4.

[29] The code is available (as of October ‘17) on the following website: https://www.audiocontentanalysis.org/code/
[30] The code for the Yin-pitch-extraction was taken from Llimona (2014/2015)
[31] The R^2-value as a returned value of the Matlab function *fit* was used as determination coefficiant

Table 3: Correlation and probability value of 46 acoustic parameters extracted from 453 utterances from movies and talk shows along the valence dimension (left) and the arousal dimension (right).

Valence			Arousal		
Parameter	**Corr.**	**p**	**Parameter**	**Corr.**	**p**
HammarbergVocalEffort	-0.1530	< 0.001	HammarbergUnstable	0.5184	< 0.001
HammarbergUnstable	-0.1455	0.0013	PeakIntensity	0.5115	< 0.001
HammarbergHead	-0.1428	0.0016	MeanRMS	0.4984	< 0.001
SpectralRatio400	0.1379	0.0023	HammarbergVocalEffort	0.4962	< 0.001
HammarbergCoarse	-0.1319	0.0035	MaxRMS	0.4503	< 0.001
MaxRelPitchDifferences	-0.1315	0.0045	PitchStandard	0.4393	< 0.001
PitchFallsAmount	-0.1300	0.0058	PitchVariance	0.4356	< 0.001
ZeroCrossingsRate	0.1291	0.0043	MaxPitch	0.4185	< 0.001
MeanRelPitchDifferences	-0.1274	0.0059	RMSVariance	-0.4092	< 0.001
PitchStandard	-0.1222	0.0069	SpectralRatio400	-0.3817	< 0.001
PitchVariance	-0.1136	0.0120	MeanAperiodicity	-0.3428	< 0.001
PitchRisesAmount	-0.1134	0.0176	MeanRelPitchDiff.	0.3367	< 0.001
VarRelPitchDifferences	-0.1122	0.0154	PeakDistance	0.3364	< 0.001
StdRelPitchDifferences	-0.1122	0.0154	PitchFallsAmount	0.3314	< 0.001
HammarbergBreathyV.	-0.1109	0.0142	MeanPitchStep	0.3247	< 0.001
MaxPitch	-0.1105	0.0146	MeanPitch	0.3128	< 0.001
MeanPitchStep	-0.1066	0.0190	PitchRisesAmount	0.2915	< 0.001
SpectralSpread	0.0869	0.0554	VarRelPitchDifferences	0.2806	< 0.001
SpectralRolloff	0.0865	0.0563	StdRelPitchDifferences	0.2806	< 0.001
RMSVariance	0.0853	0.0598	MaxRelPitchDifferences	0.2783	< 0.001
MeanPitch	-0.0850	0.0606	MaxProminence	0.2719	< 0.001
MeanRMS	-0.0774	0.0876	VoicedFraction	0.2619	< 0.001
TonalPowerRatio	-0.0738	0.1038	HammarbergHead	0.2463	< 0.001
SpeechRate	-0.0737	0.1045	SpectralRatio3000	0.2419	< 0.001
MaxProminence	-0.0678	0.1362	SpectralCentroid	-0.2318	< 0.001
MaxRMS	-0.0608	0.1800	SpeechRate	0.2268	< 0.001
SpectralFlatness	0.0594	0.1910	PitchRisesDuration	-0.1994	< 0.001
MinPitch	0.0591	0.1926	HammarbergCoarse	0.1903	< 0.001
VoicedFraction	-0.0581	0.1999	SpectralFlatness	-0.1799	< 0.001
PeakIntensity	-0.0568	0.2105	MeanRelRMSDiff.	-0.1430	0.0020
PitchRisesDuration	0.0514	0.2832	SpectralSpread	-0.1424	0.0016
SpectralSlope	-0.0449	0.3224	PitchFallsDuration	-0.1354	0.0040
MinRelPitchDifferences	-0.0416	0.3705	SpectralRolloff	-0.1267	0.0051
SpectralCentroid	0.0414	0.3620	HammarbergBreathyV.	0.1232	0.0064
MaxRelRMSDifferences	0.0321	0.4888	PitchRisesFrequency	0.1226	0.0067
MeanAperiodicity	0.0313	0.4899	MinRelRMSDifferences	-0.1040	0.0248
MinRelRMSDifferences	-0.0272	0.5587	MinPitch	-0.1016	0.0248
PitchFallsDuration	0.0205	0.6645	ZeroCrossingsRate	-0.0957	0.0347
VarRelRMSDifferences	0.0181	0.6961	VarRelRMSDifferences	-0.0902	0.0516
StdRelRMSDifferences	0.0181	0.6961	StdRelRMSDifferences	-0.0902	0.0516
SpectralRatio3000	-0.0167	0.7121	SpectralSlope	0.0821	0.0701
PitchFallsFrequency	0.0127	0.7801	PitchFallsFrequency	0.0660	0.1456
PeakDistance	0.0094	0.8364	TonalPowerRatio	-0.0626	0.1675
PitchRisesFrequency	-0.0071	0.8749	MaxRelRMSDifferences	-0.0593	0.2010
MeanRelRMSDifferences	0.0067	0.8861	MinRelPitchDifferences	0.0444	0.3386
RMSVoiced	0.0050	0.9121	RMSVoiced	0.0419	0.3556

Table 4: R^2-Coefficiant of determination of quadratic curve fitting for 46 acoustic parameters extracted from 453 utterances taken from movies and talk shows along the valence dimension (left) and the arousal dimension (right)

Valence		Arousal	
Parameter	**R^2**	**Parameter**	**R^2**
PitchStandard	0.1364	HammarbergUnstable	0.1690
PitchVariance	0.1249	HammarbergVocalEffort	0.1377
PitchFallsAmount	0.1138	PeakIntensity	0.1321
MaxPitch	0.1069	PitchVariance	0.1196
PitchRisesAmount	0.0939	PitchStandard	0.1120
HammarbergUnstable	0.0617	MaxPitch	0.1059
HammarbergVocalEffort	0.0611	MeanAperiodicity	0.1051
MeanPitch	0.0556	MeanRMS	0.1038
SpectralRatio400	0.0540	RMSVariance	0.0867
StdRelPitchDifferences	0.0390	SpectralRatio400	0.0781
MeanRelPitchDifferences	0.0385	MaxRMS	0.0771
VarRelPitchDifferences	0.0378	PitchFallsAmount	0.0673
MaxRelPitchDifferences	0.0372	MeanPitchStep	0.0587
PitchRisesDuration	0.0356	MeanRelPitchDifferences	0.0532
HammarbergHead	0.0356	MeanPitch	0.0498
MeanPitchStep	0.0324	MaxProminence	0.0440
HammarbergCoarse	0.0287	PitchRisesAmount	0.0408
PeakIntensity	0.0284	StdRelPitchDifferences	0.0364
MinPitch	0.0276	MaxRelPitchDifferences	0.0360
PitchFallsFrequency	0.0250	VarRelPitchDifferences	0.0314
HammarbergBreathyVoice	0.0220	PeakDistance	0.0269
PitchFallsDuration	0.0204	VoicedFraction	0.0269
MaxProminence	0.0193	SpectralRatio3000	0.0264
RMSVoiced	0.0164	HammarbergHead	0.0254
SpectralSpread	0.0120	RMSVoiced	0.0239
ZeroCrossingsRate	0.0115	PitchRisesDuration	0.0237
PeakDistance	0.0107	SpectralCentroid	0.0232
TonalPowerRatio	0.0094	SpectralSlope	0.0194
PitchRisesFrequency	0.0091	PitchRisesFrequency	0.0166
RMSVariance	0.0090	SpeechRate	0.0131
SpectralRatio3000	0.0080	HammarbergCoarse	0.0122
StdRelRMSDifferences	0.0079	SpectralSpread	0.0102
MaxRelRMSDifferences	0.0077	SpectralRolloff	0.0095
MeanRelRMSDifferences	0.0071	SpectralFlatness	0.0056
SpectralRolloff	0.0068	PitchFallsFrequency	0.0037
VoicedFraction	0.0065	PitchFallsDuration	0.1690
SpectralSlope	0.0058	MeanRelRMSDifferences	0.1377
SpectralCentroid	0.0051	MinPitch	0.1321
VarRelRMSDifferences	0.0039	MaxRelRMSDifferences	0.1196
MaxRMS	0.0039	StdRelRMSDifferences	0.1120
SpectralFlatness	0.0038	HammarbergBreathyVoice	0.1059
MinRelPitchDifferences	0.0037	TonalPowerRatio	0.1051
SpeechRate	0.0031	VarRelRMSDifferences	0.1038
MeanRMS	0.0030	ZeroCrossingsRate	0.0867
MinRelRMSDifferences	0.0022	MinRelPitchDifferences	0.0781
MeanAperiodicity	0.0006	MinRelRMSDifferences	0.0771

Despite the fact, that numerous correlations are classified as *significant* over both dimensions (no matter which *level of significance* is chosen), the measured correlations are quite low. The

low probability values are seemingly merely a consequence of the large sample size.[32] This becomes evident when taking a look at the scatter plots. Figure 6 shows the parameter with one of the strongest correlations over both dimensions: The Hammarberg Index for *Vocal Effort*.

The increasing distribution of values for higher arousal is well-distinguishable, hence the increasing mean and the positive correlation. However, taken into account that not all the values themselves seem to increase but only the spread of the values, a value distribution as seen for the arousal dimension allows no conclusions about the position of an utterance within the arousal dimension for small values. Only the range of possibilities towards higher values is reduced.

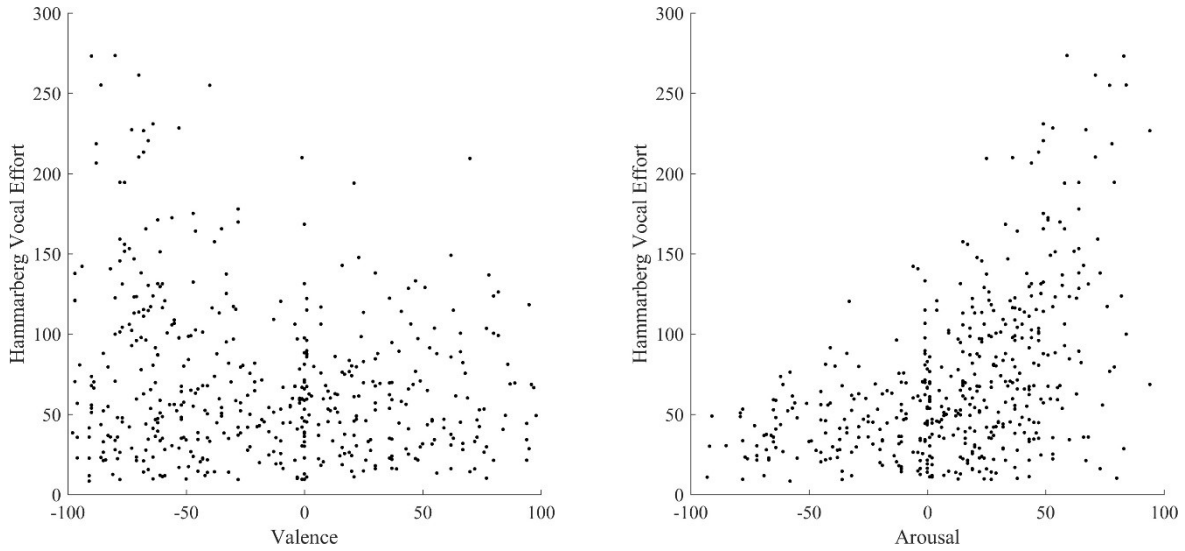


Figure 6: Acoustic parameter *Hammarberg Vocal Effort* of utterances over valence dimension (left) and arousal dimension (right).

The distribution of *Vocal Effort* values seems to decrease with an increase of the valence value, yet, the measured correlation of the values themselves is with a coefficient of −0.15 very low.

The low correlation values for pitch and intensity, which are usually the parameters with the highest and most stable correlations across studies, can be explained by two reasons:

1. Parameters were compared without any normalization. In this context, a normalization would mean, that resulting parameter values are first set in relation to the parameter values of neutral utterances from the same speaker before comparing parameter values of different speakers. The parameter extraction does therefore not take into account the personal characteristics of the speakers, e.g. the generally higher voice of a child.
   A normalization could not be provided for this examination, as it was not possible to find neutral statements for every speaker with a sufficient signal-to-noise ratio.
2. As the utterances were taken from soundtracks of movies and talk shows, the audio has been dynamically processed, which is why effects of the affective state on the intensity-related parameters are diminished.

3.1.2.2 Utterance analysis of Grimaldi's corpus

To enable a normalization and examine dynamically untreated speech signals, the same feature extraction was computed with the speech recordings made by Grimaldi et al. (2017). More

[32] Compare e.g. with the presented study by Schröder (2004, Chapter 11) in section III.3.1.1

information on these recordings is given in section 2.3.4.
766 utterances were manually extracted from the recordings using the *Sonic Visualiser* and sorted by speaker. The utterances were also placed manually on the valence-arousal-plane with the MATLAB-GUI-application. The extracted features were normalized by subtracting the parameter values of a pre-defined *neutral* utterance (closest to the center of the valence-arousal plane) for every speaker. The normalization does not only level specific characteristics of the speakers' voices, but also diminishes effects of different distances to the microphones due to different sitting postures during the recordings. The parameters were examined regarding a correlation with the two affective dimensions in the same way as for the examination described in the previous chapter. The correlation coefficients of a rank correlation after *Spearman* and respective probability values can be seen in Table 5. To examine non-linear relations, a quadratic curve fitting was computed on the features. The resulting coefficients of determination can be seen in Table 6. The very similar results of a linear correlation can be seen in Table C-2 in Appendix C.

Table 5: Correlation coefficients and p-Values of acoustic features of 766 utterances from the recordings made by Grimaldi et al. (2017) for the valence dimension (left) and the arousal dimension (right).

Valence			**Arousal**		
Parameter	**Corr.**	**p-val.**	**Paramter**	**Corr.**	**p-val.**
HammarbergCoarse	-0.3513	<0.0001	MaxRMS	0.7862	<0.0001
HammarbergBreathyV.	-0.3481	<0.0001	MeanRMS	0.7678	<0.0001
HammarbergHead	-0.3284	<0.0001	PeakIntensity	0.7621	<0.0001
MinPitch	-0.2585	<0.0001	RMSVariance	-0.6984	<0.0001
MeanPitch	-0.2219	<0.0001	MeanPitch	0.6937	<0.0001
RMSVariance	0.2216	<0.0001	MaxPitch	0.6690	<0.0001
MeanAperiodicity	0.2173	<0.0001	HammarbergVocalEffort	0.5966	<0.0001
MeanPitchStep	-0.2152	<0.0001	HammarbergHead	0.5454	<0.0001
MeanRelPitchDiff.	-0.2033	<0.0001	PitchStandard	0.5452	<0.0001
PitchRisesAmount	-0.2024	<0.0001	HammarbergCoarse	0.5288	<0.0001
PitchFallsAmount	-0.1969	<0.0001	PitchFallsAmount	0.5084	<0.0001
MeanRMS	-0.1963	<0.0001	PitchVariance	0.5025	<0.0001
ZeroCrossingsRate	-0.1623	<0.0001	PitchRisesAmount	0.4873	<0.0001
MaxPitch	-0.1487	<0.0001	MeanPitchStep	0.4838	<0.0001
MaxRMS	-0.1455	0.0001	HammarbergUnstable	0.4795	<0.0001
PeakIntensity	-0.1400	0.0001	HammarbergBreathyV.	0.4752	<0.0001
PeakDistance	-0.1400	0.0001	SpectralRatio400	-0.4185	<0.0001
SpectralCentroid	-0.1342	0.0002	MeanRelPitchDiff.	0.3965	<0.0001
PitchRisesDuration	-0.1334	0.0004	PeakDistance	0.3526	<0.0001
MaxProminence	-0.1317	0.0003	MaxProminence	0.3522	<0.0001
MaxRelPitchDifferences	-0.1280	0.0007	StdRelPitchDifferences	0.3314	<0.0001
SpectralSpread	-0.1263	0.0005	VarRelPitchDifferences	0.3213	<0.0001
RMSVoiced	-0.1161	0.0013	MaxRelPitchDifferences	0.3083	<0.0001
PitchFallsDuration	-0.1075	0.0041	TonalPowerRatio	-0.2801	<0.0001
HammarbergVocalEff.	-0.1034	0.0042	MinPitch	0.2171	<0.0001
StdRelPitchDifferences	-0.1015	0.0072	MinRelPitchDifferences	0.1822	<0.0001
SpectralRolloff	-0.1004	0.0058	SpectralFlatness	-0.1702	<0.0001
SpectralSlope	-0.0980	0.0071	MeanAperiodicity	-0.1369	0.0001
PitchStandard	-0.0980	0.0070	PitchFallsFrequency	-0.1300	0.0003
TonalPowerRatio	0.0962	0.0083	SpectralCentroid	-0.1288	0.0004
VoicedFraction	-0.0909	0.0119	ZeroCrossingsRate	0.1282	0.0004
SpectralRatio3000	-0.0895	0.0132	SpectralRatio3000	-0.0968	0.0074

VarRelPitchDifferences	-0.0862	0.0225	PitchRisesFrequency	-0.0947	0.0087
MinRelRMSDifferences	0.0794	0.0355	PitchFallsDuration	-0.0879	0.0191
PitchVariance	-0.0610	0.0934	SpectralSpread	-0.0824	0.0237
MinRelPitchDifferences	-0.0597	0.1145	VoicedFraction	-0.0808	0.0254
PitchRisesFrequency	-0.0567	0.1169	SpectralRolloff	-0.0624	0.0870
VarRelRMSDifferences	0.0532	0.1591	MinRelRMSDifferences	-0.0624	0.0989
MaxRelRMSDiff.	0.0530	0.1607	SpectralSlope	0.0431	0.2372
StdRelRMSDifferences	0.0494	0.1910	SpeechRate	0.0405	0.2673
SpectralFlatness	-0.0452	0.2157	VarRelRMSDifferences	-0.0364	0.3360
MeanRelRMSDiff.	0.0399	0.2917	StdRelRMSDifferences	-0.0316	0.4041
HammarbergUnstable	-0.0370	0.3059	MeanRelRMSDiff.	-0.0285	0.4507
SpeechRate	0.0279	0.4440	RMSVoiced	0.0194	0.5925
SpectralRatio400	0.0210	0.5612	MaxRelRMSDifferences	-0.0192	0.6115
PitchFallsFrequency	-0.0170	0.6387	PitchRisesDuration	0.0094	0.8040

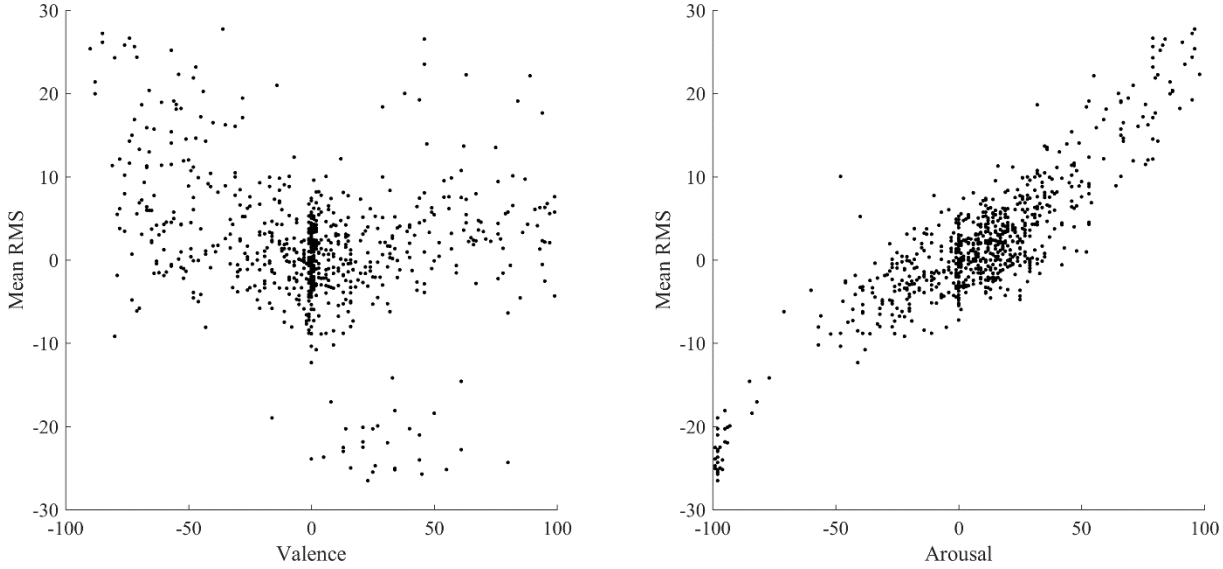
The analysis results show a high correlation for RMS- and pitch-related features with the arousal dimension, which is also clearly visible in the corresponding scatter plots (see Figure 7 and Figure 8). This confirms results of previous studies mentioned in section 3.1.1.

Table 6: Coefficients of determination of a quadratic curve fitting of acoustic features of 766 utterances from recordings made by Grimaldi et al. (2017) for the valence dimension (left) and the arousal dimension (right).

Valence		**Arousal**	
Parameter	**R^2**	**Parameter**	**R^2**
MaxPitch	0.3168	MeanRMS	0.8093
MeanPitch	0.3140	MaxRMS	0.8006
RMSVariance	0.2904	MeanPitch	0.7058
MaxRMS	0.2853	RMSVariance	0.6934
PitchStandard	0.2776	MaxPitch	0.6198
PitchFallsAmount	0.2671	SpectralSlope	0.4964
PitchVariance	0.2376	HammarbergVocalEffort	0.4856
MeanRMS	0.2248	PitchFallsAmount	0.4722
PitchRisesAmount	0.1959	HammarbergUnstable	0.4514
HammarbergVocalEffort	0.1730	PitchStandard	0.4362
HammarbergUnstable	0.1609	HammarbergHead	0.4014
MeanPitchStep	0.1301	MeanPitchStep	0.3824
PeakIntensity	0.1285	PitchVariance	0.3654
SpectralRatio3000	0.1087	HammarbergCoarse	0.3489
MeanAperiodicity	0.1073	PitchRisesAmount	0.3470
PeakDistance	0.1066	PeakIntensity	0.3469
VoicedFraction	0.1019	SpectralRatio400	0.3088
HammarbergHead	0.0979	VoicedFraction	0.2977
MaxProminence	0.0971	HammarbergBreathyVoice	0.2758
PitchRisesDuration	0.0895	PitchFallsFrequency	0.2616
PitchRisesFrequency	0.0852	SpectralRatio3000	0.2248
HammarbergCoarse	0.0780	MaxProminence	0.2242
SpectralSpread	0.0778	PitchRisesFrequency	0.2189
MinPitch	0.0738	PeakDistance	0.1966
PitchFallsFrequency	0.0672	MeanRelPitchDifferences	0.1886
SpectralRatio400	0.0663	MeanAperiodicity	0.1842
SpectralRolloff	0.0538	SpeechRate	0.1742
HammarbergBreathyVoice	0.0536	MaxRelPitchDifferences	0.1424
TonalPowerRatio	0.0520	StdRelPitchDifferences	0.1333
MeanRelPitchDifferences	0.0488	TonalPowerRatio	0.1221

SpectralCentroid	0.0354	MinPitch	0.0755
MaxRelPitchDifferences	0.0350	ZeroCrossingsRate	0.0654
PitchFallsDuration	0.0340	VarRelPitchDifferences	0.0601
VarRelPitchDifferences	0.0269	MinRelPitchDifferences	0.0470
StdRelPitchDifferences	0.0264	SpectralFlatness	0.0420
SpectralSlope	0.0226	RMSVoiced	0.0342
ZeroCrossingsRate	0.0219	SpectralSpread	0.0306
SpectralFlatness	0.0177	SpectralRolloff	0.0216
SpeechRate	0.0167	SpectralCentroid	0.0206
RMSVoiced	0.0099	PitchFallsDuration	0.0115
MinRelPitchDifferences	0.0088	PitchRisesDuration	0.0049
MinRelRMSDifferences	0.0067	MinRelRMSDifferences	0.0031
MeanRelRMSDifferences	0.0024	VarRelRMSDifferences	0.0015
MaxRelRMSDifferences	0.0011	StdRelRMSDifferences	0.0011
StdRelRMSDifferences	0.0010	MeanRelRMSDifferences	0.0009

According to the correlation coefficient, the values for the mean of the pitch and the RMS correlate slightly with a decreasing valence. A look at the scatter plots, however, reveals, that this might merely be a consequence of an uneven distribution of the utterances. As Figure 9 shows, some areas of the valence-arousal plane are not covered by utterances provided by Grimaldi et al.'s (2017) recordings and some areas contain more utterances than others: Recordings of whispered speech in a bad mood, e.g. had not been made, only few utterances with positive valence and high arousal are given. *Cheering* and *joyful shouting*, e.g. should have comparable RMS values as angry shouting, but utterances with these affective states were not available within the recordings. This distribution causes a light correlation of the two dimensions with a coefficient of −0.29, which is approximately the same level of correlation as the highest correlations found within the valence dimension. The results for this dimension are therefore discarded.[33]



[33] The correlation between the two dimensions obviously weakens results in both dimensions. Since most correlations found for the arousal dimension are much stronger than for the valence dimension, it can be assumed, that the arousal dimension results influenced the results for the valence dimension for these parameters rather than the other way around.

Figure 7: Values of utterances for parameter *Mean of RMS* relative to values of neutral utterance over valence dimension (left) and arousal dimension (right).

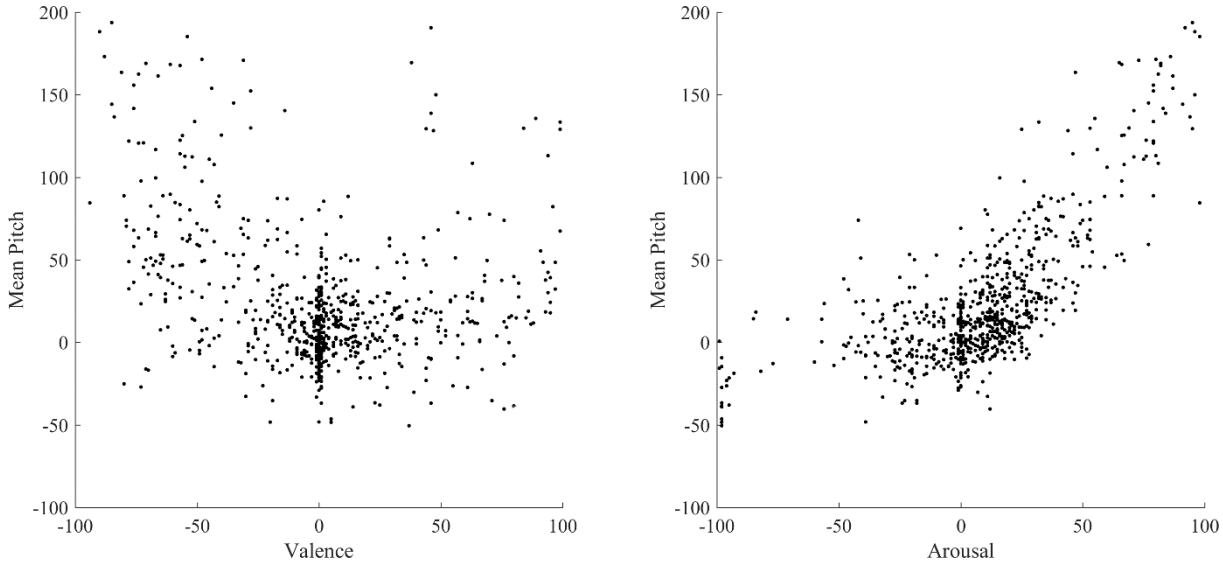


Figure 8: Values of utterances for parameter *Mean of Pitch* relative to values of neutral utterance over valence dimension (left) and arousal dimension (right)

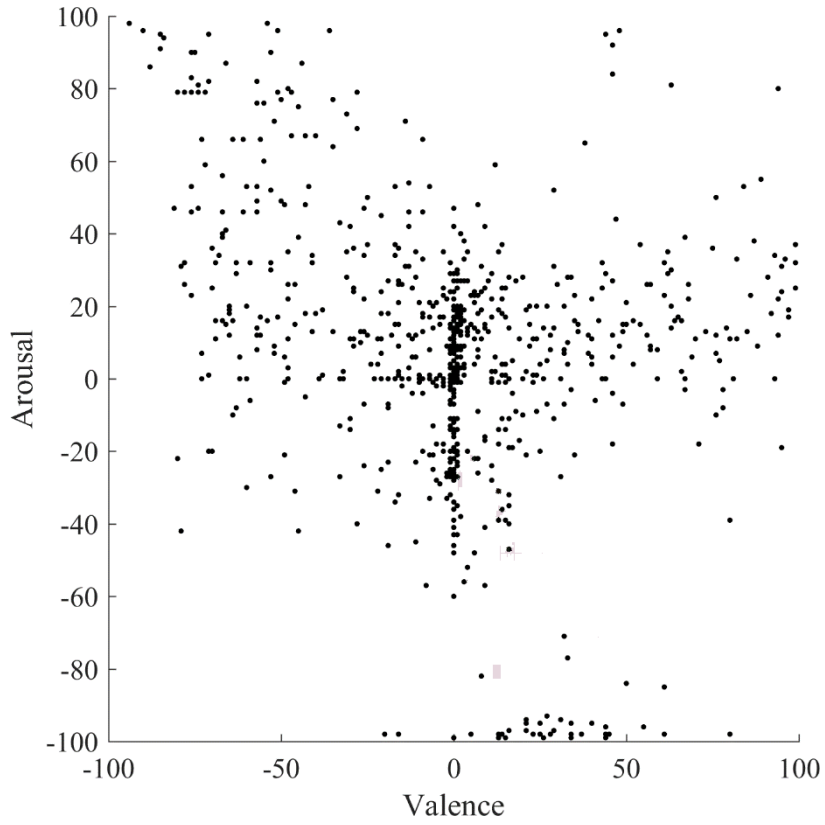


Figure 9: Positions of utterances from recordings made by Grimaldi et al.(2017)

Results of the feature extractions of movie and talk show utterances and of the recordings by Grimaldi et al. (2017) as well as the review on previous studies reveal, that an adjustable generation of affective states along the arousal dimension should be possible using intensity and/or pitch-related acoustic parameters.
Regarding the valence dimension, the examination of movie and talk show utterances could not reveal strong or moderate correlations of examined acoustic parameters and the affective dimension. The strongest correlations found are weak negative correlations for the Hammarberg Indices *Vocal Effort* and *Unstable*. These findings agree with the assumption stated in previous studies, that correlations with the valence dimension could be found for parameters describing some aspect of *voice quality*, are, however, presumably not enough to deduce rules for a generation of affective states along the affective dimension.

3.1.3 Implemented models to generate affective states

Although no evident rules for the generation of different valence *positions* could be derived from the acoustic features of the utterances, the decision was made to test three different implementations of the valence-arousal model in the human crowd noise generation. The models are explained in the following sections. Figure 10 on page 33 provides an overview on the work flow of the three models.

3.1.3.1 Model A1

As described as *Option A* in section 3.1, the grains for the synthesis are mapped to a specific position on the valence-arousal-plane. The positions are taken from the manual positioning of utterances computed for the phrase analysis described in section 3.1.2.2 and are stored together with the audio data of each grain and a respective pitch value to be read by the synthesis algorithm.

3.1.3.2 Model A2

Model A2 is also an implementation of *Option A*, but with an automated mapping of the utterances to a position on the valence-arousal-plane through machine learning algorithms. A combined regression- and classification-model was created for the automatic positioning of the utterances:

The manually placed utterances from Grimaldi et al's (2017) recordings were split into a training set and a test set. From the list of acoustic parameters described in section 3.1.2.1, the ones with the strongest correlation coefficients in the respective dimension were used as training data for 24 different *regression learner models* provided by the MATLAB Statistics And Machine Learning Toolbox. A cross-validation with ten folds was computed. The regression model with the lowest expected error rate was chosen for the respective dimension.

Parameters used for the valence regression model: Mean of pitch, minimum of pitch, maximum of pitch, RMS variance, RMS of voiced parts, peak intensity, peak distance, mean of aperiodicity, spectral slope, spectral flatness, spectral rolloff, spectral centroid, spectral spread, tonal power ratio, zero crossings rate, Hammarberg Vocal Effort, Hammarberg Breathy Voice, Hammarberg Head, Hammarberg Coarse, Hammarberg Unstable, modified Hammarberg Index[34], pivot frequency of modified Hammarberg Index, spectral ratios[35] (with border

[34] A *Hammarberg Index* with flexible pivot frequency as explained by Tamarit, Goudbeek & Scherer (2008)
[35] Spectral energy under the border frequency (which is the number in the name) divided through the spectral energy over the border frequency

frequencies 300, 500, 1000 and 3000), fraction of voiced parts, VUV ratio[36], speech rate[37], maximum prominence[38] and mean of pitch steps.

Parameters used for the arousal regression model: Mean of RMS, maximal RMS, RMS spread[39], mean of RMS of voiced parts, mean of aperiodicity[40], Hammarberg Vocal Effort, Hammarberg Breathy Voice, Hammarberg Head, Hammarberg Coarse, Hammarberg Unstable, Modified Hammarberg Index, fraction of voiced parts and speech rate.
Although the pitch-related parameters had a high correlation with the arousal dimension, they were not included into the arousal regression model, as the utterances containing whispered speech sometimes resulted in high means of pitch, which reduced the accuracy of the regression model.

For the valence dimension the regression model *bootstrap-aggregated ensemble of bagged trees* was chosen with an RMSE (root-mean-square error) of 30.38 and R^2 (as coefficient of determination) of 0.36.
An *ensemble of regression trees using the LSBoost algorithm* was chosen as regression model for the arousal dimension with an RMSE of 14.00 and R^2 of 0.85.

Since the performance of the valence regression model is quite low (as expected, based on the weak correlations described in section 3.1.2), another approach for an automated position mapping was implemented:
The valence-arousal-plane was divided up into three parts in each dimension, resulting in nine equally spaced squares. The squares were given a number and the utterances within one square were attributed its number representing an *affective class* (see Table 7 for numbering).

Table 7: Numbering of affective classes gained by splitting up the valence-arousal dimension into nine fields.

7	8	9
4	5	6
1	2	3

Using the *MATLAB Classification Learner*[41], 27 different classification models were trained with the same utterances and acoustic features as used for the training of the two regression models. With an estimated accuracy of 70.6%, the *Ensemble of Bagged Trees* - a "bootstrap-aggregated ensemble of complex decision trees" (The MathWorks Inc., 2017) – showed the best result and was chosen as classification model. A confusion matrix of the classification model is illustrated in Table 8. As the utterances are not evenly distributed across the classes, it is not surprising, that classes with only few utterances have a lower recognition rate and that

[36] The result of dividing the zero crossings per window through the energy of the waveform, which can be used to distinguish a degree of *unvoicedness* in speech (Bachu, Kopparthi, Adapa, & Barkana, 2010)
[37] As the reciprocal of the length of voiced parts
[38] Regarding the progression of the mean of the pitch values for every syllable, the *maximal prominence* is the maximal distance of a pitch peak to the highest neighbouring pitch value
[39] Maximal RMS value minus minimal RMS value
[40] Gained from the YIN pitch estimator by Llimona (2014/2015)
[41] The *MATLAB Classification Learner App* is available as part of the *Statistics and Machine Learning Toolbox* by The MathWorks Inc: https://de.mathworks.com/products/statistics html

the model tends to misplace utterances into the *neutral* class number 5 around the center, which contains by far the most utterances.

Table 8: Confusion matrix of classification model of implementation *Model A2*.

		Predicted class								
		1	2	3	4	5	6	7	8	9
True class	1					2				
	2		20	1	1	32				
	3		3	2		9	2			
	4				17	40	3	4		
	5		4		5	400	10	1	2	
	6				2	43	36	2	2	1
	7				4	5	1	52	4	2
	8				1	11	4	7	13	
	9					1	4	12		1

The classification model is thus used as a *fallback* for cases, in which the valence regression model returns an invalid value (e.g. a *NaN*[42] or a value not within the expected range). In these cases, the valence value is set to the valence value of the center position of the square corresponding to the class number returned by the classifier.
The arousal value for every utterance is directly taken from the regression model.

Since the regression models and the classification model expect normalized values, *Model A2* requires at least one neutral utterance per speaker that allows a normalization of acoustic parameters.

As with *Model A1*, position values as well as a pitch value are stored together with the audio data in order to be read afterwards by the synthesis algorithm.

3.1.3.3 Model B

Model B is an implementation of *Option B* described in section 3.1, which was mostly created to test a performance along the affective dimension of arousal. The syllables are not directly mapped on the valence-arousal-plane, but are added to the selection of syllables to be used in runtime, if their acoustic features fit the parameter profile of the currently selected *affective position*.

The extraction of parameter values is computed as a part of the corpus creation. The parameter values are normalized in a way to cover a range from 0 to 1, where 0.5 corresponds to the 50th percentile of all values. To diminish the influence of outliers and close gaps between outliers and other values, values below the 1st percentile are set to 0 and values over the 99th percentile are set to 1.

[42] Short for *not a number*, usually returned when a result is invalid (e.g. after division through zero)

The values are stored in an XML-file, which can then be read by the synthesis algorithm.

Apart from sorting the speech material by speaker, the creation of the corpus for this model is fully automated. Figure 10 illustrates the work flow for the three different, implemented models.

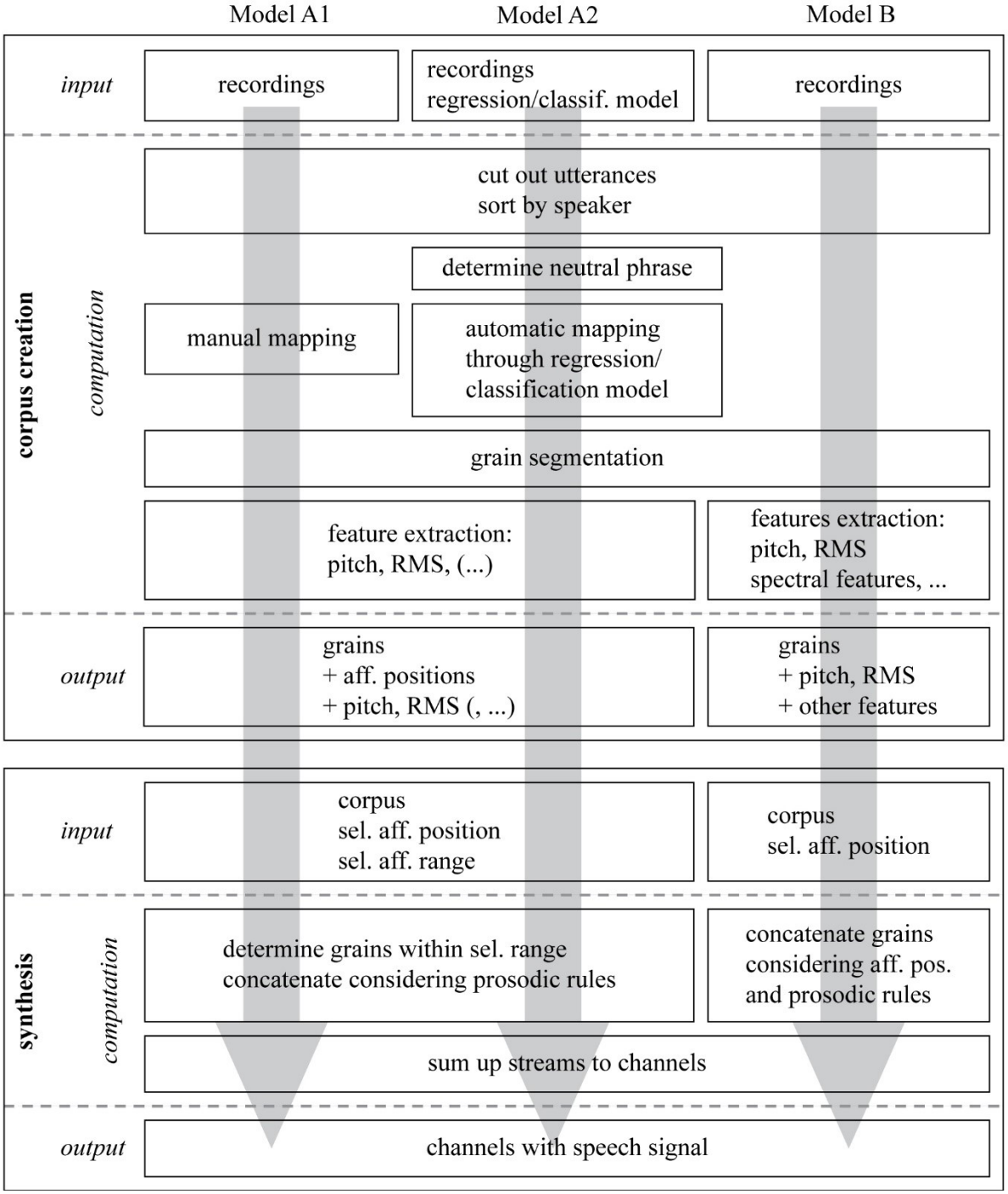


Figure 10: Workflow chart of three implemented ways to apply the valence-arousal model to the crowd noise synthesis ordered by degree of automation from left to right

3.2 Syllable segmentation

It was decided to use syllables or syllable-like grains as speech units as a compromise between naturalness and corpus size. The recordings by Grimaldi (2017), which are used for the synthesis presented here, demand for a segmentation of the speech material into these syllables or *grains*, in order to allow a recombination within the synthesis process. Grimaldi (2017) himself used a MATLAB-algorithm that modelled a syllable segmentation described by Harma (2003).[43]

Harma's segmentation method is aimed at finding syllables within bird songs and seems to perform sufficiently well for those purposes.[44] An application on the more complex human speech, however, is problematic, as the algorithm does not make a distinction between voiced or periodic sounds and unvoiced or aperiodic sounds and only regards peaks in the spectrum regardless of where in the spectrum they occur. The distinction between voiced and unvoiced parts is regarded to be an essential key to finding syllables programmatically, since "it's mostly accepted that a syllable contains a central peak of sonority, called syllable nucleus, and consonants surrounding them" (Kalinli, 2011, p. 425).

As an automated segmentation of *human* speech into syllables or "syllabic units" (Mermelstein, 1975, p. 880) is of great interest to linguistic sciences and "speech technologies"(Obin, Lamare, & Roebel, 2013, p. 2) it has been subject to numerous studies, some of which are reviewed in the following chapter.

3.2.1 State of research

Mermelstein (1975) uses a method based only on the progression of the loudness parameter. Loudness minima are weighted and sorted by means of a convex hull function and classified as syllable borders in dependency of the closeness to other minima and the exceedance of a threshold. A detailed explanation is provided in section 3.2.2.1.

Howitt (2000, Chapter 4) examines Mermelstein's method and suggests several adjustments.

Xie & Niyogi (2006) also apply a convex hull function to a parameter called "relevant energy". The resulting minima, however, are used to locate local maxima in between two minima to serve as landmarks for syllable units. The syllable borders on the other hand are obtained by a periodicity parameter, which is derived from a normalized autocorrelation function. Another convex hull algorithm returns periodicity minima, which are used as syllable borders.

Kalinli (2011) uses a biologically inspired "auditory attention model". It is based on the extraction of four different contrast features within the spectrogram.

[43] The MATLAB-implementation of the Harma Syllable Segmentation adapted by Michael Lindemuth is available on the MathWorks File Exchange platform: https://fr.mathworks.com/matlabcentral/fileexchange/29261-harma-syllable-segmentation

[44] To find syllables within an audio file, Harma creates a spectrogram of the audio. Starting from the frame containing the overall maximum both in the time and the frequency dimension, he compares the amplitude (but not the position) of spectral maxima of left- and right-sided frames. As soon as the maxima drop under a threshold of -30 dB, the frame is not considered to be part of the syllable anymore, the previous frame is considered as the syllable border and the syllable is cut out of the spectrogram to repeat the calculation with the remaining audio (Harma, 2003).

The *Syll-o-matic*-algorithm developed by Obin et al. (2013) uses a voiced/unvoiced estimation to find the syllable nuclei. After filtering the audio into 40 Mel-frequency bands, Obin et al. (2013) create two audio features called *specific loudness* and *VUV*. The specific loudness describes the power in each frequency band whereas the *VUV* describes the "degree of voicing" (Obin et al., 2013), which is calculated by dividing power of the "sinusoidal + noise representations" (Obin et al., 2013) in each frequency band through the total power in the same band. A sonority feature is then obtained by multiplying the specific loudness with the *VUVs*. Local maxima of the sonority parameter and local minima of the specific loudness are summed up over the frequency bands for each window. Opposed to Xie & Niyogi (2006), the *Syll-o-matic*-algorithm uses the resulting *sonority maxima count*-vector to distinguish landmark positions for syllable nuclei, whereas the other resulting *specific loudness minima count*-vector determines landmarks for syllable borders.

Formo (2013b) presents several different methods for syllable segmentation algorithms that were examined during the development of his *Orchestra of Speech*. He discusses some simple envelope following algorithms that – after having filtered the input speech signal – interpret envelope maxima as syllable nuclei and envelope minima as borders. Another approach presented is a periodicity estimation as a factor to weight the power of MEL-frequency bands. The periodicity estimation is produced as a part of a YIN pitch detection. He also explains a rather unusual attempt to split up utterances into syllables using the second of the signal's *mel frequency cepstrum coefficients*.[45] The best results for Formo's purposes are achieved using the glottal activity, extracted by a zero frequency resonator, as a filter for an envelope of the bandpassed speech signal. However, Formo's proposed methods exclude most unvoiced sounds from the speech signal and can therefore not be implemented in the same manner for our purposes.

The recordings made by Grimaldi (2017) contain only German speech. A study examining a syllable segmentation algorithm with German speech could not be found. As no code was published for the studies mentioned above, some approaches of these studies as well as some other ideas were remodeled in MATLAB to test the performance on the recordings provided by Grimaldi.

3.2.2 Grain segmentation implementation

Several different methods for a syllable segmentation were implemented, some of which were derived from or represented exact implementations of methods presented in the previous section. The most promising ones after a brief view on the results were subject to a performance test including a parameter optimization. The best one was chosen for the creation of the corpus from the speech material.

3.2.2.1 Method 1: Mermelstein

The first method is an implementation of the algorithm presented by Mermelstein (1975). The absolute values of the speech signal are low-pass-filtered with a threshold at 40 Hz (below the

[45] The ‚Mel Frequency Cepstrum Coefficients'(short MFCCs) are audio features that are often used for speech recognition. They can be a good indication of spectral characteristics that change over time. The coefficients are the result of a descrete cosine-transform of a Mel-Frequency-Warping of the signal's spectrum. (Logan, 2000)

lowest pitch of human speech). The resulting envelope is sent to a function that creates a convex hull of the envelope:

Starting at either end of the regarded speech segment and moving towards the total maximum position of the segment, the convex hull follows the energy envelope if the envelope values do not decrease. If they decrease, the convex hull stays at the same value, therefore creating horizontal lines above envelope *valleys*. The points of the maximal difference between the convex hull and the envelope itself on the right and on the left side of the maximum of the envelope are compared. The bigger of the two values (right- and left-sided difference maximum) is considered as a segmentation point, if both (a) the length of the resulting sub-segments and (b) the maximum difference (the *dip*) exceed a threshold. Resulting sub-segments are examined recursively until no further segmentation points exceed the thresholds.

The segmentation points are then passed to the output of the function if the segment's maximum exceeds another threshold.

3.2.2.2 Method 2: Syll-o-matic

The second method is based on the *Syll-o-matic*-algorithm by Obin et al. (2013), which has already been discussed in chapter 3.2.1. However, the implementation used here is unlikely to be an exact reproduction of the algorithm described by Obin et al. (2013), as many explanations in the *Syll-o-matic*-presentation are rather vague and leave room for interpretation.

3.2.2.3 Method 3: RMS + VUV

Method 3 can be regarded as a drastically simplified *Syll-o-matic*-implementation. Segmentation candidates are also extracted from an energy feature, and landmarks are found on the basis of a voiced/unvoiced estimation, but the two features are calculated differently:

The speech signal is split into frames of 1024 samples with an overlap of 512 samples. *Silent* frames with an RMS value below a certain threshold are cut out of the signal. A small fade to avoid clicks is added to adjacent frames. For every frame, the pitch or fundamental frequency f_0 is estimated via a simple autocorrelation function. An FFT calculates the frame's frequency spectrum. The relation between peaks in the spectrum and f_0 is examined: The number of peak positions within the spectrum that correspond to an expected partial of f_0 at the same position is summed up. In addition, the peaks are tracked over several frames. The number of peaks that have not changed their position more than within a certain range is summed up as well. The two sums are multiplied and represent a factor for the *VUV*-estimation[46] of the examined frame.

For each frame, the RMS value is calculated with the normalized, but unfiltered audio input. RMS minima positions that exceed a threshold and a minimal peak prominence are stored as segmentation candidates. If the maximum of the *VUV* factor in between two of those border candidates exceeds a threshold, it is regarded as syllable nucleus and the two borders serve as syllable borders.

3.2.2.4 Method 4: Glottal Pulse Envelope

This method is a similar approach as the one Formo (2013b) decided to use for his *Orchestra of Speech*. A *Glottal Pulse* or *Epoch* is extracted from the speech signal using a zero resonance frequency filter as proposed by Murty & Yegnanarayana (2008):

[46] VUV: Short for voiced/unvoiced.

First, the derivation of the speech signal is filtered twice with an "ideal resonator at zero frequency"(Murty & Yegnanarayana, 2008, p. 1608):

$$y_1[n] = -\sum_{k=1}^{2} a_k y_1[n-k] + x[n] \quad (1)$$

$$y_2[n] = -\sum_{k=1}^{2} a_k y_2[n-k] + y_1[n] \quad (2)$$

with *x* being the input vector, $a_1 = -2$ and $a_2 = 1$. The result is a filtered signal with an exponential trend. To get rid of this, the mean value of a 10ms window around each sample is subtracted from the exponential curve, leaving a "zero-frequency filtered signal"(Murty & Yegnanarayana, 2008, p. 1608). An example for the *de-trended*, filtered signal can be seen in Figure 11 on the right side.

Minima of the envelope of this signal serve as border candidates.

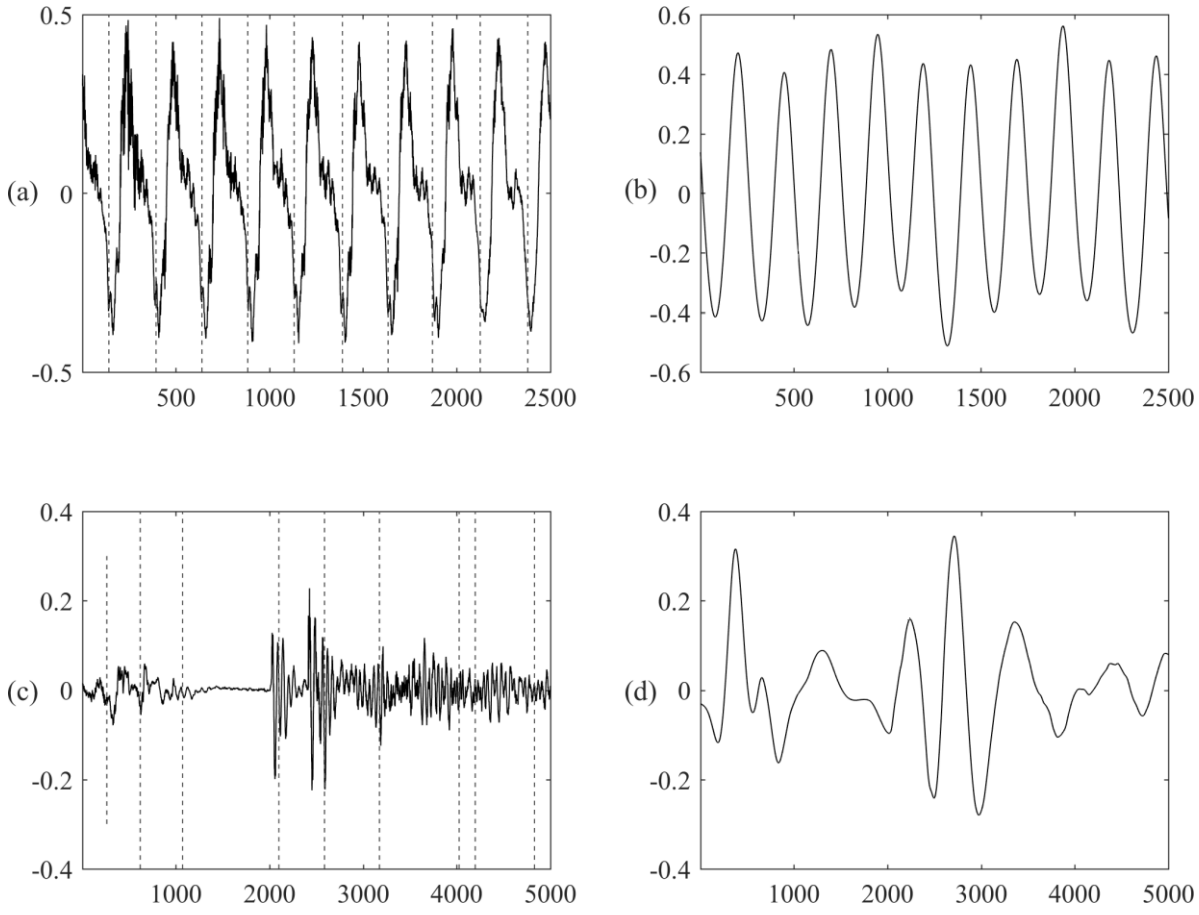


Figure 11: Example for the glottal pulse extraction. (a) shows a periodic speech signal, (b) shows the glottal pulse of the signal in (a), (c) shows an aperiodic part of a speech signal, (d) is the extracted glottal pulse of the signal in (c). The dashed lines in (a) and (c) mark the limits between periods calculated from zero crossings of the glottal pulse.

3.2.2.5 Method 5: Period Differences

All four beforehand described methods use some manifestation of an RMS feature to distinguish syllable borders. The features might be weighted differently (as in 3.2.2.2 with the

MEL-frequency bands) or be referred to a filtered version of the speech signal (as in 3.2.2.1 and 3.2.2.4), but in the end they evaluate the sum of the amplitude over a certain window.

Method 5 is a different approach: It tries to estimate the difference between two adjacent periods. This difference must be – by definition – in average big for random *noisy* signals (which for speech signals would refer to *sh* and *s*) and impulsive sounds (like *t* or *p*), whilst slow changes (as in between two vowels) should produce small values.

In the same way as for Method 4 in the previous section, a zero frequency filtered representation of the signal was created as described by Murty & Yegnanarayana (2008).

The zero crossings of the result with a positive derivation serve as division points to get single periods (see dotted lines in Figure 11). To compare two periods to one another, the mean of every period is subtracted from itself and the periods are normalized.

For all adjacent periods, the shorter period *b* is adapted to the length of the longer period *a* by Zero Padding.

For every sample, the difference between the two vectors *a* and *b* is calculated. The absolute values of the differences are summed up. Since the separation points might not always be at the exact correct position, this is repeated after vector *b* has been circularly shifted by one sample until the differences for all combinations of the two vectors were calculated. The smallest difference is kept. Noted as equation, for every period *T* the difference is

$$d_T = \min_{0 \le k \le P} \left\{ \sum_{n=1}^{P} \left| (a[n] - \bar{a}) - \left(b_0[n-k] - \overline{b_0} \right) \right| \right\} \qquad (3)$$

with *P* being the length of the longer period and b_0 the zero padded shorter period *T* or *T-1*, $\bar{a}$ and $\overline{b}$ denote the mean of the respective period.

This way, both differences in period length and sample-by-sample amplitude contribute to the overall difference value.

The maxima of the difference vector are sorted in a descending order and one by one marked as segmentation position candidates if the distance to one of the neighboring existing candidates exceeds a minimal value of 80 ms. Every segment in between two borders is rendered as a syllable if the segment is again longer than 80 ms and if the minimal difference value in between the borders is below a threshold to ensure a periodic syllable nucleus.

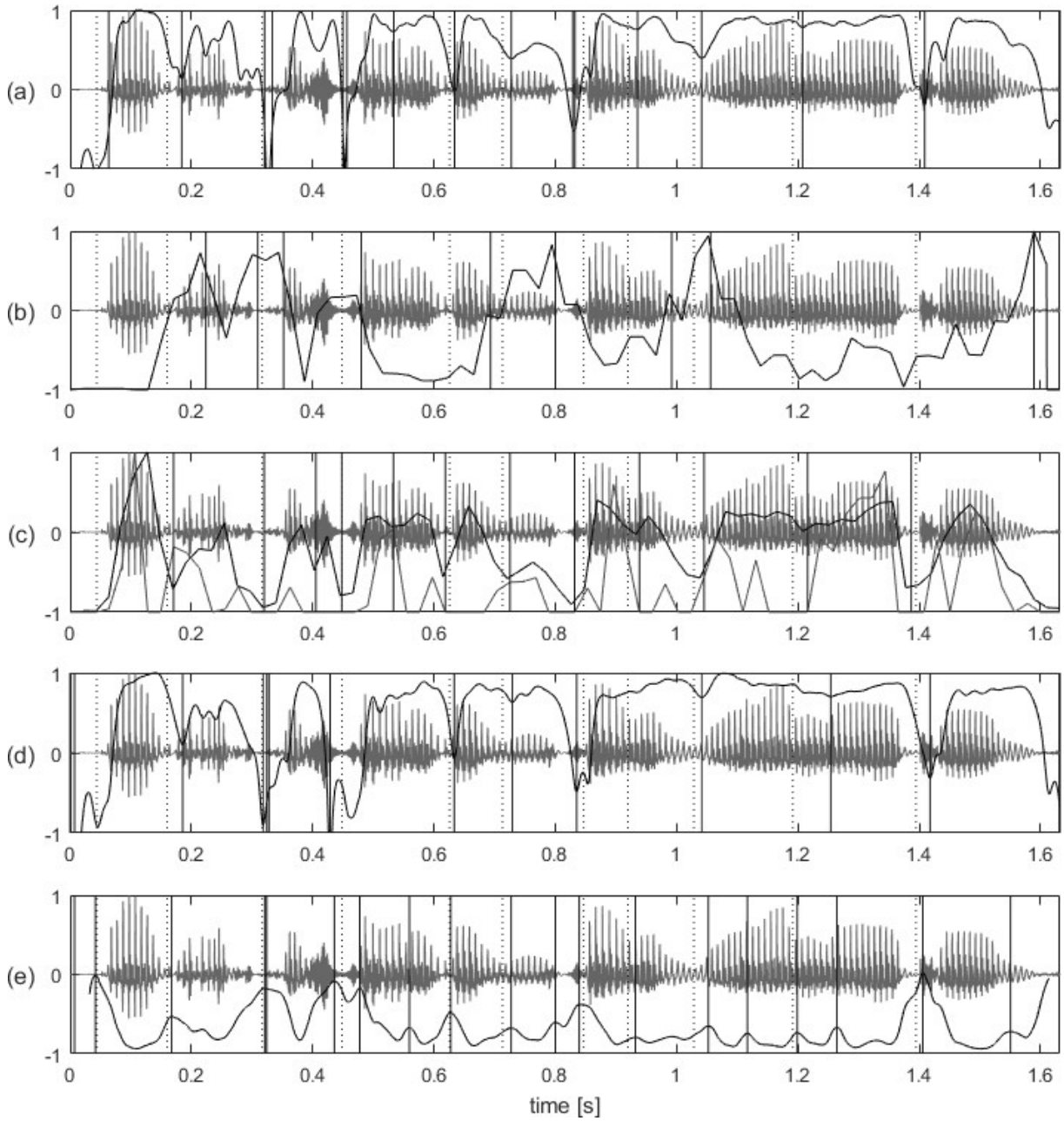


Figure 12: Comparison of grain segmentation methods. Black and grey lines mark the different parameter(s) responsible for the segmentation as explained in chapter 3.2.2.1 to 3.2.2.5. Manual borders are displayed as dotted vertical lines, automatic border estimates are full vertical lines. From top to bottom: Method 1 (a) with the energy envelope, Method 2 (b) with landmark numbers per time window, Method 3 (c) with the RMS envelope (black) and the VUV estimation (grey), Method 4 (d) with the energy envelope of the filtered signal and Method 5 (e) with the difference between adjacent period estimations.

3.2.2.6 Performance Evaluation / Parameter optimization

25 utterances from the recordings that were made for the project described in chapter 2.3.4 were chosen as test data for the 5 syllable segmentation methods. Every utterance was manually split into syllables using the *Sonic Visualiser*. The segmentation positions of the manual segmentation were then compared to the positions that were returned by the five segmentation methods. A manual border was counted as detected by the algorithm, if the distance to the closest automatic border was less than 50 ms. Bigger minimal distances are counted as *deletions*. Automatic borders are counted as *insertions*, if no manual border was found within a 50 ms-distance.

Unlike other studies in this field (Narendra & Rao, 2015; Obin et al., 2013), the TER (= Total Error Rate) as a sum of the deletions and insertions was not used to evaluate the segmentation method. As Grimaldi (2017, p. 7) states, an extraction of "lengthy elements" leads to "distinguishable repetitions of syllables", which are perceived as artefacts. Deletions should therefore be avoided as much as possible. Consequently, the performance was evaluated by a *Score* giving more weight to the deletion rate. With N being the total number of manual segmentation positions, D the number of deletions and I the number of insertions, the Score S is calculated as

$$S = \frac{2 * D + I}{1{,}5 * N}. \tag{4}$$

The lower the Score value, the closer the algorithm was to the manual segmentation.

Between 5 and 9 thresholds for every method (e.g. the minimum *dip*-height and the minimum RMS-level in between two border candidates for Method 1) were stepwise, automatically altered within a certain range to find the combination of the best performing values for each method. This was executed until no further improvement of the Score could be reached.

The results of the performance evaluations can be seen in Table 9.

Table 9: Performance evaluations of Method 1 to 5

Method	Score	Deletion	Insertion
1	33,99	16,07	18,85
3	35,85	16,47	20,83
5	47,75	12,30	47,02
4	51,19	21,23	34,33
2	64,68	25,99	38,69

After the parameter optimization, the best performance was achieved by Method 1, closely followed by Method 3. Whilst the deletion rate of Method 5 is even better than the one of Method 1 and 3, this Method tends to split the speech signal up into smaller grains (that could correspond to phonemes rather than syllables) than a manual syllable segregation does.

Method 1 was chosen to execute the syllable segmentation for the creation of the corpus. The parameters that delivered the best Score can be seen in Table 10.

Table 10: Parameters of best Score for Method 1

Parameter	Value
Silence Threshold [dB]	-20
Length Threshold [s]	0,07
Dip-Height Threshold [dB]	0,7
Low-Pass-Filter Cutoff [Hz]	5000
Minimum Energy [dB]	-21

4. Synthesis-Algorithm

This chapter covers the synthesis part of the crowd noise generation using an algorithm written in C++. It gives a brief overview on the software used to implement the algorithm as an audio plugin, which can be used with standard DAWs[47] as hosts, and presents aspects of some of the implemented classes for the three speech-texture-generating models presented in section 3.1.3. Figure 13 illustrates some of the most essential processes in the algorithm. A few code snippets of some essential functions can be seen in 0.

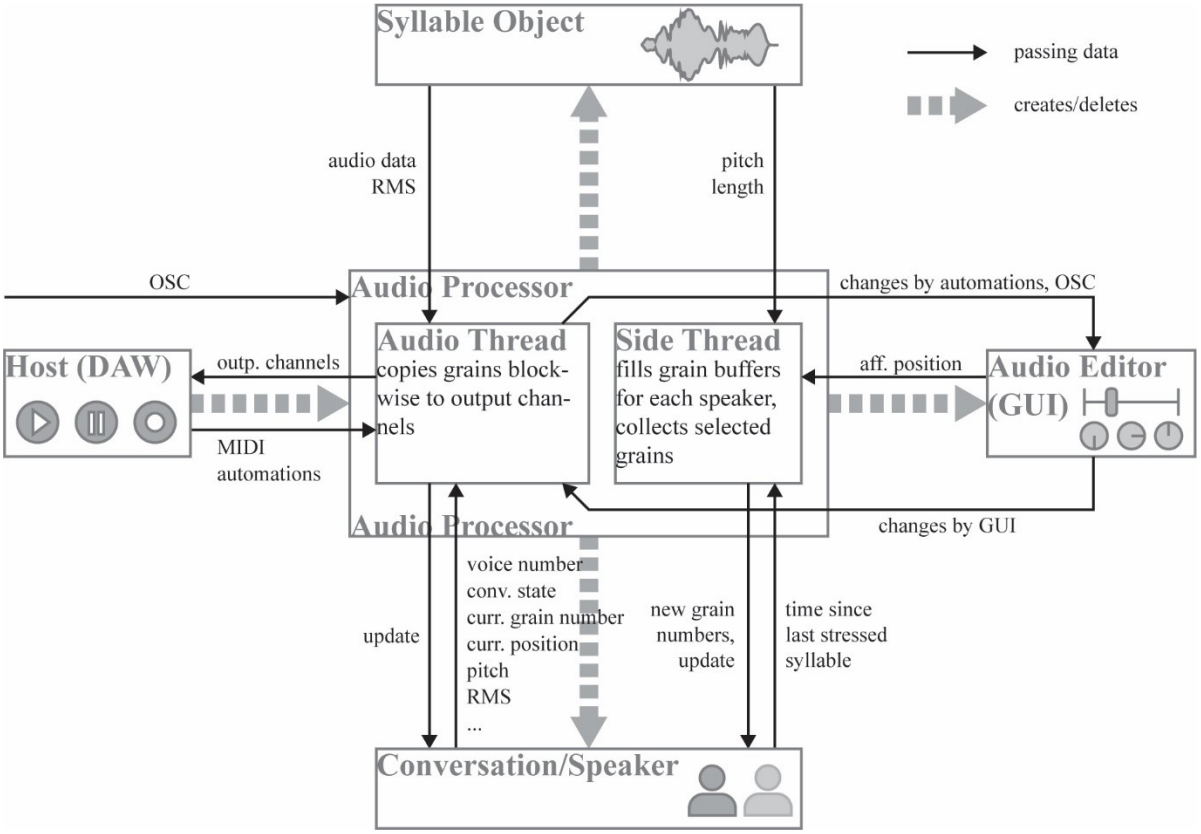


Figure 13: Simplified overview of the synthesis process implemented as an audio plugin

4.1 JUCE framework

The synthesis algorithm was implemented with the JUCE framework, a toolkit for the development of low-latency, multiplatform audio applications in the programming language C++. The name "'Juce' stands for 'Jules' Utility Class Extensions,' as it was [originally] written by one person, Julian Storer" (open-ephys, 2013/2013, sec. Introduction to Juce). The framework is today being maintained and extended by ROLI Ltd. and offers support for Microsoft, Apple and Linux systems as well as for Android. The collection of modules works as a wrapper around the user's code and allows them to compile the same code for all supported platforms, taking over the communication with the respective environment. The JUCE library contains useful classes and functions for handling audio data, which were used for the implementation of the crowd noise synthesis for (amongst others) the communication with the plugin host, to exchange parameter values from the audio processor to the user interface, to

[47] DAW = Digital Audio Workstation

store and load plugin states or to handle MIDI or OSC[48] input. (See Figure 13 for an overview on the communication between different parts of the plugin.)

The framework comes with an application called the *Projucer*, a project management tool and an application to generate basic code for specified projects, to design interfaces and to organize the modules.

4.2 Syllable object, Pitch shift

When loading a corpus from within the plugin, the original audio data of every grain is pitched up and down by a small factor (approximately corresponding to a semitone) with a linear interpolation. The small pitch-modification is supposed to slightly change the sound of the voice without making the audio data sound unnatural, thereby extending the overall variety of voices. The three versions of the grain are then stored in audio buffers within a *Syllable object*. The object also stores the pitch of the original grain (and for *Model B* other spectral, acoustic parameters describing the voice quality), which was calculated during the corpus creation, and an RMS value, which is calculated during the loading process. As only a pitch relative to the pitch of the other grains matters to the synthesis algorithm, the pitch of the altered versions does not need to be stored separately.

A very short fade-in is added at the beginning of every audio buffer to prevent clicking noises for the case of the syllable being the first one after a pause.

All Syllables objects are stored in a syllable matrix, sorted by voice.

Within the next chapters, the term *Syllable* (written with a capital *S*) refers to the C++-object described in this section.

4.3 Speaker object

A *Speaker* object represents one audio stream, which randomly (but within certain rules) plays back syllables from one voice in one of the three pitch versions. The pitch version which will be constantly used by the Speaker object is chosen randomly during the creation of the object.

Every Speaker object contains a buffer holding the indices of the Syllable objects that are to be played next. The buffer is implemented as a ring buffer and is filled on runtime by a side thread of the audio processor (see chapter 4.6).

A Speaker *state* value comprises the information, whether the Speaker is currently in the middle of *pronouncing* a syllable, fading from one syllable to another, pausing or not having any grain numbers in its ring buffer.[49]

The Speaker can be placed on any available output channels. The level for each channel can be chosen by the user, allowing *virtual* positions in between channels.

Other values stored by the Speaker object and passed to the audio processor during the render process are the original RMS value of the currently played Syllable object, the factor of RMS adaption in case an adaption was computed to avoid noticeable jumps in the signal energy, the

[48] OSC = Open Sound Control
[49] This can e.g. happen, if an area within the valence-arousal-plane is selected that does not contain grains of a voice

length of the next pause, the position within the currently *spoken* Syllable or the position within a fade.

The object offers functions to activate or deactivate the speaker, to generate pause lengths and to assert that the grain buffer is not empty.

4.4 Conversation object

Since it might be perceived as unnatural to have several Speaker objects persistently generating continuous *monologues*, a *Conversation object* was implemented.

The Conversation object consists of a Speaker object, which simulates a conversation by switching between different voices. Every available voice can be added once to the Conversation.

A selectable *chattiness*-factor with values between 0.1 and 1.0 was introduced, which affects the behavior of the Conversation in three ways:
First of all, it is used as a factor to determine the minimal length of an utterance, which was set to be twenty times the chattiness-factor without decimals. A higher chattiness means longer utterances (in average) and therefore less pauses per time unit.
The chattiness also determines the probability of a voice change after an utterance, if the Conversation contains more than one voices. If a random float value is lower than the chattiness factor, the active voice changes to a random voice of the remaining voices in the Conversation. A higher chattiness therefore raises the probability of a voice change.
The third use of the factor is for the calculation of pause lengths. Pauses are separated into short pauses (e.g. at the end of a phrase or part of a phrase) and long pauses. The pause length for short pauses L_S and for long pauses L_L in samples is calculated as

$$L_s = f_s\big(0.1 + x\,(1 - c)\big) \tag{5}$$

and

$$L_L = f_s\big(1 + 6x(1 - c)\big), \tag{6}$$

where c is the chattiness and x a random value between 0.0 s and 1.0 and f_s the sampling frequency (of the system, not of the speech material). Short pauses are therefore between 0.1 s and 1.0 s long, long pauses are between 1.0 and 6.4 seconds long. Once a pause has to be calculated, the choice between short and long pauses also depends on the chattiness. If a random floating point number between 0 and 1 is bigger than a limit l calculated as $l = 0.5\,(1 - c)$, with c being the chattiness, a short pause is triggered. Probabilities for long pauses are low for high chattiness values.

4.5 Emotion selection

The calculations to generate an affective state are the same for *Model A1* and *Model A2* (as defined in chapter 3.1) and will therefore both be explained by referring to the *Model A*.

4.5.1 Model A

In *Model A*, the Syllable objects are attached to fixed positions within the valence-arousal plane. The choice of a certain position within this plane effectuated by the user is thus linked to the algorithm using Syllables with a corresponding position. However, since the corpus size is limited, choosing only a position on a continuous scale would most likely result in no matching Syllables. The definition of a range is therefore essential to allow the algorithm to use all Syllables falling within the range. For the two *Model A* implementations, the range width can be set by the user. (Figure 14 shows the selection of a range and position on the left.) Another possibility would be to calculate the range by increasing its width until a certain amount of grains is included. Yet, the implemented method lets the user decide, in which dimension the range should be kept small to serve their purposes and in which dimension it can be increased to allow a sufficient number of Syllables to be included into the synthesis.

Model A determines the affective state by only including Syllables with positions that fall within the range, which is calculated from the currently chosen position and range width.

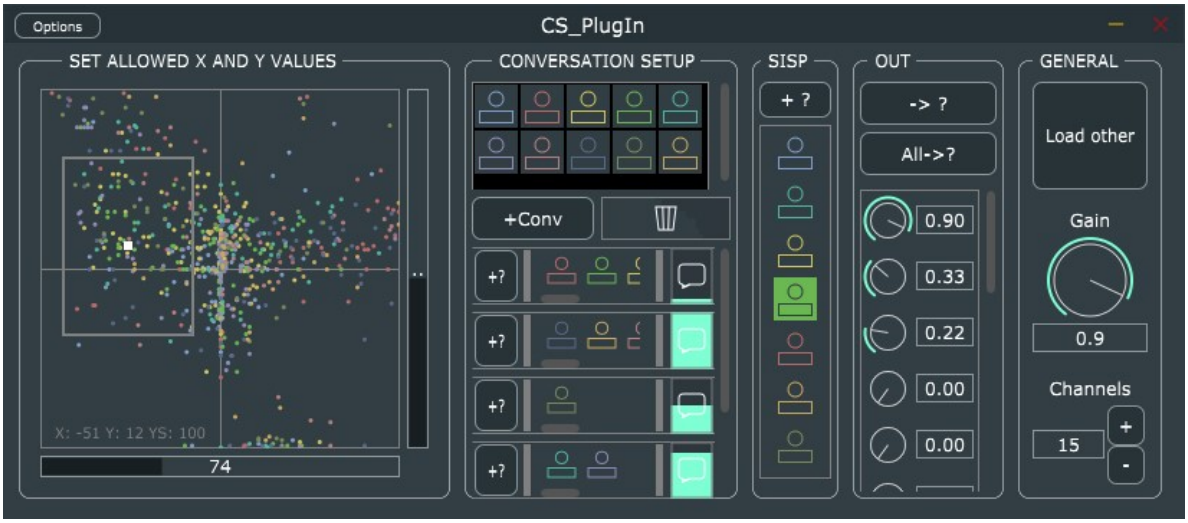


Figure 14: GUI of a plugin-implementation of *Model A* of the crowd synthesis

4.5.2 Model B

Model B generates the affective state by choosing syllables with acoustic parameters that correspond to the currently selected position within the valence-arousal plane.

As results of earlier studies showed (see chapter 3.1.1) and the examinations described in chapter 3.1.2 confirmed, the arousal dimension is highly correlated with pitch- and intensity-related acoustic parameters. Since the highest respective correlation was found for the mean of the two parameters, those two means were selected to define the *affective position* for the arousal dimension. To be included in the synthesis, the normalized mean of RMS value and mean of pitch value of the Syllable object has to be within a range defined by the intensity limits

$I_{min} = 0.7\ y$ and $I_{max} = 0.8\ y + 0.2$ and the pitch limits $P_{min} = 0.3\ y$ and $P_{max} = 0.5\ y + 0.5$, where y is the normalized position in the arousal dimension between 0 and 1.

For the valence dimension, the two parameters with the strongest correlation (see Table 3), the *Hammarberg Vocal Effort* and *Hammarberg Unstable*, determine which syllables are selected. An implementation as for the arousal dimension would have resulted in only very few (less than ten) Syllables per voice for many positions, which is why the valence limits are set relative to the mean of the values of the grains selected by the arousal dimension. The limits H_{min} and H_{max} for both normalized parameters are

$$H_{min} = a - x - t + 0.5 \tag{7}$$

and

$$H_{max} = a - x + t + 0.5, \tag{8}$$

where a is the average value, x the normalized position in the valence dimension and $t = 0.2$ a tolerance value.

4.6 Stream rendering, RMS adaption, Pauses

This chapter covers the way of (re-)concatenating the Syllable objects to streams of speech. The first issue to address is, whether certain rules have to be implemented in the concatenation of syllables to simulate a natural speech rhythm.

Research on language rhythm has mainly focused on finding features that allow for a classification or discrimination of languages, or on differences in speech rhythm between native speakers and foreign learners to improve a learning process.

Kenneth L. Pike (as cited in Arvaniti, 2012, p. 2, and Pfitzinger, 2001, p.147) introduced in 1945 a classification of languages into syllable-timed (also referred to as *machine-gum rhythm*) and stress-timed languages (also referred to as *Morse-code rhythm*), attributing an isochronal sequence of syllables or beginnings of stressed syllables respectively to the languages. He presents the English language as a reference point for a stress-timed language, while Spanish is regarded as a typical example of a syllable-timed language (Pike, 1945 as cited in Pamies Bertrán, 1999, p. 103). Abercrombie (1967, as cited in Arvaniti, 2012, p. 2) concluded, that all languages belong to either of the two classes. German has mostly been classified as a stress-timed language, meaning that the language has a tendency for a uniform temporal distribution of stressed syllables. A stressed syllable would therefore usually be followed by one or two unstressed syllables (Pfitzinger, 2001, p. 146). Numerous studies failed to support the theory of isochrony (Schmid & Dellwo, 2013, p. 110).

Newer classification methods use the percentage of vocalic parts $\%V$ and the standard deviation of the lengths of consonant parts ΔC as introduced by Ramus, Nespor & Mehler (1999) or the *Pairwise Variability Index (PVI)*. This lead to further distinctions between languages, but rules for a concatenative synthesis that does not allow *more advanced* changes[50] of the source material (due to the computational effort of generating a large number of speech *streams*) can hardly be derived from these results.

[50] E.g. time stretching or pitch changes considering formants of the speaker during runtime

Pellegrino, Coupé & Marsiko (2011, p. 544) examined a syllable rate of 5.97 syllables per second for German speakers reading a text in a (subjectively) *normal* speed, which would correspond to an average syllable length of 0.17s. Yang (1998) notes average syllable lengths from slightly under 0.1 s until slightly below 0.5 s in his studies of American English. The durations of course differ not only for different speakers and situations, but also for languages in general (Schmid & Dellwo, 2013, p. 116). Audio segments, which are shorter than 0.08 s or longer than 0.8 s are therefore rejected during the loading process. A mere concatenation of the original audio segments considers a natural syllable length and leads to a natural syllable *beat*.

To gain information on the two parameters that can be influenced in a simple way during the synthesis process, notably the RMS (by multiplication) and the pitch (by implementing rules for the selection of the next syllable), the pitch and RMS progression as examined in the phrase analysis in chapter 3.1.2.1 was repeated with relative values: The factor of the pitch and RMS progression from grain to grain was extracted to observe if these factors depend on one of the two affective dimensions and which magnitude these factors can generally have. A correlation with either of the two dimensions could not be measured, which is confirmed by the corresponding scatter plots (see Figure 15 and Figure 16).

Since no alternative, statistically established theories on the topic of simulating speech rhythm could be found, the decision was taken to alter stressed and unstressed Syllable objects in a regular way to simulate some kind of speech prosody. The fundamental frequency was chosen as stress criterion, since word stress is “almost always realized in German as an increase of F0” (Mengel, 1997, p. 12). For future research, this criterion could easily be changed by modifying the comparison operator for the Syllable object.

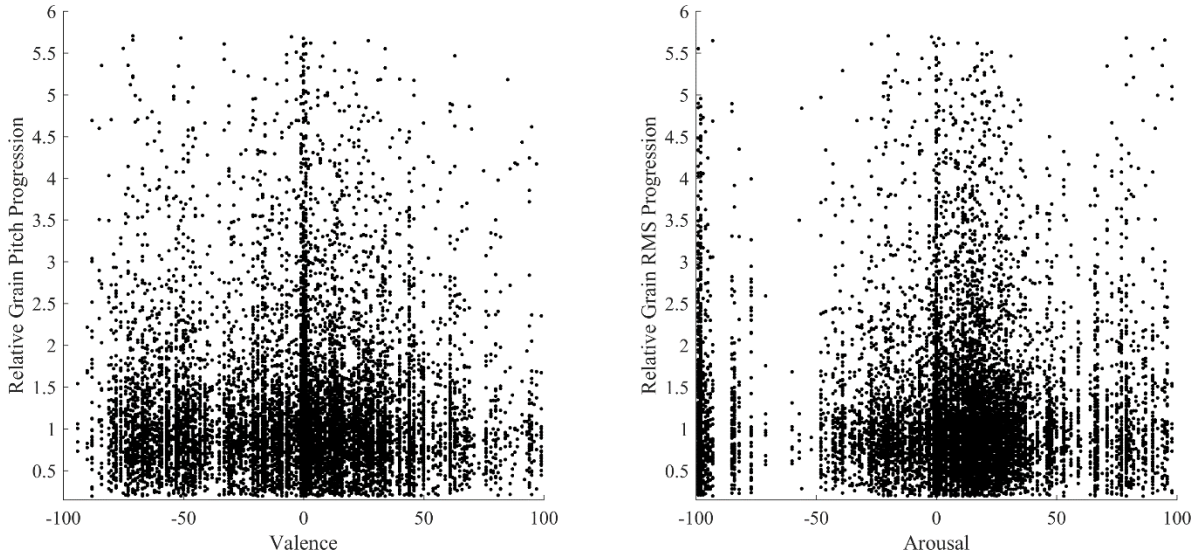


Figure 15 : Pitch factor of adjacent grains

Regarding the implementation of the C++-algorithm, two different threads manage the generation of each speech stream (as illustrated in Figure 13).

A side thread with a lower priority collects all Syllables that match the current *affective position* according to the rules specified in chapter 4.5 and fills the buffers for each Speaker or Conversation with Syllable indices. To do so, the selected Syllables are first sorted by pitch. The upper half of the sorted Syllables is regarded as containing the *stressed* Syllables, the lower one as containing the *unstressed* ones. The Syllable selection now occurs randomly following these rules: A *stressed* Syllable can only occur at the beginning of an utterance (that is after a pause, speaker activation or voice change within a Conversation) or if a certain time has passed since the last *stressed* Syllable.
After the number of Syllables in the utterance exceeds a minimal number of Syllables per utterance, which is defined by the Conversation or Speaker object (and influenced by the chattiness factor), a random float number decides, if a pause is included. The random number has to exceed a probability factor to trigger a pause, which is represented in the grain buffer as a −1. If the random number does not exceed the probability factor, this factor is multiplied with itself to increase the probability of a pause after the next Syllable has been added to the buffer.

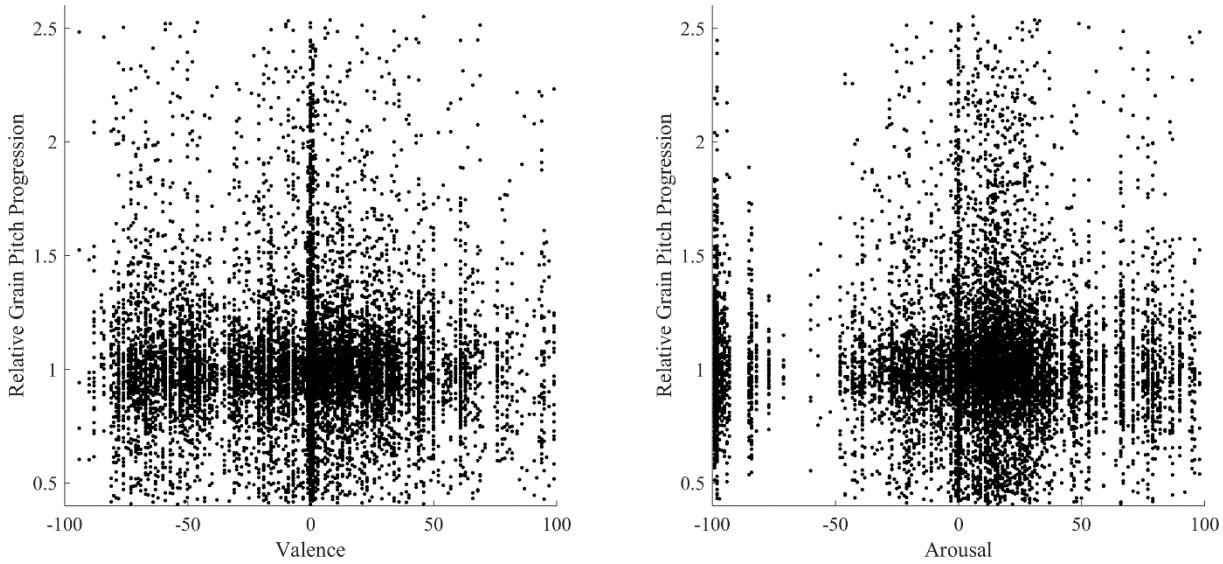


Figure 16: RMS factor of adjacent grains

However, several conditions, which were included to avoid *phrase endings* with an unnatural sound, have to be fulfilled to trigger a pause: The last Syllable has to be *unstressed*, the RMS value of the last Syllable has to be lower than the one of the Syllable before and the Syllable has to be longer than 0.15 s. (Pause lengths are calculated by the Conversation or Speaker objects themselves.)
This procedure is repeated until the buffers are full.
(More advanced conditions and dependencies for the concatenation of grains are imaginable and might lead to a higher perceived naturalness, but will also demand for a bigger corpus. Too many or too strict rules would result in a repeated concatenation of the same Syllable sequences.)

The *audio thread* is the one that is called by the host several times per second to fill the *output buffers*. It manages MIDI and OSC input data (e.g. to activate and deactivate a Conversation or

Speaker), updates parameter changed by the host or the GUI and copies the audio data that corresponds to the Syllable indices into the respective output channels. If the RMS value of a Syllable would cause an audible jump, the amplitude of the Syllable's audio data is adapted.

The plugin implementation of the algorithm allows the output channels to be routed to channels of the DAW, where they can be connected to loudspeakers or other applications or be recorded to render audio files as it was done for the listening test described in the next chapter.

5. Listening Test

5.1 Introduction

A listening test was conducted to examine, to which extent the different models can simulate affective states in the valence- and the arousal-dimension: Are stimuli which were generated with a higher or lower input value for the two dimensions also perceived as crowd noise with an associated higher or lower value for the respective dimension? In other terms: Does the *affective position input* of the three models correlate with the perceived output of the three models?

5.2 Methods

5.2.1 Stimuli

For the generation of the stimuli for the listening test, eight Conversations were created, each containing two voices. From the ten available voices, every voice was used at least in one Conversation (see Table 11). The Conversations as well as the voices within each Conversation were kept the same for all stimuli and across all three models. The *chattiness*-factor was set to 0.6 for all stimuli.

Table 11: Conversation setup and positions in listening test

Conv	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Voices	0, 1	2, 3	4, 5	6, 7	8, 9	0, 5	1, 6	2, 7
Az-Angle	0°	45°	90°	135°	180°	225°	270°	315°

For *Model A1* and *A2*, the two dimensions were evaluated separately, via separate stimuli. This is due to the fact, that some areas in the affective space were not covered by enough Syllables to create a synthesized output. Figure 17 illustrates the ranges for the respective generation of stimuli: The stimuli for *Model A1* and *A2* were created with elongated rectangles as ranges, which were moved along the axis in constant intervals. The range was set to the maximum for the dimension that was not to be evaluated with the stimulus, and set to a constantly small range for the dimension which should be evaluated with the stimulus.

The output channels, representing the speech stream of one Conversation each, were convolved with eight HRTFs[51] to be horizontally[52], evenly placed around the listener with azimuth angles in steps of 45° starting at 0° (which is supposed to be directly in front of the listener). The positions of the Conversations were kept the same for all stimuli and across all models. The convolved signals were summed up in a stereo audio file.
Following this procedure, 11 stimuli were created for each dimension for both *Model A1* and *A2*.

For the stimuli generation of *Model B*, 21 random positions on the valence-arousal plane were generated in a way that ensured, that on a grid with 21 lines in each dimension, every line or value was used once. The resulting positions can be seen in Figure 17 on the right side. These

[51] HRTF = Head related transfer function
[52] With an elevation angle of 0°

positions were used as input positions for *Model B*. The output channels were convolved in the same way as the outputs of the models *A1* and *A2*.

Since only one HRTF was used, the binaural perception is evidentially far from being perfect for every listener. Yet, this does not constitute a problem, as naturalness of the stimuli was not examined and the sole purpose of the convolution was to increase the *immersiveness* of the stimuli, to provide a setting that is comparable to one that could be offered by a VR-application (which would be an example for a usage of a crowd noise synthesis) and to simply make the participation a little more interesting for the listener.

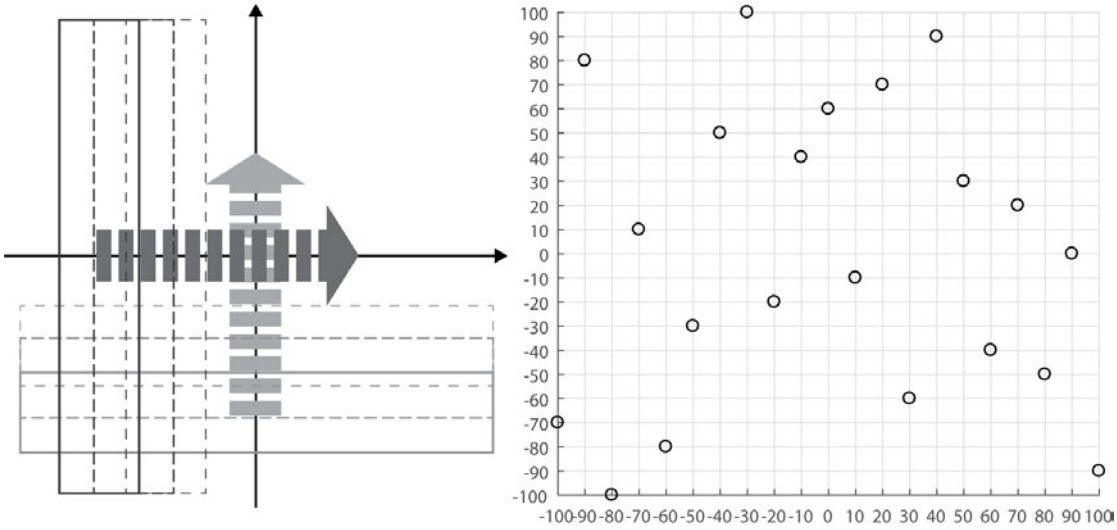


Figure 17: Stimuli generation, left: Selection ranges for valence stimuli (dark gray) and arousal stimuli (light gray) of Model A, right: *affective positions* as input for rendering stimuli of Model B

5.2.2 Subjects

50 participants of age between 20 years and 66 years and an average age of 37.4 years took part in the listening test. 31 of the subjects (62%) were male, 19 subjects (38%) were female. The majority with 30 total subjects (60%) marked German as their native language, followed by 19 subjects (38%) with French as native language. 3 subjects (6%) with English as native language, one subject with Polish and one with Dutch as native language took part. (Multiple selection was enabled here.)

5.2.3 Setup

The 65 generated stimuli were included in an online survey and presented to the participants in 3 question groups, which were each displayed on a separate page. On every page, the advices for the evaluation of the stimuli were explained on the top, followed by the list of stimuli (presented as a small audio players using the HTML5 audio tag) accompanied by one or two sliders with a *continuous*[53] scale from -10 to 10.

Question Group 1: Contained the stimuli of *Model A1* and *A2* (11 stimuli each) carrying different ranges of the valence dimension. For every stimulus, the participant had to evaluate

[53] Not continuous in a *mathematical way*, the step size was 0.1

the perceived, average affective state of the crowd on a scale from “very negative” to “very positive”.

Question Group 2: Contained the stimuli of *Model A1* and *A2* (11 stimuli each) carrying different ranges of the arousal dimension. For every stimulus, the participant had to evaluate the perceived, average affective state of the crowd on a scale from “very calm” to “very aroused/agitated”.

Question Group 3: Contained the stimuli for *Model B*. For every stimulus, the participant had to evaluate both the perceived arousal and the perceived valence of the crowd resulting in 21 different samples per dimension.

The three groups, as well as the questions within the groups, were presented in a random order.

5.3 Results

Participants used the presented scale from -10 to 10 to different extents: Some used the whole range, some only a part of it, often – but not always – spread around the neutral, middle value 0. The perceived valence and arousal values of the presented stimuli were therefore z-transformed, computing a standardization by subtraction of the mean and division through the standard deviation. The z-transformation was chosen, as not a comparison of absolute ratings of *affective positions* across different models, but the differentiation of *affective positions* within each model is of importance here. The original data is illustrated as scatter and box plots in Figure C-3 and Figure C-4 in Appendix C. The standardized results are shown as box plots in Figure 18 or as scatter plots in Figure C-2 in Appendix C.

The perceived *affective positions* were examined regarding a linear correlation with the *affective position* input of each model. Results can be seen in Table 12.

Table 12: Linear Correlation coefficients and p-Values for the perceived valence and arousal values of each model, for the original and standardized data

	Model	**Valence**		**Arousal**	
		Corr.-Coeff.	**p-Value**	**Corr.-Coeff.**	**p-Value**
original	*Model A1*	0.5741	< 0.0001	0.5968	< 0.0001
	Model A2	0.5610	< 0.0001	0.6297	< 0.0001
	Model B	-0.0396	0.1998	0.4670	< 0.0001
standardized	*Model A1*	0.6262	< 0.0001	0.6814	< 0.0001
	Model A2	0.6161	< 0.0001	0.7153	< 0.0001
	Model B	-0.0521	0.0913	0.5024	< 0.0001

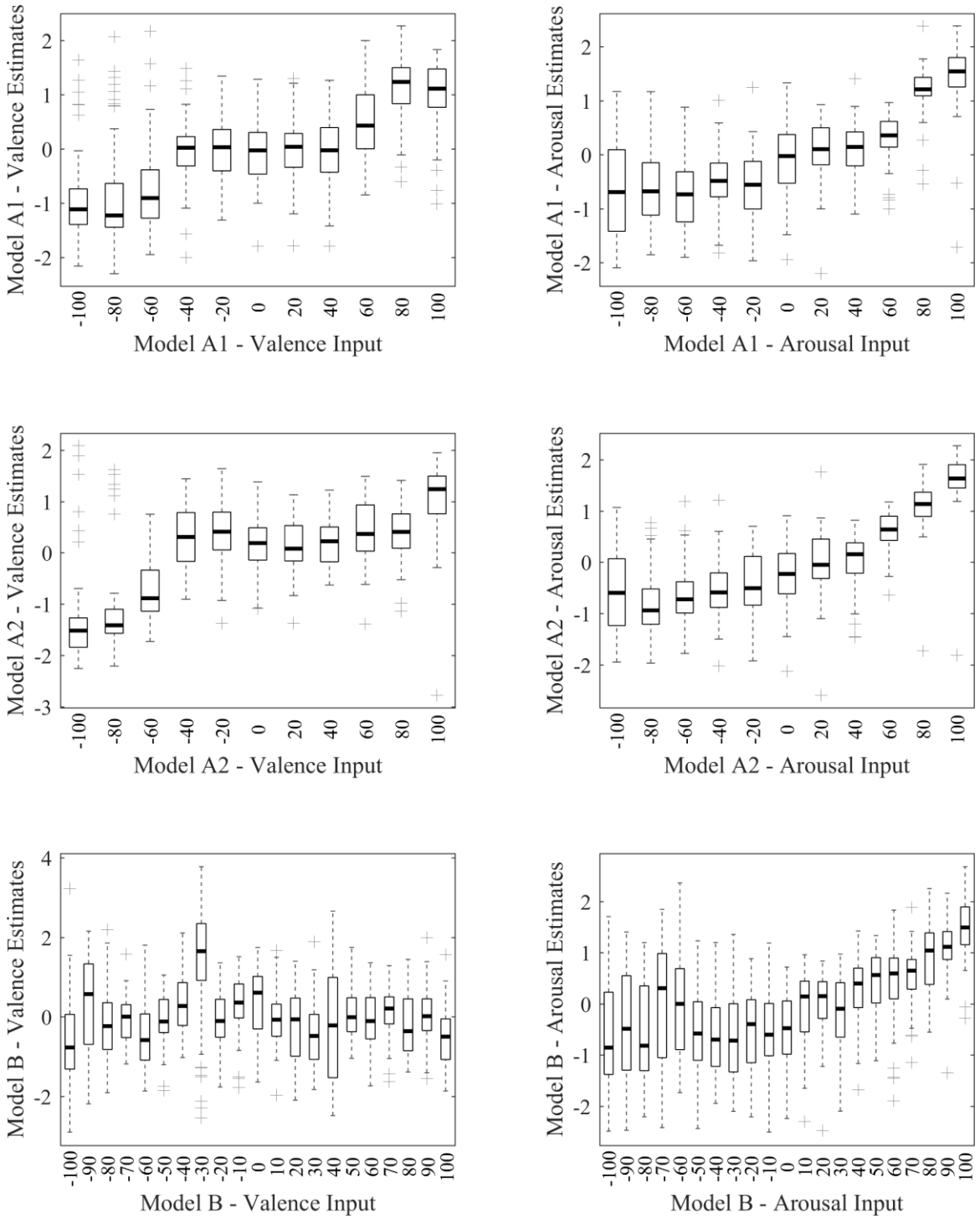


Figure 18: Standardized perceived valence (left) and arousal (right) values for the three models, the thick, central, horizontal line marks the median, bottom and top edges of the boxes indicate the 25th and 75th percentiles, whiskers mark the range of the data that is not considered as outlier, outliers are marked as crosses.

Evaluations of outputs of *Model A1* and *A2* show a medium to strong correlation of input values and perceived values in the arousal dimension, *Model B* only shows a weaker, medium correlation in the arousal dimension.

As visible in Figure 18 and Figure 19, the standard deviation for arousal estimates increases towards lower arousal input values. Especially evaluations of *Model B* show a clear decrease of the standard deviation with growing arousal input values. The correlation is measurably higher when low input values (e.g. under -50) are not included in the analysis (see Table 13).

In all comparisons in the arousal dimension, evaluations of *Model A2* show slightly stronger correlations than those of *Model A1*.

Regarding the valence dimension, estimates of the outputs of *Model A1* and *A2* show equally, medium high correlations with the valence inputs. For *Model A1*, stimuli with a valence input of between -40 and +40 were rated as approximately equally *neutral*, responses for *Model A2* show an evaluation at roughly the same level for input values between -40 and +80. For both models, there is a small gap between the constant evaluations for input values above -50 and the lower evaluations for input values below -50 (see Figure 18). Evaluations of *Model B* show no correlation with the valence input (see Table 12).
For *Model A1* and *A2*, the standard deviation of evaluations in the valence dimension is higher for valence input values lower than -50 than for the rest of the input values.

Table 13: Correlation coefficients and p-Values for arousal dimension without stimuli containing whispering (only stimuli with arousal input values higher than -50)

	Model	**Corr.-Coeff**	**p-Value**
original	Model A1	0.6216	< 0.0001
	Model A2	0.6714	< 0.0001
	Model B	0.5911	< 0.0001
standardized	Model A1	0.7240	< 0.0001
	Model A2	0.7627	< 0.0001
	Model B	0.6521	< 0.0001

As Table 14 shows, for each model, the mean of the standard deviations for arousal estimates is slightly higher than for valence estimates regarding the original results. Standard deviations are similar for the same dimension across the different models.

Table 14: Mean of standard deviations of perceived valence and arousal values

	Model	**Std. Valence**	**Std. Arousal**
original	Model A1	2.9236	3.5451
	Model A2	3.1850	3.7616
	Model B	3.2788	3.6586

Possible differences of the evaluations between participants who marked German as native language and participants with French as native language were examined graphically with the Figure C-5, Figure C-6 and Figure C-7 in the appendix on pages 95-97, but not further statistically analyzed.[54]

The results are discussed in the next chapter.

[54] No participant marked both languages as native language, only one participant marked neither of the two languages as native language, a category *other languages* was therefore not considered

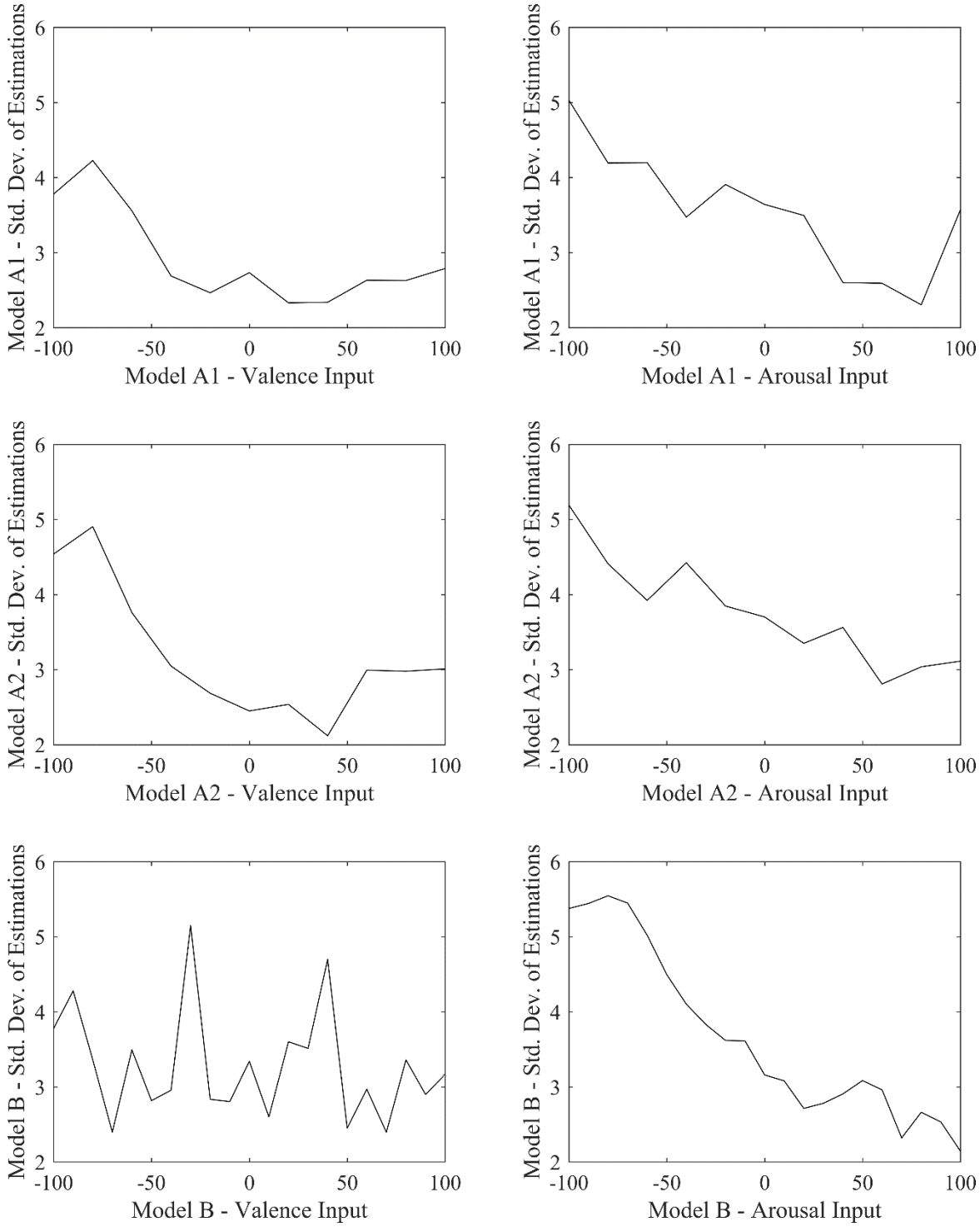


Figure 19: Standard deviations of perceived valence (left) and of perceived arousal (right) of the three models for the original (unstandardized) results

5.4 Discussion Listening Test

Regarding the evaluations for the stimuli one by one, participants had generally rather different perceptions on how to evaluate the valence and arousal of the affective state of the synthesized crowd. Of course this is not only a consequence of the synthetically generated stimuli, but also due to the complexity of the task itself and differences in individual perceptions.[55]

Looking at the sum of responses, however, for only one examined *dimension* the input value did not have an impact on the perception of the output signal: The valence dimension of *Model B*. It seems that all other dimensions can accurately be modelled with the described methods.

No evident differences in the evaluations could be found between ratings of German and French subjects.[56]

The following two sections cover the results for each dimension in further detail and suggest ways to improve conspicuous characteristics revealed by the listening test. A general discussion of methods with propositions for future research follows in chapter 6.

5.4.1 Valence Dimension

Although no *convincing* correlations (with a medium correlation coefficient of 0.5 or higher) were found between the examined acoustic parameters and the affective dimension of valence in chapter 3.1.2, an automated positioning of utterances of *Model A2* leads to a *performance* of the model in the valence dimension that is comparable to the one of the manual placement of *Model A1*. Since the regression and classification models that computed the automatic placement were trained with the very same dataset, it is to be examined by future research, how well this model performs with new speech data and *unknown* voices.

As outputs of both *Model A1* and *A2* with valence inputs in the range from -50 to +50 were judged as roughly containing the same valence, utterances that were judged to fall within this range during the corpus creation should be placed in a more *compact* way around the center in the valence dimension, that is, closer to the center.
The higher standard deviation for low valence input values for *Model A1* and *A2* indicates, that the output of those models is not clearly a crowd with a bad mood to everyone. The Outliers in form of positive rankings for those stimuli in Figure 18 confirm, that low valence inputs might generate stimuli that can be confused as crowds with a good mood. Yet, the mean of all ratings shows, that a concatenation of grains, which were previously mapped with a certain valence value, seems to be able to recreate a signal with a perceived valence similar to the one the grains were assigned with. One reason for the outliers could be, that the corpus was entirely made up of recordings by amateurs acting out scenarios in groups. The authenticity of the acted emotions might have suffered from these circumstances. A connection with the native language of subjects can be ruled out: Evaluations of both German and French subjects show the same disagreement in the discussed valence range (see Figure C-6 and Figure C-7 in Appendix C).

[55] When – for example – Schröder (2004, p. 107) let seven subjects place affective utterances in a two-dimensional activation-evaluation space, an *inter-rater* pairwise correlation of the ratings was achieved of between 0.50 and 0.88, with a generally higher correlation for the evaluation dimension than for the activation dimension.
[56] No statistical method to establish significant differences was computed here, this statement refers only to a mere observation of plots of the results for the different native languages (see Figure C-5 to Figure C-7 in Appendix C).

Model B is not able to synthesize affective states of different valences using the Hammarberg Indices as criteria for a grain selection as described in chapter 0. In order to enable a grain selection or grain concatenation with the ability to synthesize crowd noise with a specific valence, a correlation between acoustic parameters or a combination of acoustic parameters and the valence dimension comparable to the one existing for the arousal dimension has to be found.

5.4.2 Arousal Dimension

All three developed and evaluated models are able to portray the dimension of arousal to a certain extent. *Model A2* performs best in this dimension, which means that the output created by concatenating grains from automatically placed utterances is perceived as slightly more coherent with the arousal input value as it is with using grains from utterances that were manually placed. This could be due to the fact, that the manual placement was computed by only one *expert*. Letting several persons estimate the position of the original source material and taking the mean of these positions could improve the correlation coefficients for *Model A1*.

There is a striking disagreement of subjects on the arousal values for low arousal inputs in all three models. One problem that all models seem to have is processing whispered speech. The concatenation of grains from whispered speech does not seem to be able to preserve the affective state of the source material. The whispered utterances used within the corpus were calmly spoken explanations of the speakers' last holidays, thus containing a low arousal and a slightly positive valence. Both the manual placement of *Model A1* and – as a result of the low RMA value – the automatic placement of *Model A2* mapped these grains in the low end of the arousal axis. Evaluations on the outputs using these grains, however, were spread across almost the whole range of the presented scale.[57] Whispering is not automatically connected to a low arousal. A person can whisper something anxiously or happily (with a medium level of arousal) just as well as sleepily (which corresponds to a low level of arousal) or furiously (with a high level of arousal). It seems that some subjects perceived the whisper of the synthesized crowd as an agitated, maybe even nervous whisper. To avoid this *confusion*, future synthesis implementations can either exclude whispered speech entirely (which could for example be done automatically via the proportion of voiced speech within each grain) or further research should be done on how to treat whispered speech grains in order to preserve the original affective state within a concatenative synthesis.

The elevated means of the perceived arousal for *Model B* at input values -70 and -60 can be explained by the coincidental, but frequent occurrence of inhaling noises within the two generated stimuli, which presumably added a certain tension or excitement to the crowd noise. As the generation of grains for all three models is computed automatically, an occurrence of more *salient* grains cannot be ruled out. A manual elimination of problematic grains from the corpus would solve this issue.

To sum up, the listening test reveals, that – with some adaptions – the presented methods for *Model A1* and *A2* seem to be able to offer an effective implementation of a crowd synthesis with the valence-arousal model as a handle to choose an affective state. *Model B* could provide this handle over the affective dimension of arousal. The following chapter discusses

[57] This refers especially to the unstandardized responses as illustrated in Figure C-4 in Appendix C.

improvements for both the implementation of the affective model used here and the implementation of the concatenation to generate the crowd noise.

6. Discussion

It is generally arguable, whether placing affective states on a two-dimensional space makes much sense. It was chosen here to simplify the user interaction and to allow smooth transitions from any affective state to another. In this way, a relatively easy control over a wide range of affective states is offered. But the approach might deny and blur the actual complexity of human emotions: The affective states *anger* and *fear*, for example, can be placed on identical positions in this model, yet they feel very different, and most people would probably agree on the statement, that a scared person can *sound* very different from an angry person. This is directly linked to the problem, for why *Model B* failed to illustrate a positive or negative valence: Just because humans are able to map utterances to an *affective position* within a two-dimensional space with a *high level of agreement* by listening to them (Schröder, 2004, p. 107), does not necessarily mean, that there are acoustic correlates for the dimensions of this affective space. Humans can compute the mapping by (subconsciously) classifying the utterance to a distinctive affective state and deriving the *affective position* from this affective state and not directly from acoustic parameters of the utterance. Future research might find a combination of acoustic features that convincingly correlates with the affective dimension of valence, but nothing indicates up until today, that this correlation exists. Depending on the purpose of the synthesis, a different model, e.g. one that classifies utterances as distinctive affective states and lets the user select from a list of affective states and a percentage, to which it should be mixed with neutral audio material, might make more sense. The presented work shows, however, that an implementation of a concatenative synthesis with the valence-arousal model as handle for choosing an affective state is *possible*.

An improved version of the synthesis with the two-dimensional model of affect as it is presented here requires new speech recordings that cover all areas of the given affective space. Grimaldi's (2017) recordings generated a big amount of neutral, natural and spontaneous speech, but requests to act out affective scenarios in front of the other group members provoked often unconvincingly acted out speech with a noticeable ironic or even sarcastic undertone. The resulting incompleteness of the speech material in terms of *affective positions* had a negative impact on both the corpus creation and the extent to which the implemented models of synthesis could be tested with a listening test.[58]

The use of a different recording setup is therefore proposed: Each (amateur or professional) actor[59] should be recorded separately in an anechoic chamber. This actor should interact with a moderator in the chamber, who plays a counterpart for numerous scenarios. The counterpart should be played with a certain level of commitment to the scenario, allowing the actor to be more *immersed* in the scenario and possibly provoking more extreme and more authentic reactions of the actor. Having only the actor and the moderator in the chamber reduces cross-talk and background noise, which increases the percentage of the recordings that can be used

[58] The lack of grains in some areas of the affective space forced a range selection as computed in chapter 5.2.1, that is the selection of the full range for the dimension other than the one examined. This might not be a range a user would normally select to get a natural result, since grains from very different affective states are mixed together in the concatenation. An even grain distribution would e.g. allow a stimuli generation with squared selection ranges aligned in a matrix, allow to evaluate both dimensions simultaneously and to get a more profound insight in which areas of the affective space the synthesis works better or worse

[59] The male version is used here and in following sentences to keep the explanation simple. It is meant to refer to actors/actresses of all sexes.

for the creation of the corpus. As discussed in section 5.4.2, whispered speech could be excluded from the recordings or treated differently during the synthesis process to avoid a discrepancy between original *affective position* and perceived *affective position*.

With the new recordings, the examination of a correlation of acoustic parameters with the position in the affective space from chapter 3.1.2 should be repeated. A brief manual on how to use the existing MATLAB files for the respective examination is given in Appendix A.I .
To improve *Model A2*, the regression and classification models of *Model A2* should be trained again with the more *complete* set of recordings.
Attempting to receive a convincing[60] correlation of acoustic parameters with the valence dimension, new acoustic features could be included in the study. One possible example could be the examination of the utterances in a *tense/lax continuum* as proposed by Murphy, Yanushevskaya, Ní Chasaide, & Gobl (2017), a study that was published too recently to be included in the present master thesis. New findings can be applied to a new regression and/or classification model for the automated placement for *Model A2* or the grain picking algorithm in *Model B*.
As already mentioned in section 5.4.1, the performance of the regression and classification model should be examined for speech data other than the one it was trained with.

In addition to a general examination of the perceived naturalness of the generated crowd noise, numerous aspects of the synthesis allow for future research to examine their impact on the perceived naturalness:

The different syllable segmentation algorithms were evaluated here by comparing positions of the automatically generated borders to the actual, manually labelled syllable borders, hereby assuming that the manual borders are the best separation points to generate grains for the basis of a natural sound of the synthesize crowd noise, which is not necessarily true. This method was chosen here for reasons of simplicity. A more *output-driven* approach would be to have subjects rate the perceived naturalness of stimuli generated with different segmentation algorithms.

In the present synthesis, stress was merely imitated by pitch differences. A syllabic prominence descriptor that uses more than just the pitch to recombine grains might increase a perceived naturalness. E.g. Mengel (1997) states, that the length of vocal parts of syllables plays an important role for stress in the German language, values for the lengths could be taken from Yang (1998). Characteristics of the prosody of utterances could be examined over different affective states to create a more advanced model for the recombination of syllables.

In order to gain better transitions from one grain to another, a voice quality feature could be extracted from the beginning and the end of each grain to only link grains that fit to each other from a spectral point of view.

As already discussed, the density of grains over the affective space varied a lot using Grimaldi's (2017) recordings. At the neutral position, the density was very high. Bringing the other areas of the affective space to the same level regarding the grain density would drastically increase the corpus size, which in turn leads to a longer loading times and a higher memory and CPU usage, as more grains have to be analyzed during runtime. Evidentially, a bigger corpus also

[60] With a correlation coefficient comparable to the one for the arousal dimension

leads to less repetitions of grains and thus improves the naturalness of the crowd noise. An optimum has to be found between corpus size or grain density and perceived naturalness. From which point on do the improvements of the perceived naturalness gained by the growth of the corpus not *justify* the increase in loading time and memory and CPU usage anymore?

To increase the variety of voices, each grain was transposed up and down by one semitone. Vocals and consonants, or more precisely voiced and unvoiced parts were equally transposed with a linear interpolation, which can quickly lead to very unnatural speech sounds. The separation between voiced and unvoiced sounds in the loading process and the application of the interpolation on voiced parts only could improve the naturalness of the *transposed voices* or allow a transposition by more than a semitone. Filters could be used to attempt a formant shift of the original voice to further enhance the variety of voices.

The parameters used to regulate voice changes within a conversation are not based on scientific research. They were solely gained by listening to the output. A handle was then offered to the user (via the *chattiness* parameter) to adjust voice changes within defined limits. Future research could extract conversational features from recordings of conversations conducted with different affective states and build a conversation model based on the parameters gained.

Other parameters that had to be predefined but could not be set based on scientific research (since no research with an appropriate context regarding this synthesizer could be found) and therefore offer room for improvements are the length and type of crossfades between grains, the limits and factors of the RMS adaption of grains to avoid audible jumps or minimal and maximal grain lengths.

The mentioned approaches, from gathering new speech material and analyzing it over the affective dimensions to examining the naturalness of the synthesis output using different or additional concatenation methods could bring about an increase of the synthesis quality that allows a use of the synthesis algorithm as an alternative to conventional tools in sound design.

7. Conclusion

Developments in sounds synthesis are slowly turning the artificially produced sounds into a serious alternative for conventional recordings, affecting more and more areas in sound design, music production and applications for everyday life.

The work of this master thesis, the implementation of a human crowd noise synthesizer, represents a small contribution to the development of sound synthesis. The implementation of a concatenative human crowd noise synthesis allows the generation of perceivably different affective states along two examined affective dimensions, namely valence and arousal. The mere concatenation of random grains previously mapped to a specific range along one of the two dimensions of the valence-arousal-plane is able to recreate a human crowd noise with an affective state that is judged differently – concerning the respective dimension – from a crowd noise using a different range of grains within this dimension, even without applying further concatenation rules for different affective positions in the affective space. But only the use of a previously defined affective position for every grain as descriptor for the audio unit allows a discrimination of affective states along both dimensions.
For using acoustic properties as descriptors for audio units and an algorithm that picks grains with certain acoustic properties corresponding to a requested affective state, current knowledge does only allow the generation of different affective states over the dimension of arousal.

As next steps, it is proposed to acquire more speech material in order to receive an even distribution of speech material across the valence-arousal coordinates and to examine the perceived naturalness of the synthesis output for different grain segmentation and concatenation methods. These and numerous other approaches discussed in the previous chapter could improve the quality of the synthesizer to a level that allows its use in audio productions such as sound design tasks and (interactive) sound installations.

IV. BIBLIOGRAPHY

Arvaniti, A. (2012). The usefulness of metrics in the quantification of speech rhythm. *Journal of Phonetics*, *40*(3), 351–373. https://doi.org/10.1016/j.wocn.2012.02.003

Bachu, R. G., Kopparthi, S., Adapa, B., & Barkana, B. D. (2010). Voiced/Unvoiced Decision for Speech Signals Based on Zero-Crossing Rate and Energy. In *Advanced Techniques in Computing Sciences and Software Engineering* (pp. 279–282). Springer, Dordrecht. https://doi.org/10.1007/978-90-481-3660-5_47

Beller, G., Hueber, T., Schwarz, D., & Rodet, X. (2006). An Overview of Talkapillar. Retrieved from https://pdfs.semanticscholar.org/25ca/98437413986442452bb68852928ca6f6ecb1.pdf

Biassoni, F., Balzarotti, S., Giamporcaro, M., & Ciceri, R. (2016). Hot or cold anger? Verbal and vocal expression of anger while driving in a simulated anger-provoking scenario. *SAGE Open*, *6*(3), 2158244016658084.

Breazeal, C. L. (2002). *Designing sociable robots*. Cambridge, Mass: MIT Press.

Burkhardt, F., & Sendlmeier, W. F. (2000). Verification of acoustical correlates of emotional speech using formant-synthesis. In *ISCA Tutorial and Research Workshop (ITRW) on Speech and Emotion*. Retrieved from http://www.isca-speech.org/archive_open/speech_emotion/spem_151.html

Candland, D. (2003). *Emotion*. iUniverse.

Cannam, C., Landone, C., & Sandler, M. (2010). Sonic visualiser: An open source application for viewing, analysing, and annotating music audio files. In *Proceedings of the 18th ACM international conference on Multimedia* (pp. 1467–1468). ACM.

Carlsson, S. E. (n.d.). walla, walla group, ADR group [Search Engine for sound design information]. Retrieved 30 August 2017, from http://filmsound.org/terminology/walla.htm

Carstensen, K.-U., Ebert, C., Ebert, C., Jekat, S., Klabunde, R., & Langer, H. (Eds.). (2010). *Computerlinguistik und Sprachtechnologie Eine Einführung* (3., überarbeitete und erweiterte Auflage.). Heidelberg: Spektrum Akademischer Verlag.

Cowie, R., Douglas-Cowie, E., & Sawey, M. (1995). A new speech analysis system: ASSESS (Automatic Statistical Summary of Elementary Speech Structures). Presented at the International Congress of Phonetic Sciences. Retrieved from https://pure.qub.ac.uk/portal/en/publications/a-new-speech-analysis-system-assess-automatic-statistical-summary-of-elementary-speech-structures(6617f9f6-44a1-4955-b725-0e474298b704)/export.html

Cowie, R., Douglas-Cowie, E., Tsapatsoulis, N., Votsis, G., Kollias, S., Fellenz, W., & Taylor, J. G. (2001). Emotion recognition in human-computer interaction. *IEEE Signal Processing Magazine*, *18*(1), 32–80. https://doi.org/10.1109/79.911197

Duddington, J., Avison, M., Dunn, R., & Vitolins, V. (2017, July 18). eSpeak: Speech Synthesizer. Retrieved 1 September 2017, from http://espeak.sourceforge.net/index.html

Eichner, M., & Wolff, M. (2001, September 27). Verfahren zur parametrischen Synthese von Sprache. Dresden: Technische Universität Dresden.

Formo, D. (2013a). Orchestra Of Speech - Project overview. Retrieved 30 August 2017, from http://orchestraofspeech.com/project-summary/

Formo, D. (2013b, December 20). Orchestra Of Speech - Syllable detection [Blog]. Retrieved 11 April 2017, from http://orchestraofspeech.com/syllable-detection/

Formo, D. (2014, December 19). Orchestra Of Speech - Interface design. Retrieved 30 August 2017, from http://orchestraofspeech.com/interface-design/

Geier, M., Ahrens, J., & Spors, S. (2008). The SoundScape Renderer: A Unified Spatial Audio Reproduction Framework for Arbitrary Rendering Methods. In *In 124 th AES Conv*.

Goudbeek, M., & Scherer, K. R. (2008). Acoustic profiles in emotion–the GEMEP corpus. In *ISCA Tutorials and Research Workshop, Aalborg, Denmark*. Retrieved from http://www.isca-speech.org/archive_open/archive_papers/spkd2008/papers/spkd_040.pdf

Grimaldi, V. (2016). Parametric crowd synthesis for virtual acoustic environments.

Grimaldi, V. (2017). Parametric Synthesis of Crowd Noise in Virtual Acoustic Environments. In *Journal of the Audio Engineering Society* (p. 8). Berlin.

Harma, A. (2003). Automatic identification of bird species based on sinusoidal modeling of syllables. In *2003 IEEE International Conference on Acoustics, Speech, and Signal Processing, 2003. Proceedings. (ICASSP '03)* (Vol. 5, p. V-545-8 vol.5). https://doi.org/10.1109/ICASSP.2003.1200027

Howitt, A. W. (2000). Automatic syllable detection for vowel landmarks. Retrieved from http://dspace.mit.edu/handle/1721.1/4121

IRCAM. (2012, June 1). Corpus Based Synthesis - IMTR. Retrieved 28 April 2017, from http://imtr.ircam.fr/imtr/Corpus_Based_Synthesis

Kalinli, O. (2011). Syllable Segmentation of Continuous Speech Using Auditory Attention Cues (pp. 425–428). Presented at the INTERSPEECH 2011, Florence, Italy: ISCA. Retrieved from http://ai2-s2-pdfs.s3.amazonaws.com/fa74/295bd699d484a15c5144333fad9834d3141f.pdf

Kienast, M., & Sendlmeier, W. F. (2000). Acoustical analysis of spectral and temporal changes in emotional speech. In *ISCA Tutorial and Research Workshop (ITRW) on Speech and Emotion*. Retrieved from http://www.isca-speech.org/archive_open/archive_papers/speech_emotion/spem_092.pdf

Kröger, B. J., & Birkholz, P. (2009). Articulatory Synthesis of Speech and Singing: State of the Art and Suggestions for Future Research. In A. Esposito, A. Hussain, M. Marinaro, &

R. Martone (Eds.), *Multimodal Signals: Cognitive and Algorithmic Issues* (Vol. 5398, pp. 306–319). Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-00525-1_31

Lee, E. A. (2006). The Problem with Threads. *Computer*, *39*(5), 33–42. https://doi.org/10.1109/MC.2006.180

Lemmetty, S. (1999, March 30). *Review of Speech Synthesis Technology* (Master Thesis). HELSINKI UNIVERSTY OF TECHNOLOGY, Helsinki. Retrieved from http://research.spa.aalto.fi/publications/theses/lemmetty_mst/chap2.html

Lerch, A. (2012). Instantaneous Features. In *An Introduction to Audio Content Analysis* (pp. 31–69). John Wiley & Sons, Inc. https://doi.org/10.1002/9781118393550.ch3

Lerch, A. (n.d.). code | Audio Content Analysis. Retrieved 26 September 2017, from https://www.audiocontentanalysis.org/code/

Llimona, Q. (2015). *matlab: MATLAB Tools - Various useful functions we've put together for convenience*. Matlab. Retrieved from https://github.com/lemonzi/matlab (Original work published 2014)

Logan, B. (2000). Mel Frequency Cepstral Coefficients for Music Modeling. Presented at the In International Symposium on Music Information Retrieval. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.11.9216

Lu, L., Wenyin, L., & Zhang, H.-J. (2004). Audio textures: theory and applications. *IEEE Transactions on Speech and Audio Processing*, *12*(2), 156–167. https://doi.org/10.1109/TSA.2003.819947

MBROLA Project Development Team. (1999, December 17). The MBROLA Project - Towards a Freely Available Multilingual Speech Synthesizer. Retrieved 4 September 2017, from http://tcts.fpms.ac.be/synthesis/mbrola.html

McGilloway, S., Cowie, R., Douglas-Cowie, E., Gielen, S., Westerdijk, M., & Stroeve, S. (2000). Approaching automatic recognition of emotion from voice: a rough benchmark. In *ISCA Tutorial and Research Workshop (ITRW) on Speech and Emotion*. Retrieved from http://www.isca-speech.org/archive_open/speech_emotion/spem_207.html

Mengel, A. (1997). Das akustische Korrelat des deutschen Wortakzents. Retrieved 19 October 2017, from http://www.andreasmengel.de/pubs/essv-cottbus-97.html

Mermelstein, P. (1975). Automatic segmentation of speech into syllabic units. *The Journal of the Acoustical Society of America*. https://doi.org/10.1121/1.380738

Merriam-Webster. (2017a). affect. In *Merriam-Webster*. Retrieved from https://www.merriam-webster.com/dictionary/affect

Merriam-Webster. (2017b). fricative. In *Meriam-Webster*. Retrieved from https://www.merriam-webster.com/dictionary/fricative

Möhlmann, D. (2011). *A Parametric Sound Object Model for Sound Texture Synthesis*.

Murphy, A., Yanushevskaya, I., Ní Chasaide, A., & Gobl, C. (2017). Rd as a Control Parameter to Explore Affective Correlates of the Tense-Lax Continuum. https://doi.org/10.21437/Interspeech.2017-1448

Murray, I. R., Edgington, M. D., Campion, D., & Lynn, J. (2000). Rule-Based Emotion Synthesis Using Concatenated Speech. In *SpeechEmotion-2000* (pp. 173–177). Newcastle, Nothern Ireland, UK. Retrieved from http://www.isca-speech.org/archive_open/speech_emotion/spem_173.html

Murty, K. S. R., & Yegnanarayana, B. (2008). Epoch Extraction From Speech Signals. *IEEE Transactions on Audio, Speech, and Language Processing*, *16*(8), 1602–1613. https://doi.org/10.1109/TASL.2008.2004526

Narendra, N. P., & Rao, K. S. (2015). Robust Voicing Detection and F_{0} Estimation for HMM-Based Speech Synthesis. *Circuits, Systems, and Signal Processing*, *34*(8), 2597–2619. https://doi.org/10.1007/s00034-015-9977-8

Obin, N., Lamare, F., & Roebel, A. (2013). Syll-O-Matic: an Adaptive Time-Frequency Representation for the Automatic Segmentation of Speech into Syllables (p.). Presented at the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP). Retrieved from http://hal.upmc.fr/hal-00943799/document

open-ephys. (2013, June 18). Juce. Retrieved 6 December 2017, from https://github.com/jazlab/OpenEphys-GUI (Original work published 1 September 2013)

Oudeyer, P.-Y. (2003). The production and recognition of emotions in speech: features and algorithms. *International Journal of Human-Computer Studies*, *59*, 157–183. https://doi.org/10.1016/S1071-5819(02)00141-6

Pamies Bertrán, A. (1999). Prosodic typology: on the dichotomy between stress-timed and syllable-timed languages. *Language Design: Journal of Theoretical and Experimental Linguistics*, *2*, 103–130.

Pellegrino, F., Coupé, C., & Marsico, E. (2011). Across-language perspective on speech information rate. *Language*, *87*(3), 539–558.

Pereira, C. (2000). Dimensions of emotional meaning in speech. In *ISCA Tutorial and Research Workshop (ITRW) on Speech and Emotion*. Retrieved from http://www.isca-speech.org/archive_open/speech_emotion/spem_025.html

Petrushin, V. A. (1999). Emotion in Speech: Recognition and Application to Call Centers. In *In Engr* (pp. 7–10).

Pfister, B., & Kaufmann, T. (2017). *Sprachverarbeitung - Grundlagen und Methoden der Sprachsynthese und Spracherkennung*. Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-662-52838-9

Pfitzinger, H. R. (2001). *Phonetische Analyse der Sprechgeschwindigkeit*. Inst. für Phonetik und Sprachliche Kommunikation, Red. FIPKM. Retrieved from http://www.bas.uni-muenchen.de/forschung/FIPKM/vol38/f38_hp_1.pdf

Pigeon, D. I. S. (2013a, 2017). About myNoise.net. Retrieved 13 September 2017, from https://mynoise.net/about.php

Pigeon, D. I. S. (2013b, 2017). Babble Noise. Retrieved 13 September 2017, from https://mynoise.net/NoiseMachines/babbleNoiseGenerator.php

Pigeon, D. I. S. (2013c, 2017). The various uses of Noise Generators (aka Noise Machines). Retrieved 13 September 2017, from https://mynoise.net/howToUseSoundMachines.php

Posner, J., Russell, J. A., & Peterson, B. S. (2005). The circumplex model of affect: An integrative approach to affective neuroscience, cognitive development, and psychopathology. *Development and Psychopathology*, *17*(3), 715. https://doi.org/10.1017/S0954579405050340

Ramus, F., Nespor, M., & Mehler, J. (1999). Correlates of linguistic rhythm in the speech signal. *Cognition*, *73*(3), 265–292.

Russell, J. A. (1980). A Circumplex Model of Affect. *Journal of Personality and Social Psychology*, *39*(6), 1161–1178.

Scherer, K. R. (1995). Expression of emotion in voice and music. *Journal of Voice*, *9*(3), 235–248. https://doi.org/10.1016/S0892-1997(05)80231-0

Scherer, K. R. (1996). *Adding The Affective Dimension: A New Look In Speech Analysis And Synthesis*.

Scherer, K. R., & Oshinsky, J. S. (1977). Cue utilization in emotion attribution from auditory stimuli. *Motivation and Emotion*, *1*(4), 331–346. https://doi.org/10.1007/BF00992539

Schlosberg, H. (1952). The Description of Facial Expressions in Terms of Two Dimensions. *Journal of Experimental Psychology*, *44*(4), 229.

Schmid, S., & Dellwo, V. (2013). Sprachrhythmus bei bilingualen Sprechern. *Revue Tranel (Travaux neuchâtelois de linguistique)*, *59*, 109–126.

Schröder, M. (2001). Emotional speech synthesis: A review. In *Seventh European Conference on Speech Communication and Technology*. Citeseer. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.420.7760&rep=rep1&type=pdf

Schröder, M. (2004). *Speech and Emotion Research: An Overview of Research Frameworks and a Dimensional Approach to Emotional Speech Synthesis*. Institut für Photetik, Universität des Saarlandes.

Schröder, M. (2005, June 27). A new speech analysis system: ASSESS. Retrieved 27 November 2017, from http://emotion-research.net/biblio/CowieSaweyDouglas-Cowie1995

Schröder, M., Cowie, R., Douglas-Cowie, E., Westerdijk, M., & Gielen, S. C. (2001). Acoustic correlates of emotion dimensions in view of speech synthesis. In *INTERSPEECH* (pp. 87–90). Retrieved from ftp://134.96.191.190/papers/local/schroeder_etal2001.pdf

Schwarz, D. (2010). CataRT App Documentation. Retrieved 11 September 2017, from http://imtr.ircam.fr/imtr/CataRT_App_Documentation

Schwarz, D. (2011). State of the Art in Sound Texture Synthesis (pp. 1–1). Presented at the Digital Audio Effects (DAFx). Retrieved from https://hal.archives-ouvertes.fr/hal-01161296/document

Schwarz, D. (2017, May 3). CataRT - Real-Time Corpus-Based Concatenative Synthesis. Retrieved 28 April 2017, from http://imtr.ircam.fr/imtr/CataRT

Schwarz, D., Beller, G., Verbrugghe, B., & Britton, S. (2006). Real-time corpus-based concatenative synthesis with catart. In *9th International Conference on Digital Audio Effects (DAFx)* (pp. 279–282). Retrieved from https://hal.archives-ouvertes.fr/hal-01161358/

Strobl, G., Eckel, G., Rocchesso, D., & Le Grazie, S. (2006). Sound texture modeling: A survey. In *Proceedings of the 2006 Sound and Music Computing (SMC) International Conference* (Vol. 61, pp. 1–7). Retrieved from http://iem.kug.ac.at/fileadmin/media/iem/altdaten/projekte/publications/paper/soundtexture/8-soundtexturemodelingSMC06.pdf

Tamarit, L., Goudbeek, M., & Scherer, K. (2008). Spectral slope measurements in emotionally expressive speech. *Proceedings of Speech Analysis and Processing for Knowledge Discovery*, 169–183.

The MathWorks Inc. (2017). Matlab Classification Learner App (Version R2017a). Natick, Massachusetts: The MathWorks Inc.

Tokuda, K., Nankaku, Y., Toda, T., Zen, H., Yamagishi, J., & Oura, K. (2013). Speech Synthesis Based on Hidden Markov Models. *Proceedings of the IEEE*, *101*(5), 1234–1252. https://doi.org/10.1109/JPROC.2013.2251852

Vine, D. S. G., & Sahandi, R. (2000). Synthesis of emotional speech using RP-PSOLA. In *IEE Seminar on State of the Art in Speech Synthesis (Ref. No. 2000/058)* (p. 8/1-8/6). https://doi.org/10.1049/ic:20000325

Wulff, H. J. (2012, October 13). Soundscape - Lexikon der Filmbegriffe. Retrieved 28 November 2017, from http://filmlexikon.uni-kiel.de/index.php?action=lexikon&tag=det&id=6211

Xie, Z., & Niyogi, P. (2006). Robust acoustic-based syllable detection. In *Interspeech*. Citeseer. Retrieved from

http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.156.9546&rep=rep1&type=pdf

Yang, L. (1998). Contextual effects on syllable duration. In *The Third ESCA/COCOSDA Workshop (ETRW) on Speech Synthesis*. Retrieved from http://www.isca-speech.org/archive_open/ssw3/ssw3_037.html

V. List of Figures

VI. LIST OF TABLES

VII. LIST OF FORMULAS

VIII. APPENDIX

Appendix A Manuals

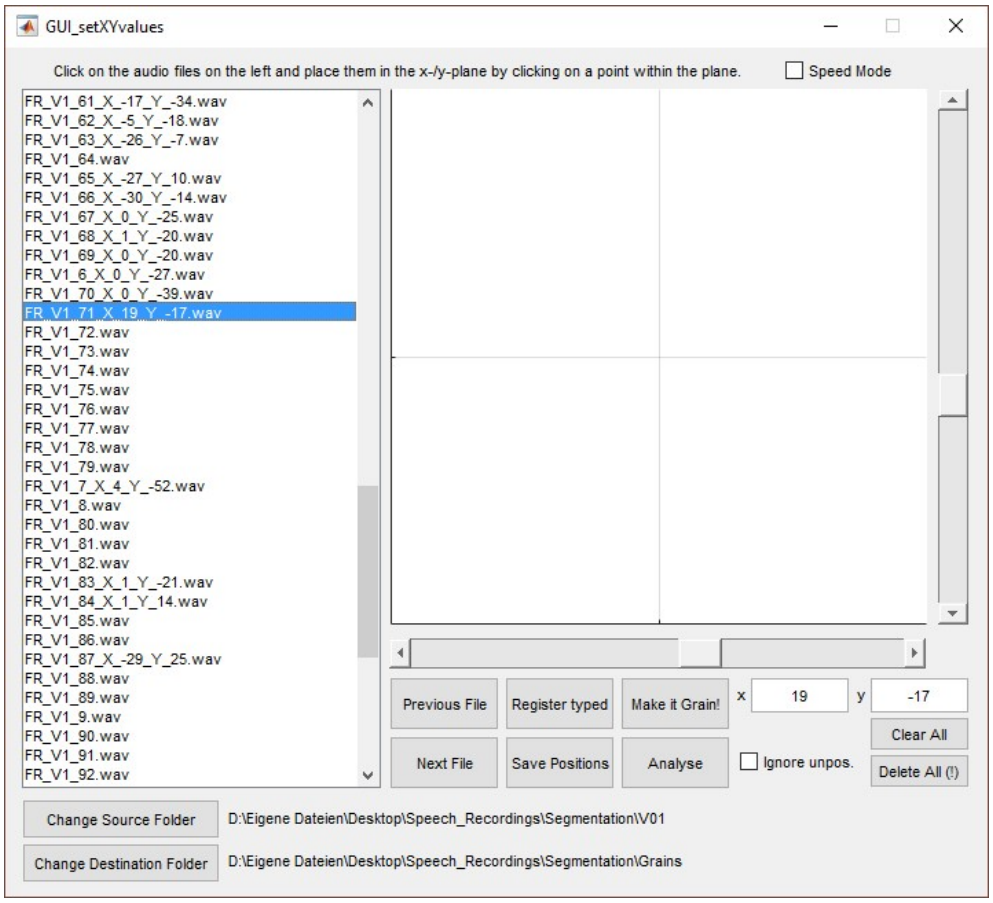
Appendix A.I Examination of correlation of acoustic parameters with affective positions

To examine correlations of the same acoustic parameters as shown in chapter 3.1 with the affective dimensions for new recorded utterances, the utterances have to be sorted by voice and stored in one folder per voice. The *affective position* has to be marked in each file name, e.g. *X_10_Y_-70_255.wav* for a valence of 10 and an arousal of -70. The algorithm specifically looks for *X_* and *Y_* in the file name to read the numbers after those characters as position values. The positioning can be done with the MATLAB *GUI_setXYValues.m* file as described in the next chapter. To start the parameter extraction and correlation examination, open the script *ParameterExtractionVoiceFolders.m* and set the path to the parent folder containing the voice subfolders. Set the target path and write the filename of one neutral file per voice (from voice 1 to x) in the indicated cell vector. Execute the script. The extracted parameters and tables containing the sorted correlation values will be saved under the target path. To change the acoustic parameters, rewrite the corresponding parts in the functions *analyseFolder5.m* and *analysePhrase5.m*

To extract parameters for the regression and classification model for an automatic mapping of grains, do the same as described in the previous paragraph but with the MATLAB script *DataExtractionForMachineLearning6.m*. To change the acoustic parameters here, rewrite the corresponding parts in the functions *analyseFolder6.m* and *analysePhrase6.m*.
The resulting table can be used to train or create a new regression or classification model. The first three columns of the resulting table are the x-value, y-value and the *affective class* between 1 and 9.

Appendix A.II Corpus Creation with MATLAB Scripts

To create a corpus for *Model A1*, recordings of utterances from different voices without reverb or any kind of room information must first be sorted by voice and stored in folders labeled *V + voice number*, e.g. *V01* for voice 1. Every utterance has to be manually mapped to a position in the valence-arousal plane. The position has to be stored in the filename. Values should be integer values between −100 and 100, with 0 being *neutrality*. The name can e.g. be extended by *_X_30_Y_-10* to store a position with an x-value of 30 and a y-value of −10.
The manual mapping can be done with the MATLAB GUI-application that was written for this task. To use it, open the MATLAB file *GUI_setXYvalues.m* and run it. Two dialogue windows appear to choose the source path (with the audio files of one speaker that should be mapped) and the target path (where the grains for the respective speaker should be stored).



All the audio files found in the source folder are listed on the left. Clicking on the file name selects the file and starts the playback of the respective file. The file can then be positioned via three ways:
A click within the coordinate system on the right representing the valence-arousal plane assigns the clicked position to the audio file. Two sliders under and on the right side of the coordinate system allow a separate evaluation of the x- and y-value. Alternatively, the position values can also be entered in two text fields labeled *x* and *y*. To register the positions entered in the text fields, click on the button *Register typed*.

The selected file can be changed via two buttons labeled *Previous file* and *Next file* or simply by clicking on the name in the list on the left. If the toggle switch *Speed mode* is switched on, the next file is automatically played and selected after a placement via a click in the coordinate system.

Before closing the application, click on *Save Positions* to add the new positions to the file names. *Save Positions* transfers the file names from the list on the left to the actual file names. The button *Clear All* deletes the positions in the list of file names on the left, the button *Delete All* actually deletes the positions that have previously been saved within the file names. In other words, *Delete All* combines a click on *Clear All* and *Save Positions*. A pop-up window asking for the confirmation inhibits accidental deletions.

The button *Make it Grain!* reads the files in the selected source folder that contain the required position information, splits them up into grains and stores them in a subfolder of the selected target folder. The subfolder is labeled with a time stamp. The files in the source folder are not deleted or changed by this operation.

The button *Analyse* analyzes the length, RMS and pitch values of the grains either in the selected target folder or in the subfolder with the newly created grains. A dialogue window will ask for which folder to analyze if new grains have been created. Grains with values that are outliers in one of the categories or don't contain one of the required values can be deleted. Again, a dialogue window asks if these files should be deleted. The distribution of RMS, pitch and length values is plotted in histograms. The toggle button *Ignore unpos.* can be activated to exclude grains without x- or y- values from the analysis. This is only important if the grains in the target folder that are analyzed have not been created by the function that is triggered by the *Make it Grain!* button.

The grain segregation can be done faster (not for each voice individually, but over all subfolders representing different voices at once) with the MATLAB function *createCorpusA1.m*. The function expects the path of the source folder (containing all the subfolders of all voices) and the target folder as inputs. The finished corpus is stored in a subfolder of the target folder labeled with a time stamp.

The corpus for *Model A2* can be created with the function *createCorpusA2.m*. In addition to source and target path, it also expects the filenames of a neutral utterance for each voice in a cell vector. So you first have to listen to some utterances and select one that comes closest to being in the center (x: 0 y: 0) of the valence-arousal plane for each voice.

If new utterances are recorded, it is recommended to also train the regression and classification model again or to generate new models. To extract the acoustic parameters from the new data, the script *DataExtractionForMachineLearning6.m* can be used. Source and target paths, as well as file names of neutral files have to be specified before execution.

The corpus for *Model B* can be created with the function *createCopusB.m*. It stores – together with the audio files – an XML-file containing absolute paths of the grains and some acoustic parameters that are used later in the synthesis process. The function only expects source and target folders. Optionally, it can create graphs showing the parameter distributions of the grains.

Appendix A.III Use of Synthesizer Plugins

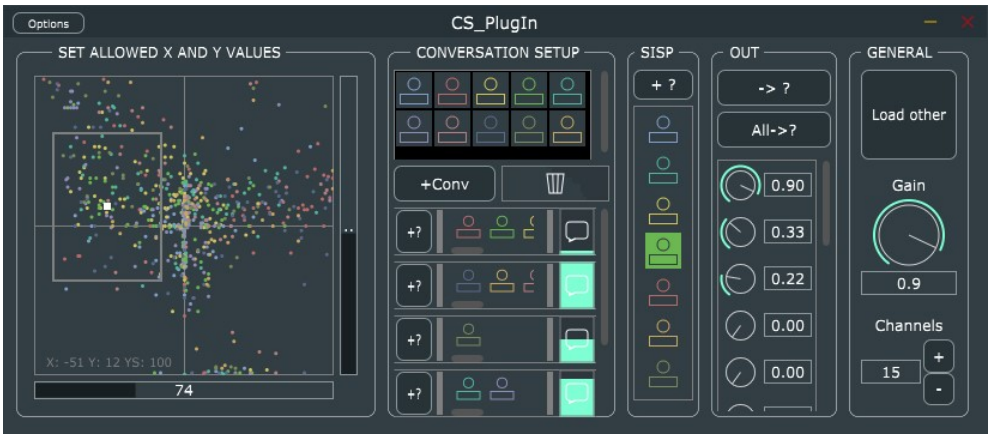


Figure A-1: GUI of *Model A* Synthesizer Plugin

The synthesizing part of *Model A1* and *Model A2* is identical and will thereby be referred to as *Model A*.

Depending on the DAW or Host used, the first thing to do after loading the Plugin is to activate all output channels, since some Hosts only activate one channel by default. For

information on how to activate the output channels, look in the manual of the respective Host.

Once the output channels are activated, the audio data for the corpus has to be loaded. To do so, click on the button *Load*. The *Model A* expects a certain data structure for the audio files to be loaded: The grains for each voice have to be stored in a separate folder. After clicking on *Load*, select the *parent folder* of the folders containing the audio data. Each subfolder within this parent folder that contains audio data will be read by the plugin as one *voice*, that is all grains within one subfolder will be stored in one vector and used as source material for one voice.
After selecting the parent folder, the Plugin scans the subfolders and reads all the audio files, applies a pitch change upwards and downwards to create three different variations (original, higher and lower) of each voice, calculates the RMSs and stores the data in buffers. This can take several minutes, depending on the corpus size. During the process, the label of the *Load* button shows *Loading*.
If you use *Model B*, an XML-file has to be selected containing the absolute file paths and the corresponding acoustic parameters for all the audio grains. The loading process is about as long as for *Model A*. The window of the Plugin can be closed during the loading process, which will continue in the background. It is recommended not to interrupt the loading process by deleting the plugin during the loading process. Adding speakers or conversations is not possible before or during the loading process. (Before loading, the plugin does not know how many voices are available.)

As soon as the loading process is finished, the buttons for adding speakers and conversations can be used. The *Load* button now shows *Load other* to indicate that a different corpus can be loaded without having to restart the plugin.

In *Model A*, the system of coordinates on the left show the *affective positions* of the grains with dots in different colors. Each color represents one voice. With two sliders on the bottom and the right of the coordinates, the width and height of the selected range can be set. The selected number for the width can be seen in the slider on the bottom. The selected height is displayed in a grey font on the bottom left inside of the coordinates, on the right side of the currently selected position. The position is indicated in the coordinates by a small white square, the selected range for the available grains is shown as a grey square around the position. If no grain exists in the selected area for a voice, the speakers and conversations with (only) that voice will remain silent.

In *Model B*, the system of coordinates is left blank, as the grains are not assigned to a position within the system. Only a white square indicates the current position.

The rest of the user interface is identical for all models.

To *add* a conversation, click on the + *conv* button. An (empty) conversation row will appear underneath the buttons. To add a speaker to a conversation, click on a speaker icon on the top area of the *conversation setup* and drag and drop the icon on the middle of the conversation area. To add a single speaker without making them part of a conversation, drag the icon over the *Speaker Panel* on the right side of the *Conversation Setup*. A right rectangle indicates a drop-zone.

Figure A-2: Drag-and-drop examples to add and delete speakers or conversations.

To *delete* a speaker, a voice of a conversation or a whole conversation, click on the respective icon and drag and drop it above the *bin icon*. The grey rectangles within each conversation panel serve as handles for selecting or dragging the conversation.

The *+?*-button on the left side of every conversation panel allow to add a random voice to the conversation. A slider on the right side, marked with a speech bubble, can be used to set the *chattiness* of the conversation. The default value is 0.6.

Once a conversation or speaker is selected by clicking on it (the background color changes), its routing can be adjusted via the panel labeled with *out*. Every rotary knob represents one output channel and sets the level, with which the audio of the selected item will be added to the channel. The button on the top of the routing panel places the selected item on a random channel. The button below places all speakers and conversations on random channels. IMPORTANT: Choosing a random placement deletes previous routing values. Each speaker or/and conversation will be placed on only *one* output channel.

Apart from the button to load the corpus, the *general settings* panel contains also two sliders: The *gain* knob allows the control of the main volume, the channel number selection sets the number of output channels. With some Hosts, after having adapted the number of output channels, the output channels have to be deactivated and activated again to render audio.

Appendix A.IV OSC commands for plugin

Address pattern	Argument(s)	Description
/cs/talk	-	Starts crowd noise generation
/cs/stop	-	Stops crowd noise generation
/cs/gain	float	Sets gain/volume to float in argument, values under 0 (or over 1) are set to 0 (or 1)
/cs/load	string	Loads grain data stored under path given in the string argument
/cs/setX	float	Sets the current x position
/cs/setY	float	Sets the current y position
/cs/setXSpread	float	Sets the current x spread
/cs/setYSpread	float	Sets the current y spread
/cs/conversation/add	int (optional)	If no arguments are passed, one conversation is added, if an integer value is passed, it sets the number of conversations added
/cs/conversation/delete	int	Deletes the conversation with the number of the int argument. If the conversation doesn't exist, nothing happens
/cs/conversation/addVoice	int, int	Adds voice with number of second integer argument to conversation with number of first integer argument
/cs/conversation/deleteVoice	int, int	Deletes voice with number of second integer argument from conversation with number of first integer argument
/cs/conversation/setChattiness	int, float	Sets the chattiness of the conversation with the number of the first integer argument to the float value of the second argument

Appendix B Statement Sound Designer Markus Rebholz

Statement of Markus Rebholz (Sound Designer) regarding the use of Wallas in today's sound design in Germany in a conversation via Facebook Messenger on December 12th, 2017:

> Aufgrund von Zeit- und Budgetaufwand werden in Deutschland eher Wallas aus Librarys verwendet als explizit aufgenommene. Selbstverständlich kommt es immer darauf an, ob das ausgesuchte "Stimmengewirr" zur Szene passt (Kinderanteil, Frauen- vs. Männeranteile, Stimmung, Stereo vs. Mono, etc.) und tontechnisch einwandfrei ist. Manchmal kann man sich mit Universal-Wallas behelfen, in denen die Sprecher eine Art Fantasiesprache sprechen, die keinesfalls herauszuhören ist. Es können auch Wallas zusätzlich rückwärts über das Original gelegt werden. Bei größeren, höher budgetierten Produktionen wird jedoch selbstverständlich ein passendes Walla für bestimmte Szenen aufgenommen. In Hollywood geht man im Gegensatz zu hiesigen Produktionen fast ausschließlich immer her und nimmt das benötigte Walla auf.
> (Markus Rebholz, 2017)

Appendix C Figures and Tables

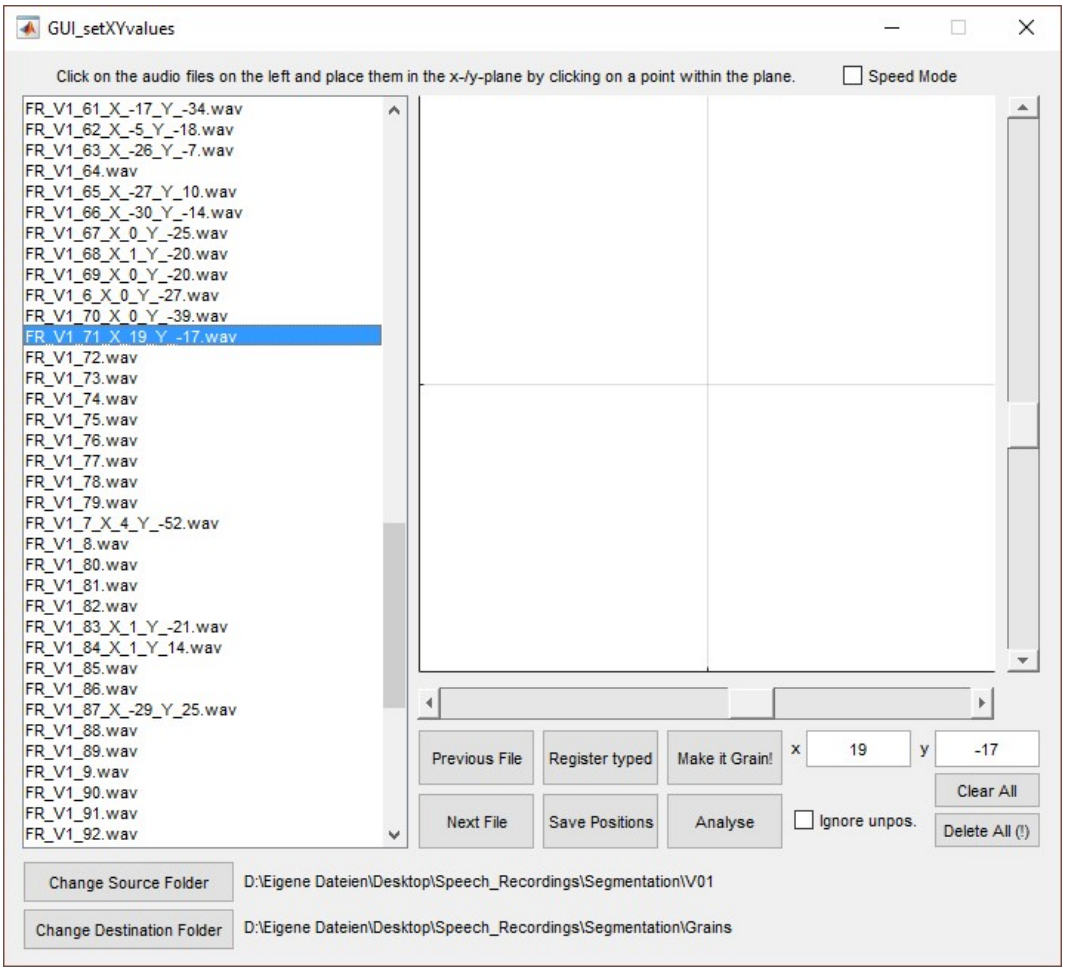


Figure C-1: The developed MATLAB GUI-application to facilitate manual mapping of audio files on a two-dimensional plane. Chosen positions are stored in the file name.

Table C-1: Results for linear correlation of utterances from movies and talk shows with valence and arousal dimension

Valence			Arousal		
Parameter	**Corr.**	**p-val.**	**Paramter**	**Corr.**	**p-val.**
HammarbergVocalEff.	−0.2106	< 0.0001	HammarbergVocalEff.	0.5061	< 0.0001
HammarbergUnstable	−0.1946	< 0.0001	PeakIntensity	0.5046	< 0.0001
HammarbergHead	−0.1869	< 0.0001	HammarbergUnstable	0.4993	< 0.0001
HammarbergCoarse	−0.1695	0.0002	MeanRMS	0.4377	< 0.0001
HammarbergBreathyV.	−0.1473	0.0011	RMSVariance	−0.4307	< 0.0001
MeanRelPitchDiff.	−0.1383	0.0028	PitchVariance	0.4267	< 0.0001
MaxRelPitchDiff.	−0.1359	0.0033	PitchStandard	0.4266	< 0.0001
PeakIntensity	−0.1280	0.0046	MaxPitch	0.4231	< 0.0001
VarRelPitchDiff.	−0.1273	0.0059	MeanAperiodicity	−0.4126	< 0.0001
StdRelPitchDiff.	−0.1246	0.0071	MaxRMS	0.4114	< 0.0001
MeanPitchStep	−0.1198	0.0084	SpectralRatio400	−0.3298	< 0.0001
ZeroCrossingsRate	0.1073	0.0179	MeanPitch	0.3077	< 0.0001
RMSVariance	0.0938	0.0382	PitchFallsAmount	0.3068	< 0.0001
SpectralRatio400	0.0917	0.0428	MeanPitchStep	0.2983	< 0.0001
PitchStandard	−0.0878	0.0525	MeanRelPitchDiff.	0.2949	< 0.0001

MaxProminence	−0.0838	0.0655	MaxProminence	0.2885	< 0.0001
MaxPitch	−0.0837	0.0647	PeakDistance	0.2794	< 0.0001
SpectralSpread	0.0828	0.0678	VoicedFraction	0.2772	< 0.0001
MeanPitch	−0.0823	0.0693	PitchRisesAmount	0.2751	< 0.0001
PitchVariance	−0.0807	0.0750	MaxRelPitchDiff.	0.2682	< 0.0001
SpectralRolloff	0.0805	0.0760	StdRelPitchDiff.	0.2501	< 0.0001
TonalPowerRatio	−0.0803	0.0766	SpectralRatio3000	0.2439	< 0.0001
PitchRisesAmount	−0.0796	0.0963	HammarbergHead	0.2343	< 0.0001
PitchFallsAmount	−0.0765	0.1055	VarRelPitchDiff.	0.2130	< 0.0001
PeakDistance	0.0682	0.1323	SpectralCentroid	−0.2092	< 0.0001
MaxRMS	−0.0599	0.1865	SpeechRate	0.1834	< 0.0001
SpectralFlatness	0.0555	0.2212	HammarbergCoarse	0.1763	0.0001
SpeechRate	−0.0551	0.2251	PitchRisesDuration	−0.1739	0.0003
PitchRisesDuration	0.0534	0.2651	SpectralSlope	0.1733	0.0001
MeanRMS	−0.0528	0.2442	SpectralFlatness	−0.1576	0.0005
MinRelRMSDiff.	0.0467	0.3143	PitchRisesFrequency	0.1508	0.0008
SpectralRatio3000	−0.0414	0.3617	MeanRelRMSDiff.	−0.1249	0.0070
MinRelPitchDiff.	−0.0318	0.4941	PitchFallsDuration	−0.1187	0.0118
SpectralCentroid	0.0270	0.5519	SpectralSpread	−0.1170	0.0097
VoicedFraction	−0.0249	0.5830	StdRelRMSDiff.	−0.1099	0.0176
RMSVoiced	0.0230	0.6126	HammarbergBreathyV.	0.1097	0.0153
PitchFallsFrequency	0.0227	0.6168	SpectralRolloff	−0.1029	0.0231
MeanAperiodicity	0.0204	0.6535	MaxRelRMSDiff.	−0.0973	0.0357
MeanRelRMSDiff.	−0.0105	0.8214	PitchFallsFrequency	0.0927	0.0407
VarRelRMSDiff.	−0.0104	0.8223	VarRelRMSDiff.	−0.0904	0.0510
MaxRelRMSDiff.	−0.0074	0.8731	MinPitch	−0.0877	0.0530
SpectralSlope	0.0051	0.9111	ZeroCrossingsRate	−0.0859	0.0582
PitchRisesFrequency	−0.0039	0.9323	TonalPowerRatio	−0.0772	0.0889
PitchFallsDuration	−0.0026	0.9555	MinRelRMSDiff.	−0.0607	0.1907
MinPitch	0.0026	0.9545	MinRelPitchDiff.	0.0579	0.2118
StdRelRMSDifferences	−0.0023	0.9607	RMSVoiced	0.0058	0.8982

Table C-2: Results of linear correlation for utterances from Grimaldi's (2017) recordings with valence and arousal dimension

Valence			**Arousal**		
Parameter	**Corr.**	**p-val.**	**Paramter**	**Corr.**	**p-val.**
MeanPitch	−0.3203	< 0.0001	MeanRMS	0.8983	< 0.0001
MeanRMS	−0.2906	< 0.0001	MaxRMS	0.8789	< 0.0001
RMSVariance	0.2881	< 0.0001	RMSVariance	−0.7937	< 0.0001
MeanAperiodicity	0.2833	< 0.0001	MeanPitch	0.7500	< 0.0001
PitchRisesAmount	−0.2730	< 0.0001	MaxPitch	0.6897	< 0.0001
MinPitch	−0.2691	< 0.0001	PitchFallsAmount	0.5834	< 0.0001
PitchFallsAmount	−0.2546	< 0.0001	MeanPitchStep	0.5428	< 0.0001
HammarbergVocalEff.	−0.2529	< 0.0001	PitchStandard	0.5348	< 0.0001
MaxRMS	−0.2419	< 0.0001	PitchRisesAmount	0.5253	< 0.0001
HammarbergUnstable	−0.2418	< 0.0001	PitchVariance	0.4806	< 0.0001
MeanPitchStep	−0.2376	< 0.0001	HammarbergVocalEff.	0.4608	< 0.0001
MaxPitch	−0.2244	< 0.0001	HammarbergHead	0.4582	< 0.0001
PeakIntensity	−0.2165	< 0.0001	SpectralSlope	0.4438	< 0.0001
HammarbergHead	−0.1983	< 0.0001	HammarbergUnstable	0.4383	< 0.0001
PitchRisesDuration	−0.1967	< 0.0001	PeakDistance	0.4359	< 0.0001
HammarbergCoarse	−0.1790	< 0.0001	HammarbergCoarse	0.4351	< 0.0001
PeakDistance	−0.1781	< 0.0001	MeanRelPitchDiffs	0.4264	< 0.0001
MaxProminence	−0.1752	< 0.0001	MaxProminence	0.4202	< 0.0001
MeanRelPitchDiffs	−0.1644	< 0.0001	HammarbergBreathyV.	0.3972	< 0.0001
VoicedFraction	−0.1608	< 0.0001	MaxRelPitchDiff.	0.3666	< 0.0001

SpectralSpread	−0.1546	< 0.0001	PeakIntensity	0.3662	< 0.0001
HammarbergBreathyV.	−0.1514	< 0.0001	MeanAperiodicity	−0.3595	< 0.0001
PitchFallsDuration	−0.1499	0.0001	StdRelPitchDiff.	0.3590	< 0.0001
SpectralSlope	−0.1439	0.0001	TonalPowerRatio	−0.3476	< 0.0001
SpectralRolloff	−0.1435	0.0001	SpectralRatio400	−0.2851	< 0.0001
ZeroCrossingsRate	−0.1424	0.0001	MinPitch	0.2703	< 0.0001
SpectralCentroid	−0.1277	0.0004	VarRelPitchDiff.	0.2188	< 0.0001
MaxRelPitchDiff.	−0.1036	0.0061	VoicedFraction	0.2146	< 0.0001
SpectralRatio3000	−0.0965	0.0076	MinRelPitchDiff.	0.2072	< 0.0001
PitchStandard	−0.0964	0.0080	SpectralFlatness	−0.2048	< 0.0001
SpectralRatio400	0.0908	0.0120	SpeechRate	0.1809	< 0.0001
RMSVoiced	−0.0903	0.0125	ZeroCrossingsRate	0.1702	< 0.0001
MinRelRMSDiff.	0.0777	0.0398	RMSVoiced	0.1489	< 0.0001
PitchRisesFrequency	−0.0690	0.0564	SpectralCentroid	−0.1415	0.0001
TonalPowerRatio	0.0664	0.0686	PitchRisesFrequency	0.0992	0.0060
PitchVariance	−0.0630	0.0833	PitchFallsFrequency	0.0952	0.0084
StdRelPitchDiff.	−0.0597	0.1143	SpectralSpread	−0.0833	0.0222
SpectralFlatness	−0.0572	0.1165	MinRelRMSDiff.	−0.0555	0.1423
VarRelPitchDiff.	0.0548	0.1473	SpectralRolloff	−0.0528	0.1481
MeanRelRMSDiffs	0.0431	0.2547	SpectralRatio3000	0.0290	0.4229
PitchFallsFrequency	−0.0422	0.2439	PitchFallsDuration	−0.0189	0.6156
MinRelPitchDiff.	−0.0408	0.2811	PitchRisesDuration	−0.0178	0.6370
StdRelRMSDiff.	0.0303	0.4225	VarRelRMSDiff.	0.0093	0.8062
MaxRelRMSDiffs	0.0226	0.5495	StdRelRMSDiff.	0.0092	0.8079
SpeechRate	0.0120	0.7422	MeanRelRMSDiffs	0.0033	0.9297
VarRelRMSDiff.	−0.0002	0.9967	MaxRelRMSDiff.	0.0004	0.9920

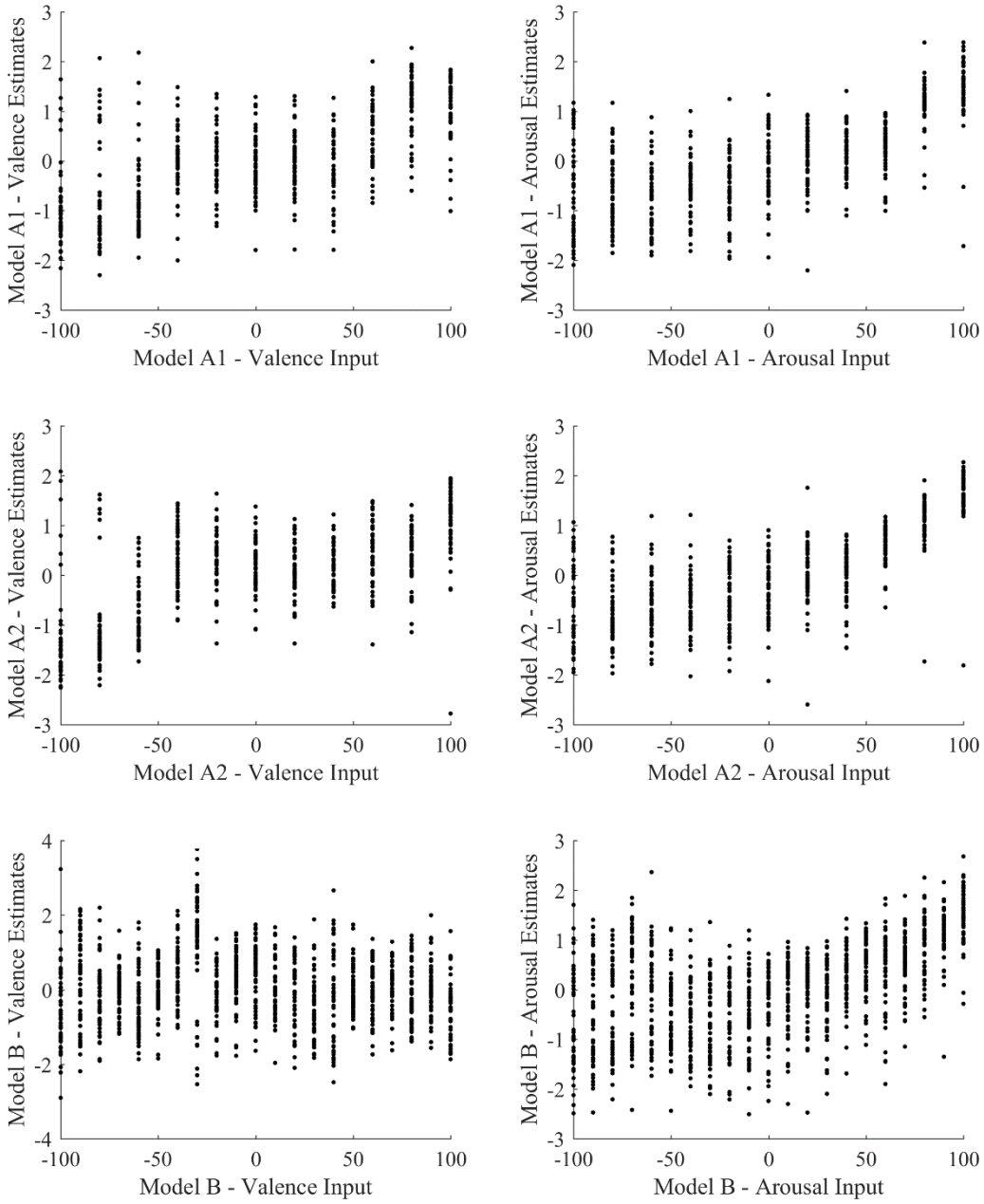


Figure C-2: Scatter plots of standardized, perceived valence and arousal values for the three models

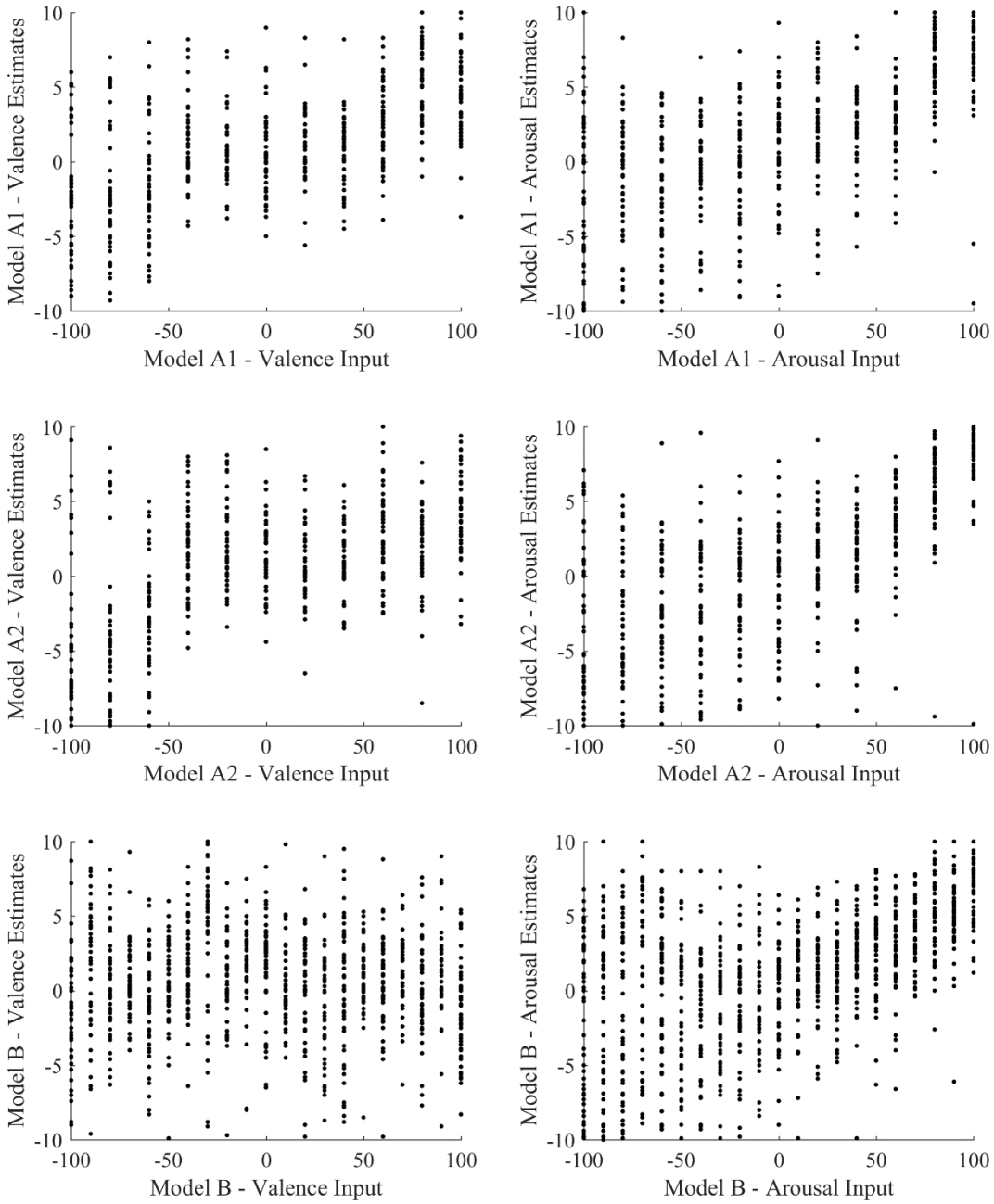


Figure C-3: Scatter plot of original results of the listening test for the three models and both affective dimensions

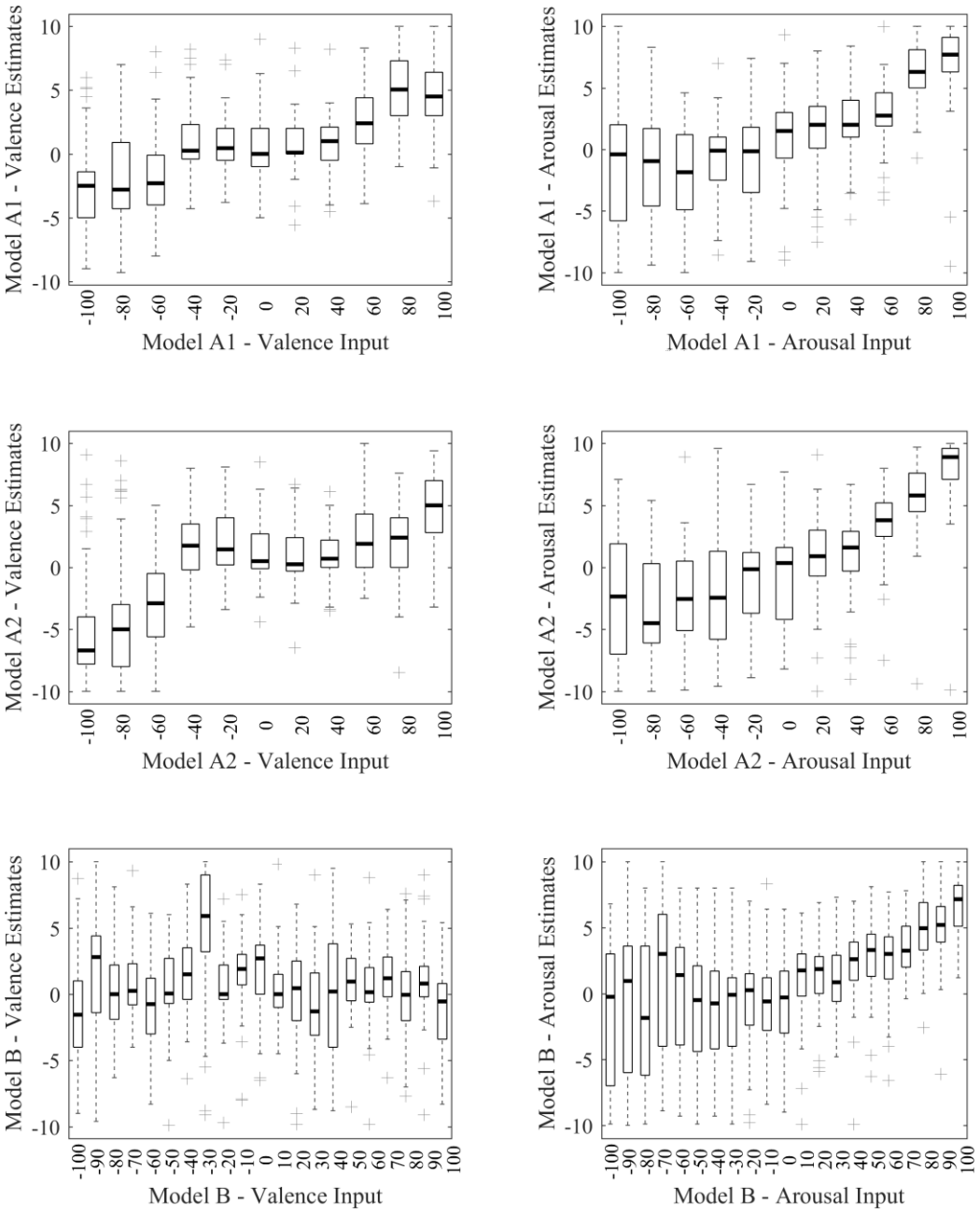


Figure C-4: Original results (without normalization) of perceived valence and arousal values

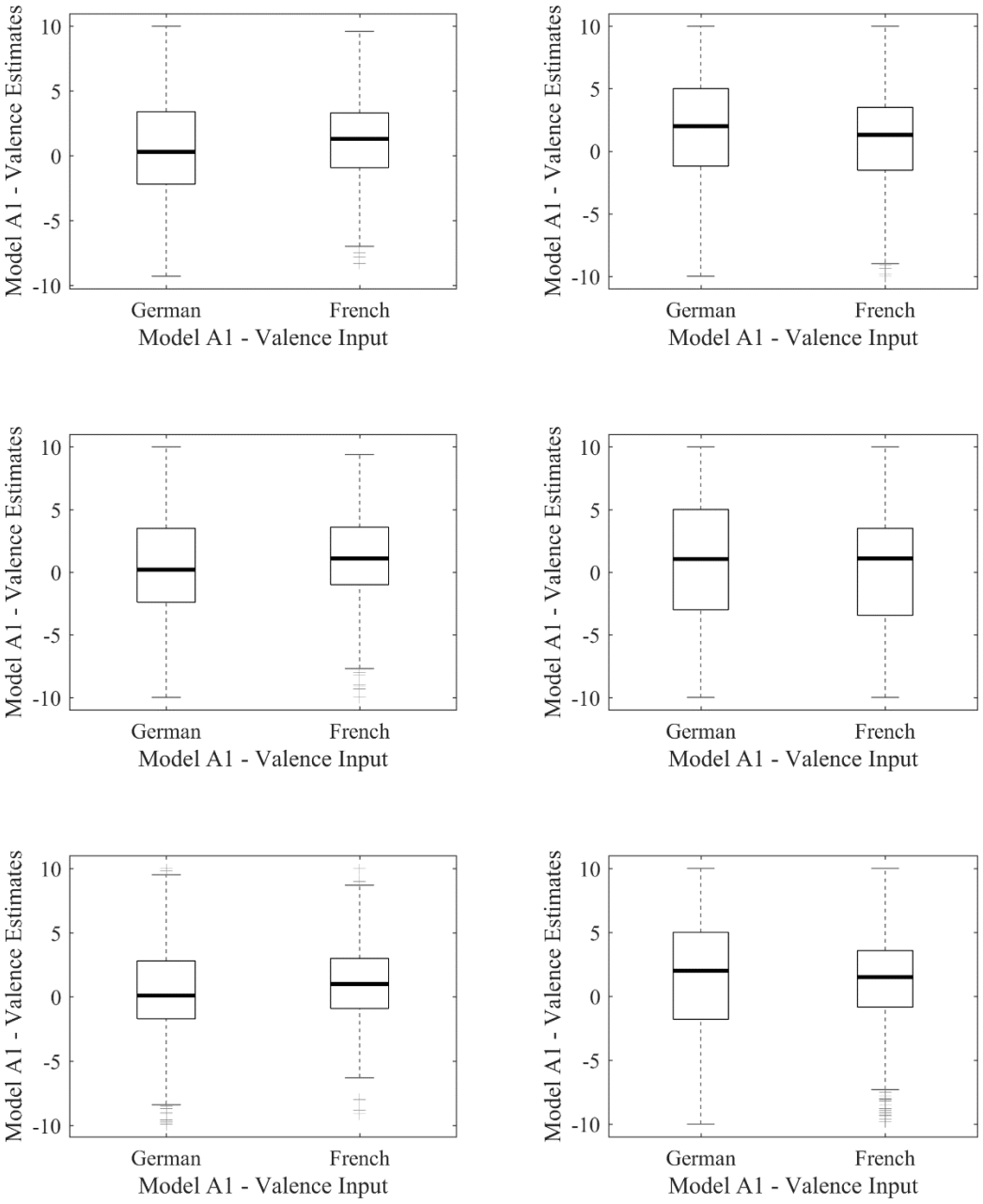


Figure C-5: Comparison of evaluations of participants who indicated German or French as native language

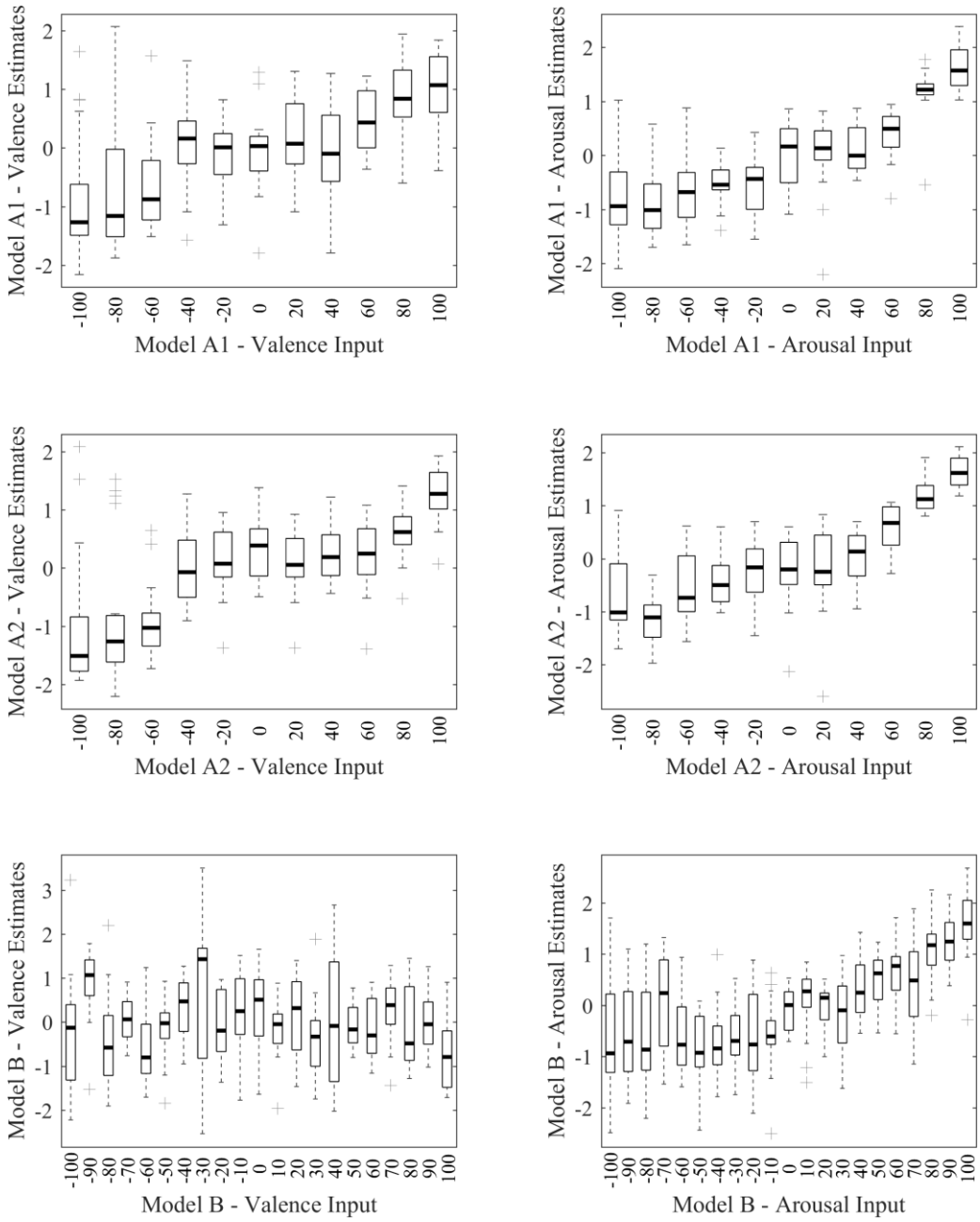


Figure C-6: Evaluations of subjects who indicated, that French was their native language

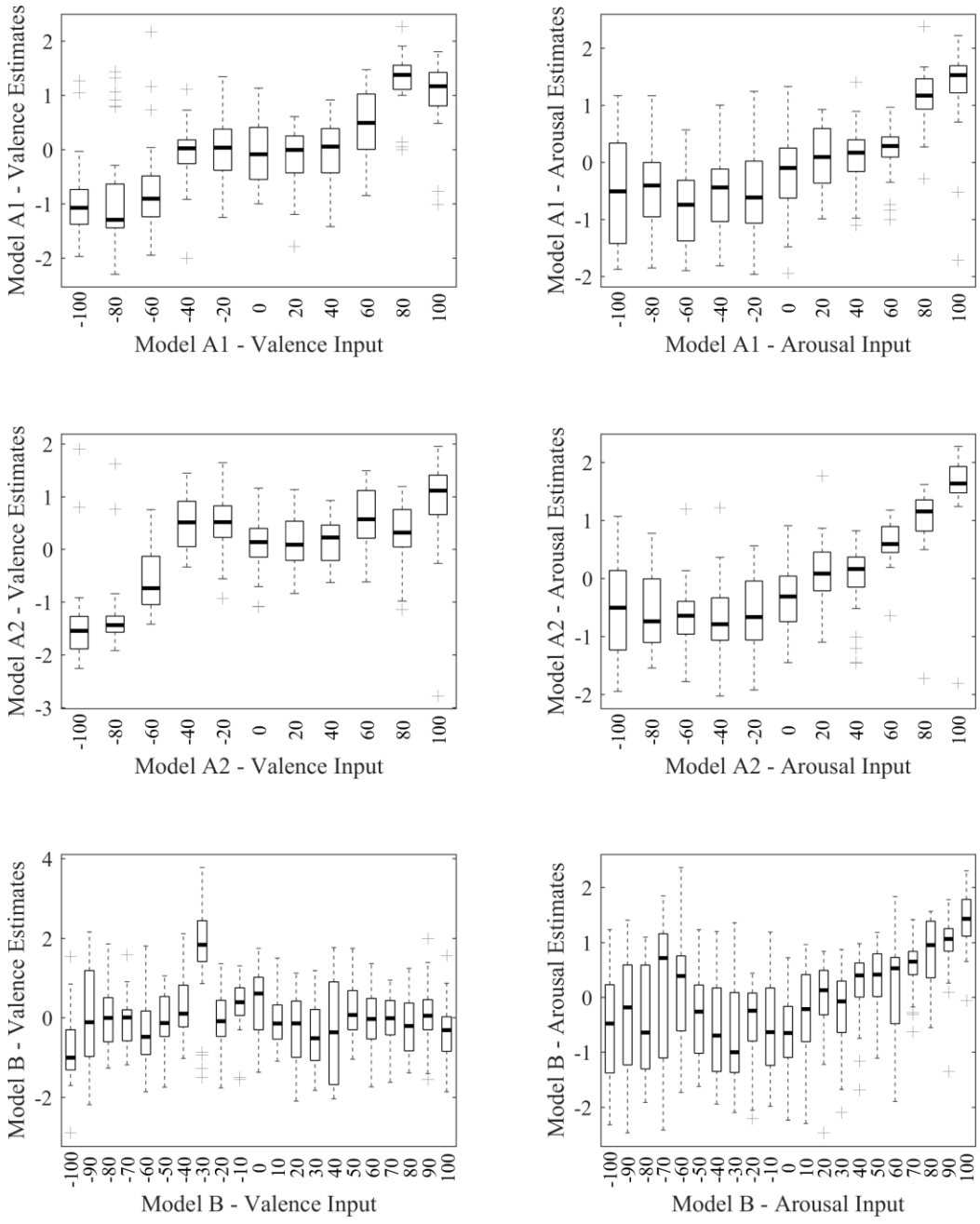


Figure C-7: Evaluations of subjects who indicated, that German was their native language

Appendix D Code snippets

Appendix D.I Function Process Block

```
void Cs_plugInAudioProcessor::processBlock(AudioSampleBuffer& buffer, MidiBuffer&
                        midiMessages)
{
  /// Function to fill output buffers. For every voice and speaker, respective syllable
  /// buffers are copied, RMS adapted if necessary and added to the output channels. This
  /// happens only on a midi or OSC request to play.

  // create some constants to make sure these values don't change during this runtime of
  // the function

   const int totalNumInputChannels = getTotalNumInputChannels();
   const int totalNumOutputChannels = getTotalNumOutputChannels();

   // clear buffer
   buffer.clear();

   // update some parameters
   checkForUpdatedParams(buffer);

   // if no audio grains are loaded, we can't play anything - so we return
   if (datastate == Loaded == 0) return;

   // MIDI PROCESSING =======

   handleMIDIEvents(midiMessages);

   // AUDIO PROCESSING =======

   // loop through every speaker

   for (int speakerNumber = 0; speakerNumber < speakers.size(); ++speakerNumber)
   {
        // get current speaker as constant to allow channel changes during execution
        SingleSpeaker curr_speaker = speakers[speakerNumber];
        processSpeaker(buffer, curr_speaker);
        speakers[speakerNumber] = curr_speaker;
   }

   // loop through conversations

   for (int c = 0; c < conversations.size(); ++c)
   {
      if (conversations[c].activeVoice >= 0)
      {
         NewConversation conv_copy = conversations[c];
         processConversation(buffer, conv_copy);
         conversations[c] = conv_copy;
      }
   }

   // apply current Gain (with ramp if necessary)

   const float currentGain = *parameters.getRawParameterValue("gainParam");

   if (currentGain == previousGain)
   {
      buffer.applyGain(currentGain);
   }
   else
   {
      buffer.applyGainRamp(0, buffer.getNumSamples(), previousGain, currentGain);
      previousGain = currentGain;
   }
}
```

Appendix D.II Function Process Stream

```
template<typename T> void Cs_plugInAudioProcessor::processStream(AudioSampleBuffer&
buffer, T & curr_speaker)
{

  const int totalNumOutputChannels = getTotalNumOutputChannels();

  int outputSamplesRemaining = buffer.getNumSamples();
  int outputSamplesOffset = 0;

  // loop through all output samples to fill
while (outputSamplesRemaining > 0)
{

  // check speaker's state
  switch (curr_speaker.speakerState)
  {

    // speaker is saying a syllable
    case StreamState::PlaySyll:
    {

     // determine how many samples are copied this time
        int bufferSamplesRemaining = syllables[curr_speaker.getVoice()]
                        [curr_speaker.newSyllable].audio[curr_speaker.getPitch()].
                        getNumSamples() - curr_speaker.position - crossfadeLength;

        int samplesThisTime = jmin(outputSamplesRemaining, bufferSamplesRemaining);

     // copy audio data from syllable matrix to copybuffer

        copybuffer1.copyFrom(0,
                        0,
                        syllables[curr_speaker.getVoice()]
                        [curr_speaker.newSyllable].audio[curr_speaker.getPitch()],
                        0,
                        curr_speaker.position,
                        samplesThisTime);

     // scale audio to currently in Syllable object saved RMS factor (which was determined
     // in previous run to avoid audible RMS jumps)

        if (curr_speaker.old_RMS_factor != 1.f)
        {
          copybuffer1.applyGain(curr_speaker.old_RMS_factor);
        }

     // get the output channels of the current speaker

        Array<int> channels = curr_speaker.channels;

     // loop through every channel of current speaker

        for (int channelAddress = 0; channelAddress < channels.size(); ++channelAddress)
        {
           if (channels[channelAddress] < totalNumOutputChannels)
           {
              // copy audio data from copybuffer to channel of output buffer

                 buffer.addFrom(channels[channelAddress],
                        outputSamplesOffset,
                        copybuffer1,
                        0,
                        0,
                        samplesThisTime,
                        curr_speaker.getLevel(channels[channelAddress]));
           }
        }

     // subtract copied samples from remaining sample number and update speaker's
     // position within syllable
```

```
        outputSamplesRemaining -= samplesThisTime;
        outputSamplesOffset += samplesThisTime;
        curr_speaker.position += samplesThisTime;

     // if speaker has rached end of syllable, prepare everything for the next syllable or
     // pause

        if (curr_speaker.position == syllables[curr_speaker.getVoice()]
                        [curr_speaker.newSyllable].audio[curr_speaker.getPitch()].
                        getNumSamples() - crossfadeLength)
        {

        curr_speaker.prepareNewState(StreamState::PlaySyll, crossfadeLength, fs);

        // if speaker speaks another syllable after the one that has just been copied, the
        // RMS of the new syllable has to be verified and maybe adapted

        if (curr_speaker.speakerState == StreamState::Crossfade)
        {
           // compare rms values, adapt the new grain's rms to the previous one to avoid
           // jumps

           float rms_fraction = syllables[curr_speaker.getVoice()]
                        [curr_speaker.newSyllable].rms / average_rms;

           if (rms_fraction > RMSFACTOR)
           {
              curr_speaker.new_RMS_factor = RMSFACTOR / rms_fraction;
           }
           else if (rms_fraction < 2 - RMSFACTOR)
           {
              curr_speaker.new_RMS_factor = (2 - RMSFACTOR) / rms_fraction;
           }
           else
           {
              curr_speaker.new_RMS_factor = 1.f;
           }
        }
     }

     break;
}
// speaker is changing from one to another syllable
case StreamState::Crossfade:
{
   int bufferSamplesRemaining = crossfadeLength - curr_speaker.fadePosition;
   int samplesThisTime = jmin(outputSamplesRemaining, bufferSamplesRemaining);

   // copy data from syllable matrix to temporate buffers for old and new syllable

   copybuffer1.copyFrom(0,
                        0,
                        syllables[curr_speaker.getVoice()][curr_speaker.oldSyllable].audio
                        [curr_speaker.getPitch()],
                        0,
                        curr_speaker.position + curr_speaker.fadePosition,
                        samplesThisTime);

   copybuffer2.copyFrom(0,
                        0,
                        syllables[curr_speaker.getVoice()][curr_speaker.newSyllable].audio
                        [curr_speaker.getPitch()],
                        0,
                        curr_speaker.fadePosition,
                        samplesThisTime);

   // create fades

   copybuffer1.applyGainRamp(0,
                        samplesThisTime,
                        (1 - curr_speaker.fadePosition / crossfadeLength) *
                        curr_speaker.old_RMS_factor,
```

```
                        (1 - (curr_speaker.fadePosition + samplesThisTime) /
                        crossfadeLength) * curr_speaker.old_RMS_factor);

   copybuffer2.applyGainRamp(0,
                        samplesThisTime,
                        curr_speaker.fadePosition / crossfadeLength *
                        curr_speaker.new_RMS_factor,
                        (curr_speaker.fadePosition + samplesThisTime) / crossfadeLength *
                        curr_speaker.new_RMS_factor);

    // loop through every channel

    for (int channelAddress = 0; channelAddress < curr_speaker.channels.size();
                        ++channelAddress)
    {
        int channel = curr_speaker.channels[channelAddress];

        if (channel < totalNumOutputChannels)
        {
           // copy data from copybuffers to output channel

           buffer.addFrom(channel,
                          outputSamplesOffset,
                          copybuffer2,
                          0,
                          0,
                          samplesThisTime,
                          curr_speaker.getLevel(channel));

           buffer.addFrom(channel,
                          outputSamplesOffset,
                          copybuffer1,
                          0,
                          0,
                          samplesThisTime,
                          curr_speaker.getLevel(channel));
         }
      }

   outputSamplesRemaining -= samplesThisTime;
   outputSamplesOffset += samplesThisTime;
   curr_speaker.fadePosition += samplesThisTime;

   // if speaker has reached the end of the crossfade, the new state (PlaySyll) has to be
   // prepared

   if (curr_speaker.fadePosition == crossfadeLength)
   {
      curr_speaker.prepareNewState(StreamState::Crossfade, crossfadeLength, fs);
   }

break;

}
// speaker will have a pause next and fades out current syllable
case StreamState::Fadeout:
{
   int bufferSamplesRemaining = crossfadeLength - curr_speaker.fadePosition;
   int samplesThisTime = jmin(outputSamplesRemaining, bufferSamplesRemaining);

   // copy audio data from matrix to temporate buffer and apply fade-out

   copybuffer1.copyFrom(0,
                        0,
                        syllables[curr_speaker.getVoice()][curr_speaker.oldSyllable].audio
                        [curr_speaker.getPitch()],
                        0,
                        curr_speaker.position + curr_speaker.fadePosition,
                        samplesThisTime);

   copybuffer1.applyGainRamp(0,
                        samplesThisTime,
```

```
                        (1 - curr_speaker.fadePosition / crossfadeLength) *
                        curr_speaker.old_RMS_factor,
                        (1 - (curr_speaker.fadePosition + samplesThisTime) /
                        crossfadeLength) * curr_speaker.old_RMS_factor);

   // loop through every channel

   for (int channelAddress = 0; channelAddress < curr_speaker.channels.size();
                        ++channelAddress)
   {
      int channel = curr_speaker.channels[channelAddress];

      if (channel < totalNumOutputChannels)
      {
          // copy data from copybuffer to output channel

          buffer.addFrom(channel,
                        outputSamplesOffset,
                        copybuffer1,
                        0,
                        0,
                        samplesThisTime,
                        curr_speaker.getLevel(channel));
      }
   }

   outputSamplesRemaining -= samplesThisTime;
   outputSamplesOffset += samplesThisTime;
   curr_speaker.fadePosition += samplesThisTime;

   // if speaker reaches the end of the fade out, the pause has to be prepared

   if (curr_speaker.fadePosition == crossfadeLength)
   {
      curr_speaker.prepareNewState(StreamState::Fadeout, crossfadeLength, fs);
   }

   break;

}
// speaker is pausing
case StreamState::Pausing:
{
   int pauseSamplesRemaining = curr_speaker.pauseLength - curr_speaker.position;
   int samplesThisTime = jmin(outputSamplesRemaining, pauseSamplesRemaining);

   // no data is copied, position within pause is updated

   outputSamplesRemaining -= samplesThisTime;
   outputSamplesOffset += samplesThisTime;
   curr_speaker.position += samplesThisTime;

   // if speaker reaches end of pause, the new state (PlaySyll) has to be prepared

   if (curr_speaker.position == curr_speaker.pauseLength)
   {
      curr_speaker.prepareNewState(StreamState::Pausing, crossfadeLength, fs);
   }

   break;

}
// speaker is not talking due to missing available grains
case StreamState::NoGrainsInBuffer:
{
   outputSamplesRemaining = 0;
   outputSamplesOffset = 0;

   // check if speaker buffer contains syllable numbers; if yes, prepare new state

   if (curr_speaker.isReadyToPlay())
   {
      curr_speaker.prepareNewState(StreamState::Pausing, crossfadeLength, fs);
```

```
   }

   break;

}
// speaker is not active, waiting to be activated by MIDI or OSC signal
case StreamState::Inactive:
{
   outputSamplesRemaining = 0;
   outputSamplesOffset = 0;

   // check if speaker has been activated

        if (curr_speaker.shallSpeak)
        {
           curr_speaker.prepareNewState(StreamState::Pausing, crossfadeLength, fs);
        }
      }
    }
  }
}
```

Appendix D.III Function Prepare New State

```
void prepareNewState(StreamState oldState, int crossFadeLength, double fs)
{
   switch (oldState)
   {
   case StreamState::PlaySyll:

      oldSyllable = newSyllable;
      fadePosition = 0;
      --currUtteranceLength;

      if (!grainbuffers[activeVoice].isEmpty() && shallSpeak && currUtteranceLength > 0)
      {
         newSyllable = grainbuffers[activeVoice].pop();
         speakerState = StreamState::Crossfade;
      }

      if (grainbuffers[activeVoice].isEmpty() || newSyllable == -1 || !shallSpeak ||
                                                                  currUtteranceLength < 1)
      {
         speakerState = StreamState::Fadeout;
         generatePause(fs);
      }

   break;

   case StreamState::Crossfade:

      position = crossFadeLength;
      speakerState = StreamState::PlaySyll;
      old_RMS_factor = new_RMS_factor;

   break;

   case StreamState::Fadeout:

      position = 0;
      speakerState = StreamState::Pausing;
      resetRMS();

   break;

   case StreamState::Pausing:

      if (currUtteranceLength < 1)
      {
         changeVoiceLottery();
      }

      position = 0;
      oldSyllable = newSyllable;
```

```
      if (!grainbuffers[activeVoice].isEmpty() && shallSpeak)
      {

         newSyllable = grainbuffers[activeVoice].pop();

            if (newSyllable < 0)
            {
               speakerState = StreamState::Pausing;
               generatePause(fs);
            }
            else
            {
               speakerState = StreamState::PlaySyll;
            }
      }
      else if (!grainbuffers[activeVoice].isEmpty() && !shallSpeak)
      {
         speakerState = StreamState::Inactive;
      }
        else
        {
           speakerState = StreamState::NoGrainsInBuffer;
        }

   break;

   case StreamState::NoGrainsInBuffer:

      position = 0;
   }
}
```

Appendix D.IV Function Fill Grain Buffers

```
Cs_plugInAudioProcessor::fillNextGrainBuffers(NewConversation & conv)
{

   const int voicenum = conv.getVoice();
   GrainBuffer & grainbuffer = conv.getBuffer();

   if (selectedGrains[voicenum].size() > 4)
   {
      Array<int> temp_grainVector = selectedGrains[voicenum];

      while (!grainbuffer.isFull() && temp_grainVector.size() > 4)
      {
         int halfIndex = (int)temp_grainVector.size() / 2;
         int curr_RMS_difference = 0;

         if (grainbuffer.timeUnstressed < minUnstressedTime)
         {
            // unstressed syllable

            int nextIndice = random.nextInt(halfIndex);
            int nextSyllable = temp_grainVector[nextIndice];

            if (grainbuffer.tryPush(nextSyllable))
            {
               temp_grainVector.removeAndReturn(nextIndice);
               grainbuffer.timeUnstressed += syllables[voicenum][nextSyllable]
                                          .audio[conv.getPitch()].getNumSamples();
               grainbuffer.lastGrainLength = syllables[voicenum][nextSyllable]
                                          .audio[conv.getPitch()].getNumSamples();
               grainbuffer.lastGrainStressed = false;
               ++grainbuffer.grains_since_last_pause;
               grainbuffer.firstGrainRMS = grainbuffer.secondGrainRMS;
               grainbuffer.secondGrainRMS = syllables[voicenum][nextSyllable].rms;
               curr_RMS_difference = grainbuffer.firstGrainRMS –
                                          grainbuffer.secondGrainRMS;
            }
         }
         else
         {
```

```
            // stressed syllable
            int nextIndice = halfIndex + random.nextInt(halfIndex - 1);
            int nextSyllable = temp_grainVector[nextIndice];

            if (grainbuffer.tryPush(nextSyllable))
            {
               temp_grainVector.removeAndReturn(nextIndice);
               grainbuffer.timeUnstressed = 0;
               grainbuffer.lastGrainLength = syllables[voicenum][nextSyllable]
                                    .audio[conv.getPitch()].getNumSamples();
               grainbuffer.lastGrainStressed = true;
               ++grainbuffer.grains_since_last_pause;
               grainbuffer.firstGrainRMS = grainbuffer.secondGrainRMS;
               grainbuffer.secondGrainRMS = syllables[voicenum][nextSyllable].rms;
               curr_RMS_difference = grainbuffer.firstGrainRMS –
                                    grainbuffer.secondGrainRMS;
             }
          }


         if (grainbuffer.lastGrainStressed == false && grainbuffer.grains_since_last_pause
                > grainbuffer.min_grains_before_pause && curr_RMS_difference > 0 &&
                (float)grainbuffer.lastGrainLength / fs > LASTGRAINLENGTH)
         {

            // pause lottery

            float pauselottery = random.nextFloat();

            if (pauselottery > grainbuffer.notPauseProbability)
            {
               if (grainbuffer.tryPush(-1))
               {
                  grainbuffer.notPauseProbability = default_notPausePropability;
                  grainbuffer.grains_since_last_pause = 0;
                  grainbuffer.lastGrainLength = 0;
               }
            }
            else
            {
               grainbuffer.notPauseProbability *= default_notPausePropability;
            }
         }

      // refill vector with possible numbers when only x grains are left
      if (temp_grainVector.size() < 5)
      {
         temp_grainVector = selectedGrains[voicenum];
      }
    }
  }
}
```