Extraction of articulatory parameters from audio files used for sound sketching

Master's thesis
January 1, 2016

Student: Martin Hellwagner, 357516
Supervisor: Prof. Stefan Weinzierl (TU Berlin)
Advisor: Prof. Anders Friberg (KTH Stockholm)

Statutory Declaration
Eidesstaatliche Erklärung

I hereby declare that I have created this work autonomously and without illicit external help as well as having solely used the sources and tools listed.

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig sowie ohne unerlaubte fremde Hilfe und ausschließlich unter Verwendung der aufgeführten Quellen und Hilfsmittel angefertigt habe.

Date / Signature
Datum / Unterschrift

Acknowledgements

First and foremost, I would like to thank Prof. Anders Friberg for introducing me to this interesting project and helping me choose the topic of my work as well as supporting me during my months of research in Stockholm. The same acknowledgements are due to Prof. Stefan Weinzierl for the many hours he spent reading and correcting my thesis as well as giving me invaluable advice on it.

Furthermore, I'd like to thank my friends - in particular Martha, Matthias, Max and Niels - for making sure I retain my sanity during the process of crafting this work. Last but not least, I want to express my endless gratitude to my parents for their constant support during my studies, be it with generous financial help as well as with the recurrent stern reminder that I ought to finish what I started.

Abstract. Any design activity starts from sketching. In the visual domain, sketches effectively communicate verbal and non-verbal expressions as writings and drawings, whereas in the sonic domain, they are conveyed as speech and imitations. In order to create novel interaction paradigms for sound designers, it is crucial to understand and exploit the potential of vocal imitations. This Master's thesis contributes to the academic research by investigating the automatic extraction of articulatory parameters from audio files of everyday sounds vocally imitated by actors. In particular, three such parameters, namely *myoelastic*, *phonation* and *turbulent* are introduced. Thus, the overall goal of the work at hand is to create a classification system which is able to automatically determine whether or not an audio file exhibits aforementioned parameters. At first, an overview of the subject matter is given. Subsequently, the program is explained and its results presented. Eventually, problems are discussed and potential improvements are put forward.

Zusammenfassung Jede Design-Aktivität beginnt mit einer Skizze. In der visuellen Domäne können Skizzen erfolgreich verbale und non-verbale Ausdrücke wie Schriftstücke und Zeichnungen vermitteln. In der sonischen Domäne werden diese wiederum als Sprache und Imitationen ausgedrückt. Um neue Interaktions-Paradigmen für Sound-Designer zu schaffen, ist das Wissen und die Ausschöpfung des Potenzials stimmlicher Imitationen äußerst wichtig. Diese Masterarbeit trägt durch die Untersuchung der automatischen Extraktion bestimmter Artikulations-Parameter aus Audiodateien, welche von Schauspielern imitierte Alltagsgeräusche enthalten, zur akademischen Forschung bei. Drei solcher Parameter, nämlich *myoelastic*, *phonation* und *turbulent*, werden vorgestellt. Die allgemeine Zielsetzung der vorliegenden Arbeit ist hierbei die Erschaffung eines Klassifikations-Systems, welches automatisch erkennt, ob eine Audiodatei obige Parameter enthält oder nicht. Zuerst wird ein Überblick über die Thematik gegeben. Anschließend wird das Programm genauer erläutert und dessen Resultate präsentiert. Schlussendlich werden Probleme diskutiert und eventuelle Verbesserungen vorgestellt.

Table of Contents

List of Abbreviations

API Application Programming Interface
CUIDADO Content-based Unified Interfaces and Descriptors for Audio/Music Databases available Online
IRCAM Institut de Recherche et Coordination Acoustique/Musique
KTH Kungliga Tekniska högskolan
PCA Principal Component Analysis
PCR Principal Component Regression
PLS Partial Least Squares
RMS Root Mean Square
SkAT-VG Sketching Audio Technologies using Vocalization and Gestures
SOM Self-Organising Maps
SVM Support Vector Machine

List of Figures

List of Tables

1 Introduction

Any design activity starts from sketching. By means of sketches the designer produces, verifies, selects, communicates and refines ideas [14]. Much like visual sketching yields a draft of the concept to be devised (e.g. the drawing of an elephant), sound sketching too strives to create an approximation of the desired sound (e.g. the imitation of an elephant's trumpeting). While the former relies heavily on the hand and the pencil, the latter is ideally represented by the acoustic equivalent of its visual counterpart: the human voice. Both "tools" allow their users to produce verbal and non-verbal expressions in a natural way [19]. In the visual domain, these expressions are communicated as writings and drawings, whereas in the sonic domain, they are conveyed as speech and imitations.

It might therefore not be surprising that academic research is currently trying to investigate and exploit the potential of vocal imitations. The EU project SkAT-VG[1] (Sketching Audio Technologies using Vocalization and Gestures, 2014-2016), which this Master's thesis is part of, aims at finding ways to exploit voice and gestures in sonic interaction design [12, 18, 20]. Its goal is threefold: (i) improve the understanding of how sounds are communicated through vocalizations and gestures, (ii) look for physical relations between vocal sounds and sound-producing phenomena, (iii) design tools for converting vocalizations and gestures into parametrised sound models [14]. In order to attend to such an extensive workload, three universities (KTH[2] in Stockholm, IRCAM[3] in Paris as well as IUAV[4] in Venice) and one company (Genesis Acoustics[5] in Aix-en-Provence) cooperate with each other.

In particular, the project's aim is to bridge the gap between utterances and sound models so that designers can start using the rich domain that is their voice as naturally and fluidly as when drawing with aforementioned pencil [18]. If so, sonic interaction ideas can thus be quickly and painlessly produced, compared and evaluated, ultimately leading to a smoother design process. Moreover, sonic sketches do not require particular skill and are both readily available and highly performative, irrespective of locality and situation [14].

In fact, the recognition of the human voice's remarkable applicability for sound sketching might even enable novel human-machine interactions, allowing users to communicate with software in a more natural way [3]. If a machine understood vocal imitations, users could, for example, set parameters in music production applications, program a synthesiser or even look for sounds in audio search applications using solely their voice. Other areas of operation include films and multimedia shows (sound effects), games (sound-mediated sense of agency), everyday products, environments and vehicles [20].

[1] http://skatvg.iuav.it
[2] http://www.kth.se/en
[3] http://www.ircam.fr/?&L=1
[4] http://www.iuav.it/English-Ve/About-Iuav/Iuav-profi/index.htm
[5] http://genesis-acoustics.com/en

Since the thesis was created while researching at KTH, the student's efforts were focused on the university's assigned goal of improving the understanding of the communication of sounds through vocalizations. Specifically, the work at hand will investigate the automatic extraction of articulatory parameters from audio files of everyday sounds (e.g. animal sounds, mechanical sounds) vocally imitated by actors.

In order to collect the necessary data, the recording of four actors (equally distributed by gender) has been undertaken at KTH before the arrival of the student. Each subject was asked to imitate a number of sounds that were played to them. The resulting sound files were then annotated by the department's phonetic experts, following the array of parameters introduced by Helgason [7]: *myoelastic*, *phonation* and *turbulent*. Applying these annotations on the raw data, a large number of short audio files (i.e. fragments) are eventually extracted using an algorithm developed by the student. These fragments are either marked as positive or negative, depending on their possession of the respective parameter or the lack thereof.

Subsequently, a set of features is created using different criteria and statistical measurements of the fragments in question. At the core of the program lies the auto-correlation method introduced by Cheveigné and Kawahara [2]. Note that all the features are calculated using only the time domain, thus hypothesising that good performance can be obtained without examining the frequency domain. In fact, the final results of the classification only slightly fall behind the performance of the classification operating on the same files' spectrograms, which is simultaneously developed by Friberg and his team at KTH [5]. This points out a promising alternative approach to some kinds of speech processing, in which the use of the Fourier transformation is rendered unnecessary [24].

That being said, the overall goal of the program lies in the correct classification of as many fragments as possible according to their parametric characteristics. The ground truth is hereby provided by aforementioned human annotation of the fragments. Two types of prediction models are used to forecast the results of the classification: Partial Least Squares (PLS) regression [6, 22] and Support Vector Machine (SVM). Both models use the built-in functions of MATLAB[6], the programming environment which the code is written in. To ensure flexibility and applicability in real-world scenarios, N-fold and leave-one-out cross-validation are additionally performed on the data sets.

The remainder of the Master's thesis is organised as follows: Section 2 gives an overview of the subject matter's background, dealing with both the topic of sound sketching in general and the specific academic work carried out as part of the SkAT-VG project in particular. Section 3 introduces the methodology of the project, accompanied by different parts of the source code. Subsequently, Section 4 presents the results revealing good performance of the algorithm in question, and comments on different parameter combinations thereof. Eventually in Section 5, problems are discussed and potential improvements are put forward. Section 6 contains concluding remarks on the topic.

[6] http://www.mathworks.com/products/matlab

2 Background

Whereas a fair amount of papers are available in the research community dealing with sound analysis (feature extraction), machine learning (classification) and sound synthesis (parameter mapping and control) in general, the topic of vocal imitations being used for sonic interactions has only recently been given attention. Nevertheless, some notable publications in the field are still worth mentioning. In order to structure the subject matter's background, four subsections have been chosen:

1. **Vocal imitations and verbal descriptions** as well as their applicability as sketching tools for sounds
2. **Classification frameworks** for vocal imitations and verbal descriptions
3. **Sound analysis and feature extraction**, in particular concepts used in the program at hand
4. **Sound sketching parameters** and their pivotal role in this thesis

Each subsection will introduce its respective concepts by presenting and drawing on a number of publications. Cross-references to other subsections will be made when necessary.

2.1 Vocal imitations and verbal descriptions

Although both vocal imitations and verbal descriptions are ways to mimic sounds, their paradigms are rather different. On the one hand, verbal descriptions use words to depict a sound, which are usually associated with a certain language. For example, the word commonly used to paraphrase a dog's bark in the English language is "woof", whereas in German the same sound might be denoted as "wuff" or "wau". This process of defining sounds with language is called *onomatopeoia*, stemming from the Greek words for "name" and "I make"[7]. Naturally, verbal descriptions are linked to a person's spoken languages and generally work best in their native tongue [11]. On the other hand, vocal imitations are identified by the actual reproduction of sounds (like a dog's bark), which are not associated with any language in particular. In fact, vocal imitations are constrained solely by the vocal ability of the speaker due to their lack of symbolic conventions. Since the human voice is a surprisingly versatile instrument, vocalizations are well-suited for mimicking all kinds of non-linguistic sounds.

Delle Monache et al. [14] even argue that, in order to sketch sonic interactions, vocal imitations could play a role as pivotal to the sound designer as the pencil does to its visual counterpart. In order to gain initial insight into the vast potential of vocal imitations, the researchers describe in their paper a workshop aimed at investigating the effectiveness of these sketches [14]. Twenty-four students of sound and music technologies are thus gathered to become familiar with the basics of sound sketching as well as be engaged in

[7] http://en.wikipedia.org/wiki/Onomatopoeia

a corresponding design session. In particular, two vocal techniques are introduced during the workshop [14]:

1. **Inhaled fry technique**: used to produce friction sounds
2. **Palate grind technique**: used to produce grinding and scraping sounds

Interestingly enough, the participants not only employ vocalizations but also gestures and movements when imitating the referent sounds (i.e. the sound-producing events), especially when familiarity with the latter is low. In general, dense and irregular temporal patterns are rather difficult to reproduce for the students, whereas direct and clearly defined actions yield better imitations [14]. However, limitations of their own voices are also noticed by the participants, in particular when sounds do not frequently occur in their everyday lives.

In order to further address the question of how effectively vocal imitations and verbal descriptions are able to communicate referent sounds to a listener, Lemaitre and Rocchesso [11] conduct and explain in their paper an adequate experiment. The researchers hypothesise that correct recognition of sounds would be better for vocal imitations than for verbal descriptions, and that the former would be affected by the identifiability of the sounds to a lesser extent than the latter. To start with, a total of 58 sounds are recorded in the following four predefined categories:

1. **Identifiable complex events**: very easy to recognise (e.g. coins dropped in a jar)
2. **Elementary mechanical interactions**: possible to recognise (e.g. tapping on a surface)
3. **Artificial sound effects**: difficult but not impossible to recognise (e.g. FM synthesis signal)
4. **Unidentifiable mechanical sounds**: nearly impossible to recognise (e.g. rubbing a pen against an umbrella)

Next, participants are asked to indicate how confident they feel about imitating each sound. A confidence score is derived from this classification, which is in itself negatively correlated with the causal uncertainty of the corresponding sound [11]. Nine sounds are subsequently selected from each category, with regard to maintaining both a large range of and a smooth distribution over the confidence score values. The resulting 36 sounds are then played to the participants, all of which have no expertise or prior training in either audio engineering, acoustics or auditory perception. Afterwards, participants are asked to record descriptions of each sound by both vocally and verbally imitating what they are played, bearing in mind the necessity of communicating the referent sounds to another person. In total, 576 sound files containing vocal imitations and verbal descriptions are thus recorded.

Thereupon, participants are asked to judge descriptors in terms of their adequacy to describe referent sounds. Not surprisingly, perhaps, vocalizations and verbalisations are judged less adequate for less identifiable sounds. Overall, the adequacy does not vary

between the two descriptors. In detail, however, verbal descriptions are more effective when the referent sounds are identifiable, whereas the effectiveness of vocal imitations is not affected by the type of sound they refer to [11].

Finally, Lemaitre and Rocchesso aim to test how well the descriptors can communicate corresponding sounds. The participants thus have to correctly identify the referent sound from a list of nine sounds per group (nine-alternative forced choice), based on the vocalizations and verbalisations recorded previously [11]. Although overall accuracy is good, the former fare much better than the latter in terms of correct sound identification. Again, vocal imitations are not affected by the type of sound they refer to, whereas verbal descriptions are indeed. The authors assume that the lower recognition accuracy found for verbalisations may result from participants not being able to reproduce either the spectral information (e.g. pitch, energy spectrum, resonance modes) or the temporal information (e.g. temporal envelope) of the signal, or both. In fact, highest recognition accuracies are reached when both content patterns are communicated correctly. Furthermore, being able to identify the cause of the sounds greatly improves recognition by the participants.

The same approach is taken by Lemaitre et al. [10] in their paper about the identification of sound events based on vocalizations. Essentially, participants are required to sort a set of sound imitations into clusters according to their spectral and temporal properties. In the best-case scenario, these clusters coincide exactly with the sound categories previously defined by the researchers. The aforementioned everyday referent sounds are imitated prior to classification, yielding a total of 72 imitations. Participants are indeed able to classify the imitations with convincing accuracy, suggesting that vocalizations alone can effectively explain referent sounds by conveying the features necessary for their identification [10]. This insight reinforces the assumption that vocal imitations are a promising way of communicating non-verbal expressions (including, but not limited to sound design [12]).

Drawing on the papers presented above, Cartwright and Pardo [3] create a similar system for studying the connection between imitations and their descriptions. Much like Lemaitre and Rocchesso [11], the researchers create four distinct groups of sounds, albeit leaning towards the musical aspect of sounds:

1. **Everyday sounds**: wide variety of everyday acoustic events (e.g. brushing teeth)
2. **Acoustic instruments**: especially of orchestral nature, all playing musical note C (e.g. plucked violin)
3. **Commercial synthesisers**: basic synthesiser sounds derived from factory presets in Apple's Logic Pro[8] music production software (e.g. resonant clouds synthesiser)
4. **Single synthesisers**: more complex synthesiser sounds with FM and AM capabilities, all playing musical note C

Again, participants are given both imitation and recognition tasks. The first group of participants is asked to imagine sounds simply by looking at a label (e.g. "barking dog")

[8] http://www.apple.com/logic-pro

before recording their imitation of the respective sound (e.g. audio file of a dog barking). Contrarily to [11] though, they are not allowed to use verbal descriptions. Subsequently, the second group has to describe randomly played vocal imitations of aforementioned referent sounds and identify them from a list of ten audio files (ten-alternative forced choice) [3]. Not surprisingly, the subset of everyday sounds again yields the most accurate results. According to the authors, this may be due to the familiarity but also reproducibility of these sounds. In fact, the harder to imitate a sound seems to be, the less likely is it to be recognised correctly by the participants.

While all of the above publications deal with the connection between descriptions and sounds as well as the manual categorisation thereof, the automatic classification of audio files by an algorithm ultimately is the goal the Master's thesis at hand is contributing to. Even though the referenced papers give invaluable insight into the characteristics of vocalizations and verbalisations, the novel paradigms in collaborative audio design can only be fully exploited when the notions driving the human classification can be transferred onto an algorithm. This is why in the next subsection, frameworks for the classification and retrieval of audio files are discussed, eventually leading to the phonetic parameters used for algorithmic categorisation of audio files.

2.2 Classification frameworks

After examining both vocal imitations and verbal descriptions of sound events, one follow-up step might be their distribution into distinct groups according to their characteristics. Cartwright and Pardo [3], amongst others, deal with the task by asking participants to assign labels to the referent sounds played to them. This classification, however, is solely performed by humans, without attempting at automating the process. Dessein and Lemaitre [4] therefore introduce an experiment which, similar to the previous publications, requires participants to imitate given sounds and subsequently classify them into groups. Additionally though, the researchers introduce an algorithm able to divide the corresponding audio files into clusters by analysing their acoustic properties. The goal of their work is to emphasise the characteristics which allow listeners to classify the imitations into groups, only to employ these properties for automatic classification.

First of all, the authors introduce a total of 12 environmental sounds, evenly grouped into *liquid*, *solid*, *gas* and *electric* types. Each sound is imitated by all of the experiment's six participants (equally distributed by gender). Eventually, the following distance matrix D is computed for all imitations x and y [4]:

$$D_{x,y} = \begin{cases} 0 & \text{if } x \text{ and } y \text{ are of the same type} \\ 1 & \text{else} \end{cases} \tag{1}$$

The participants are subsequently asked to divide the imitations into groups, receiving no limitations on how to do so. Naturally, this free classification is performed using rather different strategies. Nevertheless, the majority of participants' descriptions exhibit causal

and semantic similarities (i.e. the cause and meaning associated with the identified source) [4]. This backs the assumption that spectro-temporal properties are indeed essential for the imitation and recognition of sounds. In order to process the classification data, both pairwise and between-participant similarity coefficients are calculated from the descriptive results, which have been classified into similarity groups prior thereto.

Next, the imitations are converted into a tree by means of the distance matrix and similarity coefficients computed before. The distance between two nodes (i.e. two imitations) is called height of fusion h. This parameter gives rise to the computation of the inconsistency coefficient i, calculated as follows. Parameters μ_d and σ_d thereby denote the mean and standard deviation of the height of fusion of the d highest non-leaf subnodes, respectively [4]:

$$i = \frac{h - \mu_d}{\sigma_d} \tag{2}$$

Using i, the participants' classifications are divided into seven distinct clusters. Note that these clusters do not necessarily have to correspond to the natural types of the sounds specified above. In fact, the imitated sounds are clustered by common acoustic invariants rather than source [4]. In general, the clusters serve as the ground truth which the classification algorithm will, in the best case, comply with. A two-level hierarchy is established over the clusters by considering the height of fusion. Each level is explained by a set of descriptors, corresponding to the imitations' acoustic features. The first two descriptors are chosen for the first level of hierarchy, whereas the latter four are applied to the second [4]:

1. **Modulation amplitude**: repetitive pattern or one-block sound
2. **Mean of spectral centroid**: unvoiced with high-frequency noisy part or voiced with low-frequency fundamental part
3. **Temporal increase of energy envelope**: brutal attack or smooth attack
4. **Effective duration of energy envelope**: short duration or long duration
5. **Standard deviation of spectral centroid**: varying timbre or constant timbre
6. **Mean of zero-crossing rate**: high pitch or low pitch

All of the above descriptors enable clear distinction of sounds based on their acoustic features. Thus, the clusters can be formed automatically depending on their respective imitations exceeding or falling below a certain descriptor threshold, arbitrarily determined by the researchers. The discussion of the paper points out that flawless discrimination of the imitations into clusters can be achieved using solely the descriptors mentioned above, thus supporting the assumption that sound files can indeed be classified by analysis and parameter extraction. Moreover, the acoustic similarities overlap with the similarities used by the participants, further backing the approach taken by Dessein and Lemaitre [4].

Rocchesso and Mauro [19] also focus on the clustering of sound samples by investigating their acoustic similarity. In fact, their paper already constitutes a preliminary

study on the notions dealt with in the EU project SkAT-VG. The authors make use of several features that, although introduced here, will be thoroughly discussed in the next subsection. Contrarily to [4] though, the human categorisation of sound events plays a subordinate role in their experiment.

First of all, a total of 152 audio segments are gathered, each representing a single sound event exactly 500 milliseconds of length. Subsequently, the dimensionality of the sonic space is reduced by applying Principal Component Analysis (PCA) on the signals' magnitude Fourier spectra. Thus, the data set can be reduced to 95 segments, which nevertheless account for 90 % of the energy. By doing so, however, invertibility has to be given up, thereby conceding the possibility of reproducing each sound by inverse transformation [19]. With this reduced sonic space at hand, the researchers then extract several features from each audio file, taken from the toolboxes in [13] and [16]. In particular, mean and interquartile range values of the signals' *spectral flux*, *spectral centroid*, *roughness*, *flatness*, *entropy*, *skewness* and *Root Mean Square (RMS) energy* are considered. Additionally including temporal characteristics of the signals, a total of 18 features are defined [19].

After selecting two principal components for the distribution, the following clusters are computed using aforementioned features. Exemplarily, Figure 1 shows these groups plotted in a two-dimensional space (with one component on each axis). The larger circles denote the cluster representatives (i.e. their spectral centroids).

1. **First cluster**: pitched or intonated sounds
2. **Second cluster**: continuous sounds
3. **Third cluster**: impulsive or temporally evolving sounds

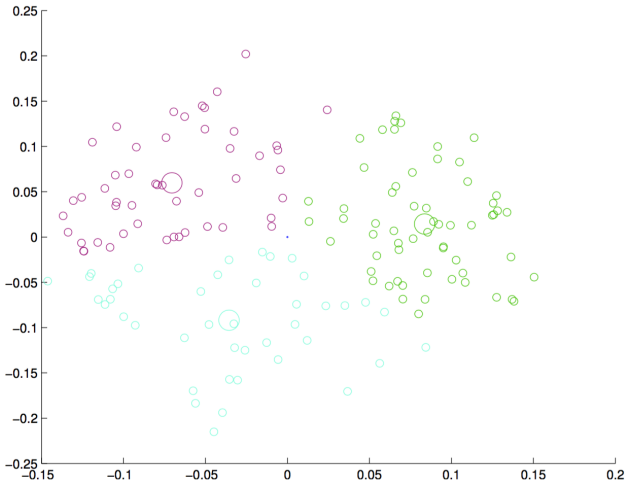

Fig. 1. Color-coded clusters plotted against their principal components' values [19].

Whereas Rocchesso and Mauro [19] mostly concentrate their efforts on the automatic classification based on feature extraction, they nevertheless conduct a small human verification experiment. Three subjects who are not familiar with the project are asked to categorise the sound files into aforementioned clusters. Their classification accuracies amount to 48 %, 53 % and 65 %, respectively, while a random assignment would return 33.33 % [19]. Despite this rather inadequate accuracies, the fact that the signals corresponding to the cluster representatives (i.e. the clusters' spectral centroids) all sound perceptually different from each other supports the idea put forward by the authors.

Furthermore, Blancas and Janer [1] present a paper that introduces the idea of incorporating vocal imitations as input queries for content-based systems. In particular, the Application Programming Interface (API) of the *freesound.org*[9] collaborative database, whose content is uploaded by registered users, is chosen for the task. According to the authors, the aim of the work is to fill the gap between voice interaction and sound retrieval [1]. Therefore, an experiment is conducted by extending the approach taken in [10] with a machine learning algorithm. Once more, a number of features from the toolbox in [13] is extracted, and Support Vector Machine (SVM) is chosen as the classification algorithm due to its high accuracy. Additionally, 10-fold cross-validation is performed on the data [1]. Figure 2 depicts the well-arranged workflow of the researchers' experiment, the concept of which will also be applied to the Master's thesis at hand.

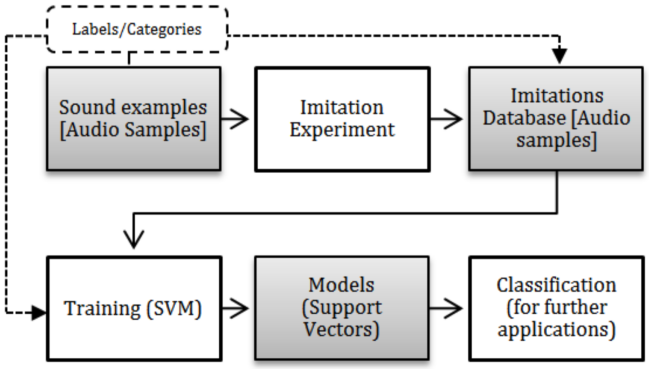

Fig. 2. Exemplary workflow of the imitation and classification of sounds [1].

Subsequently, referent sounds are selected from the *freesound.org* database, divided into the categories *cat*, *dog*, *car* and *drums*. Note that not the researchers themselves, but rather the users of this collaborative system are responsible for defining the categories' names, suggesting a promising approach to transferring this rather theoretical domain onto real-life applications. A total of 17 participants are asked to vocally imitate the selected sounds, thereby refraining from using onomatopeoia. As a matter of fact, cat and

[9] http://www.freesound.org

dog sounds are rather easily imitated by the participants, whereas car and drum sounds seem to be more difficult [1].

It might therefore not come as a surprise that the training data of the machine learning algorithm reveals very good accuracy for these cat and dog sounds, while still retaining good accuracy for car and drum sounds. This suggests that participants are able to convey both spectral and temporal information of the referent sounds in their imitations. Moreover, the integration of the algorithm into aforementioned collaborative database yields promising results as well. Whereas nearly 700 sounds are presented when searching for the category "cat meowing", and a little over 200 when using the category name plus a specific tag (e.g. an action or entity inside the category), only five sounds are displayed when combining the name with the corresponding imitation. The same holds true for the category "drums hihat", which yields just over 16.000 sounds when looking solely for the name, around 600 when combining the name with a specific tag, and merely 20 when querying with the name and the corresponding imitation [1]. These numbers clearly highlight the potential of vocalizations when it comes to facilitating the search for suitable sound samples in content-based systems.

Kwon and Kim [9] too recognise the importance of easy-to-use search and editing tools for content authoring systems based on vocal imitations. The authors propose a novel method that modifies both the time scale and energy of sound samples simply by processing sound-imitation words [9]. It is thus possible for the user to create, search and edit content more easily and naturally, using solely one's voice. Specific knowledge of or experience in the field of audio technology is not required, therefore making the retrieval of sound files suitable for non-experts as well. Four kinds of sound samples are used in the experiment: *car horns*, *doorbells*, *coughs* and *dog barks*. Similarly to [1], analysis of the signals strongly encourages the possibility of using spectro-temporal features for sound classification tasks.

2.3 Sound analysis and feature extraction

When performing the automatic classification of audio files discussed above, the sounds' acoustic characteristics are analysed and features thereof are extracted. Two relevant publications are briefly introduced in this subsection, namely the MATLAB Toolbox for the extraction of musical features from audio by Lartillot and Toiviainen [13] and the Timbre Toolbox for the extraction of audio descriptors from musical signals by Peeters et al. [16].

In the former, an overview of a number of features related to *timbre*, *tonality*, *rhythm* or *form* of a sound signal is given. The authors especially highlight the ease of use and adaptability of their toolbox, which is able to support multiple input types due to a backend object-oriented architecture. In order to thereby retain computational efficiency, frequently used components as well as musical features are interdependently assembled, much like building blocks of a larger structure [13]. Thus, redundancy is avoided while

giving rise to flexibility. Interestingly enough, each musical feature is related to one of the many dimensions traditionally defined in music theory [13]. Figure 3 gives an overview of the toolbox's vast feature extraction possibilities. The components related to pitch, tonality and dynamics appear in bold, whereas the features associated with timbre are highlighted by italics. In addition, bold italics denote the components related to rhythm. The acronym SOM stands for Self-Organising Maps.

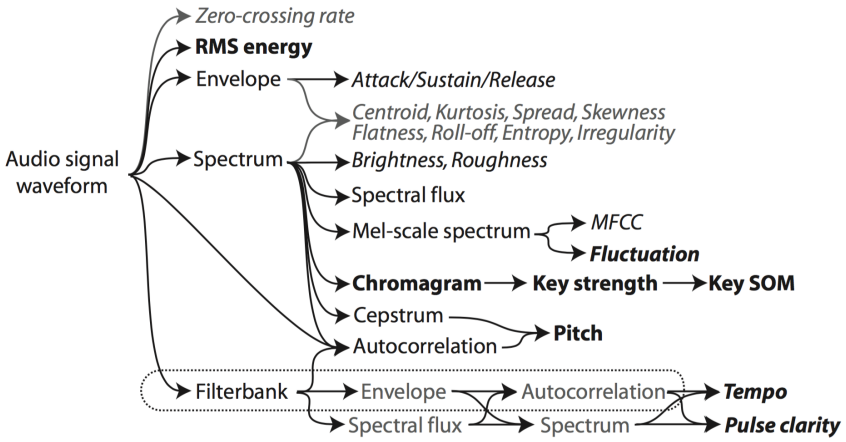

Fig. 3. Overview of the features which can be extracted by the toolbox described in [13].

Another feature extraction system worth mentioning is the Timbre Toolbox by Peeters et al. [16]. According to the authors, it is an instrument for the measurement of the acoustical structure of complex sound signals. Various input representations (including Fourier transformation) are put forward, wherefrom a number of audio descriptors capturing temporal, spectral, spectro-temporal as well as energetic properties of the referent sounds are selected [16]. These features are divided into ten groups determined by hierarchical clustering. Similar features are employed by Peeters [15] in the EU project CUIDADO[10] (Content-based Unified Interfaces and Descriptors for Audio/Music Databases available Online, 2001-2003). In particular, the latter aims at developing new applications by employing audio/music content descriptors, including, amongst others, the design of appropriate description structures and the development of extractors for deriving high-level information from audio signals [23].

All of these publications and the features introduced by them should ultimately serve as a starting point to the methodology of the thesis at hand. At the core of the program lies the auto-correlation function, used for the extraction of its most important features. Thereby, periodicity and pitch of a signal are determined, eventually leading to the discrimination of the sound sketching parameters discussed in the following subsection. The

[10] http://anasynth.ircam.fr/home/english/projects/cuidado

theoretic sketch of this function is presented as part of a paper by Cheveigné and Kawahara [2], who put forward a fundamental frequency estimator for speech and music. The corresponding equation - which serves as the foundation for the student's MATLAB implementation - is in turn introduced in a PhD thesis about vocal melody extraction by Rao [17]:

$$d_t(\tau) = \sum_{n=0}^{N-1} (x(n) - x(n+\tau))^2 \tag{3}$$

Described with words, the auto-correlation vector d_t is defined as the cumulative sum of all squared differences between every value $x(n)$ and $x(n+\tau)$, whereas $n = 0...N-1$ denotes the index of a signal with N samples and τ an offset which the resulting vector is a function of. It is important to bear in mind that N needs to be at least twice as long as one period of the smallest frequency to be detected.

Depending on the fundamental frequency F_0 of an analysed signal, the smallest value (i.e. difference between two samples) of the resulting vector indicates a possible periodicity. In fact, when dividing a signal's sampling frequency F_s by the index τ_0 of this value, the result directly corresponds to the sound's assumed fundamental frequency:

$$F_0 = \frac{F_s}{\tau_0} \tag{4}$$

This rather straightforward calculation serves as the first step of the feature computation in the program at hand. Based on their sonic characteristics, the input audio files are analysed by a machine learning algorithm, which determines whether or not they contain aforementioned parameters. Further illustrations of the equations as well as the classification method are given in Section 3 by means of the corresponding source code. Note again that all computations are executed merely on the time domain, without the need of performing Fourier transformation. However, Friberg and his team [5] develop the same classification system in the frequency domain, thus additionally covering the spectral characteristics of signals. Combined with the temporal features put forward in this work, a robust and versatile classification system is created.

2.4 Sound sketching parameters

The aim of the EU project SkAT-VG is to understand and exploit human vocalizations. The researchers at KTH in particular focus their attention on extracting the primitives of vocal production, as they emerge from non-linguistic tasks [20]. For this reason, four voice artists (equally distributed by gender) were asked to imitate a number of everyday sounds. Thereafter, the recordings were studied and annotated by phonetic experts, utilising articulators specially created for this purpose. While these annotations and the resulting database of imitations are presented in more detail in Section 3, the parameters leading to the creation thereof are introduced in the following paragraphs.

In order to understand how these parameters are defined, it is crucial to first offer a brief overview of the mechanisms responsible for speech production. In general, humans generate sounds by driving an airstream past one or more obstacles [7]. The nature of the produced sound is thereby defined both by its initiation mechanism and its source (i.e. obstacle). Figure 4 depicts a simplified model of the human throat, with the airstream represented by arrows pointing upwards and the obstacle (in this case the tongue arching on the soft palate) denoted by a red circle [14]. Drawing on the origin of this airstream, Helgason [7] introduces the following types of sound initiation:

1. **Pulmonic**: airstream is initiated from the lungs
2. **Glottalic**: airstream is initiated from the glottis[11] (i.e. the opening between the vocal folds)
3. **Velaric**: airstream is initiated from the tongue

All three types can be used with air flowing either inwards (*ingressive*) or outwards (*egressive*). Thus, a total of six sound initiation types are defined. Conversely, when discussing the obstacles which the airstream has to pass, Helgason [7] and Friberg et al. [5] agree on the following types of sound sources:

1. **Myoelastic**: muscle and elastic tissue (i.e. vocal folds) are made to oscillate in an airstream
2. **Phonation**: muscle and elastic tissue (i.e. vocal folds) are also made to oscillate in an airstream, albeit with faster frequency
3. **Turbulent**: airstream is channelled through a constriction in the glottis or the vocal tract

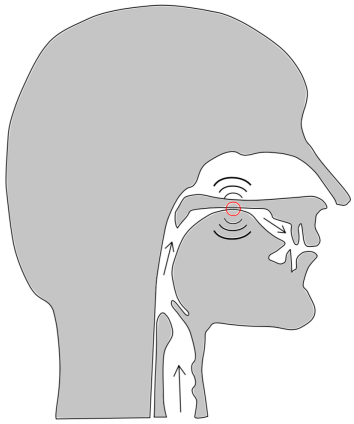

Fig. 4. Model of the human throat while making a sound [14].

[11] http://en.wikipedia.org/wiki/Glottis

While the latter type is perceived as fricative noise, the former two usually manifest themselves as periodic sounds or intermittent vibrations, with the myoelastic type exhibiting lower frequencies than the phonation type (around 20-70 Hz and 70-140 Hz, respectively). Interestingly enough, it is possible for all three types to appear in the same signal. This might lead to problems in the algorithmic classification concerning myoelastic and phonation signals, especially if one frequency is a multiple of the other (e.g. 50 Hz for myoelastic and 100 Hz for phonation).

Aforementioned initiation and source types can be combined in multiple ways. The most frequently used combinations in both speech and vocal imitations are by far pulmonic egressive myoelastic and pulmonic egressive phonation. This is due to the vocal folds being a highly versatile instrument when it comes to controlling the onset, offset, timbre and oscillation frequency of a sound [7]. In general, everyday sounds are almost exclusively created with air flowing outwards (egressive initiation). Prominent exceptions thereof are emotive sounds, e.g. sucking in air through one's teeth to indicate pain [7].

Before examining the annotation tiers constituting the three source types, a couple of sounds are exemplarily specified for each. Myoelastic and phonation types usually make use of the pulmonic egressive initiation mechanism to create animal sounds (e.g. cat meowing, cow mooing, elephant trumpeting) and engine noises (e.g. motor rumbling). The pulmonic ingressive initiation mechanism also produces animal sounds (e.g. dog barking, pig squealing, crow cawing) as well as imitations of squeaky noises (e.g. wiping a window pane) [7]. Surprisingly, both glottalic and velaric initiation mechanisms exhibit no vocalizations whatsoever for myoelastic and phonation source types in the recordings compiled at KTH.

Turbulent sources are, on the other hand, used for more "basic" sounds. For example, the pulmonic egressive initiation mechanism is common when imitating interactions with solid materials (e.g. knocking, scraping) and the sounds of gases in motion (e.g. blowing, puffing, hissing) [7]. Pulmonic ingressive initiation, although rather scarce in imitations, is nevertheless employed for aforementioned emotive sounds. Much like before, glottalic turbulent sounds barely manifest themselves in vocalizations. However, while velaric egressive initiation is common in the imitation of sputtering liquids as well as spray cans, velaric ingressive initiation is used in the production of click-like sounds (e.g. trickling water).

With these definitions of sound initiation and source types (henceforth referred to as parameters) in mind, the team at KTH analysed the voice artists' recordings in terms of the articulatory and aerodynamic conditions involved in their production [7]. The data was annotated using the ELAN software[12] by the Dutch Max Planck Institute for Psycholinguistics[13]. Figure 5 shows a screenshot of the software during the process of annotating the sound of a ringing alarm clock imitated by one of the female performers [8]. Eight layers (or rather the acronyms thereof), clearly visible as the rows' labels on the

[12] http://tla.mpi.nl/tools/tla-tools/elan
[13] http://www.mpi.nl

left side, denote the annotation tiers that are created in the database. In the retrieval of the source types, the following combinations of four of those tiers are used:

1. **Myoelastic**: Supraglottal laryngeal vibration, Nasality, Lip manner, Tongue manner
2. **Phonation**: Vocal fold phonation, Nasality, Lip manner, Tongue manner
3. **Turbulent**: Vocal fold phonation, Nasality, Lip manner, Tongue manner

Each layer can feature one of many variables at a time, whose allocations in the aforementioned screenshot are recognisable in the respective rows below the signal's waveform (e.g. *modal*, *oral*, *open*, *flat* and so on). It is thus possible to extract several different subsets for each combination of annotation layers, depending on which variables are used. However, only two subsets are derived from the raw data files for each of the three source types listed above: one exhibiting the respective parameter, and one lacking it. Therefore, the corresponding variable values have been determined by KTH's phonetic experts during the annotation process. These subsets are ultimately used as ground truth in the program at hand, which is examined more closely in the following section.

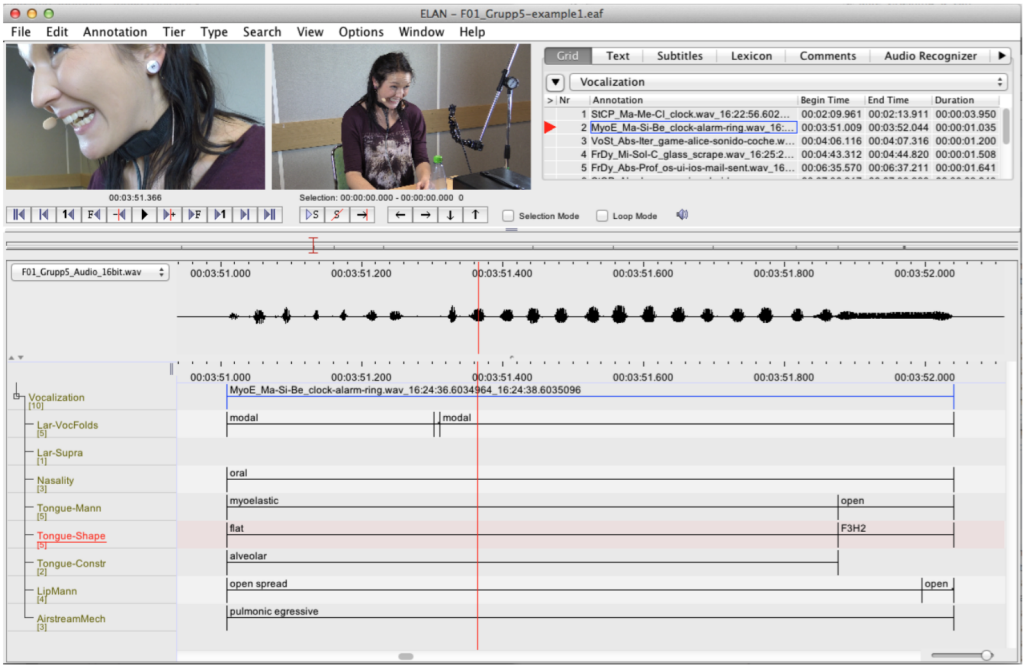

Fig. 5. Screenshot of the ELAN software used for the annotation process [8].

3 Methodology

This section deals with the individual concepts employed in the Master's thesis at hand, explaining each in more detail and inserting small pieces of source code if needed. The recorded audio files are discussed first, followed by the annotations based on the layers introduced above. Subsequently, the annotated audio samples extracted from the raw data are described. These samples (henceforth called fragments) will serve as input to the auto-correlation function, ultimately leading to the creation of features using different statistical measures thereof. Eventually, two prediction models are introduced to forecast the results of the feature-based classification: PLS regression [6, 22] and SVM. To ensure flexibility and applicability in real-world scenarios, N-fold and leave-one-out cross-validation are additionally performed on the data sets. Even though each subsection will introduce its respective concepts rather individually, cross-references to other subsections will be made when necessary.

3.1 Audio files

The sound imitation files, which serve as the source for fragment extraction further down the line, were recorded at KTH in the wake of the EU project SkAT-VG [5]. Four voice actors (equally distributed by gender) were asked to imitate a number of sounds that were played to them. The subjects, between 20 and 40 years of age and native speakers of the Swedish language, were recruited through an agency and paid for their participation in the experiment [21]. The recordings were carried out in a sound-proof booth, with audio and video tracks taped separately by microphones and cameras. Instead of permanently starting and stopping the recording, the whole session was recorded in one take, thus including potential errors and mishaps in the imitation process. Participants were not revealed the source of the sound (e.g. by means of a label indicating its origin), as they should concentrate solely on its imitation.

Upon consideration of the different sound categories put forward in Section 2.1, it becomes evident that the selection of referent sounds was a rather challenging task. Eventually, the team at KTH decided on a total of 60 sounds, thereby striving to reach a good balance between the use of voiced and fricative articulations in an attempt to provide the same number of referent sounds for each source parameter (myoelastic, phonation, turbulent) [21]. The sounds, none of which were longer than 10 seconds, were available in the WAV file format.

Supported by the partnering researchers at IRCAM, four groups of sounds were subsequently drafted. Table 1 lists all groups and their respective sounds in alphabetical order. For the final classification, however, the mechanical interaction sounds were split into three subgroups (gases, liquids, solids), thereby yielding a total of six groups. Since the files bear slightly cryptic titles in the reference literature, the sources of the sounds rather than their file names are mentioned in the table.

Table 1. Referent sounds used in the imitation task.

Abstract: 13 sounds – Android 1 (Popup) – Android 2 (Tweeters) – Game 1 (Alice, car) – Game 2 (OLO, main menu) – Game 3 (Tiny Thief, robot magnet loop) – Game 4 (Super Brothers, marker) – iOS 1 (mail sent) – iOS 2 (message received) – iOS 3 (code typed) – Pica concha – Plastic pipe – Wii (item selected) – Windows Phone 7 (menu)	**Animals**: 10 sounds – Common raven – Cow (mooing) – Dog (barking, growling) – Donkey (braying) – Elephant (trumpeting) – Horse (whinnying, blowing) – Lion (growling) – Mosquito – Pig (snorting) – Songbird (singing)
Machines: 20 sounds – Alarm (ringing) – Blender – Bulldozer – Car door – Clock – Coffee (perking) – Door (squeaking) – Drill hammer – Electric shaver – Fog horn (whistling) – Honda 650 – Honda Accord – Large pepper mill – Lawn mower – Motorbike – Siren (wailing) – Smoke detector – Tractor – Wood (lathing) – X-ray buzzer	**Mechanical interactions**: 17 sounds – Air (flowing) – Anvil (struck) – Bubbles (boiling) – Can (crushed) – Cloth (ripped) – Cold gusts (howling) – Container (filled) – Egg (cracked) – Explosion – Flame thrower – Glass (scraped) – Glass vase (sloshed) – Jet of water – Knife sharpener – Match (struck, ignited) – Paper (ripped) – Sanding

As mentioned above, the whole imitation session was recorded in one take. To ensure good sound quality, the resulting files were stored in the WAV file format using 24 bit sample resolution and 48 kHz sampling rate, thus leading to rather large file sizes. However, since ELAN can only support up to 16 bit, the files were truncated to this smaller resolution prior to annotation, with the sampling rate staying the same [21]. The imitations were subsequently named according to their imitator's acronym, sound group, type of recording and sample resolution, resulting in names such as *F01_Animals_Audio_16bit_trim.wav* for the animal sound group imitated by the first female performer. Table 2 offers an overview of all the resulting files, including their length in the format *mm:ss* as well as their size in megabytes. Note that for the first male performer, some files are split into two parts.

Table 2. Audio files resulting from the imitation task.

#	**File name**	**Subject**	**Group**	**Length** (mm:ss)	**Size** (MB)
1	F01_Animals_Audio_16bit_trim.wav	Female 1	Animals	24:20	140.20
2	F01_Grupp1_Audio_16bit_trim.wav	Female 1	Group 1	16:03	92.50
3	F01_Grupp2_Audio_16bit_trim.wav	Female 1	Group 2	19:31	112.50
4	F01_Grupp3_Audio_16bit_trim.wav	Female 1	Group 3	24:36	141.80
5	F01_Grupp4_Audio_16bit_trim.wav	Female 1	Group 4	17:58	103.50
6	F01_Grupp5_Audio_16bit_trim.wav	Female 1	Group 5	11:42	67.50
7	F02_Animals_Audio_16bit_trim.wav	Female 2	Animals	14:35	84.10
8	F02_Grupp1_Audio_16bit_trim.wav	Female 2	Group 1	13:33	78.10
9	F02_Grupp2_Audio_16bit_trim.wav	Female 2	Group 2	15:18	88.20
10	F02_Grupp3_Audio_16bit_trim.wav	Female 2	Group 3	25:08	144.80
11	F02_Grupp4_Audio_16bit_trim.wav	Female 2	Group 4	25:08	144.80
12	F02_Grupp5_Audio_16bit_trim.wav	Female 2	Group 5	22:05	127.30
13	M01_Animals-1_Audio_16bit_trim.wav	Male 1	Animals	10:28	60.30
14	M01_Animals-2_Audio_16bit_trim.wav	Male 1	Animals	05:42	32.90
15	M01_Grupp1_Audio_16bit_trim.wav	Male 1	Group 1	16:24	94.50
16	M01_Grupp2_Audio_16bit_trim.wav	Male 1	Group 2	09:36	55.40
17	M01_Grupp3_Audio_16bit_trim.wav	Male 1	Group 3	10:55	62.90
18	M01_Grupp4_Audio_16bit_trim.wav	Male 1	Group 4	10:48	62.30
19	M01_Grupp5-1_Audio_16bit_trim.wav	Male 1	Group 5	01:56	11.20
20	M01_Grupp5-2_Audio_16bit_trim.wav	Male 1	Group 5	11:54	68.50
21	M02_Animals_Audio_16bit_trim.wav	Male 2	Animals	11:13	64.60
22	M02_Grupp1_Audio_16bit_trim.wav	Male 2	Group 1	22:41	130.70
23	M02_Grupp2_Audio_16bit_trim.wav	Male 2	Group 2	34:01	196.00
24	M02_Grupp3_Audio_16bit_trim.wav	Male 2	Group 3	21:42	125.00
25	M02_Grupp4_Audio_16bit_trim.wav	Male 2	Group 4	19:49	114.20
26	M02_Grupp5_Audio_16bit_trim.wav	Male 2	Group 5	17:06	98.50

In total, 114 minutes of imitations are available for the first female performer, and nearly 116 minutes for the second. For the first male performer, however, only a little over 77 minutes are on hand, whereas for the second, as much as 126 minutes are available. Moreover, the cumulated file sizes of the recorded imitations are impressive in their own right: for the first female actor, a total of 658 MB are on record, and a little over 667 MB for the second. For the first and second male actor, all files amount to 448 MB and 729 MB, respectively. Considering this immense repository of more than 7 hours and 2.5 GB of imitations, it is perhaps not surprising that the program put forward in the thesis at hand takes some time to extract samples, compute features and classify fragments. Before these processes are explained in more detail though, it is crucial to first understand how fragments can be extracted from the files introduced above.

3.2 Annotations

In order to extract short sound samples containing the three source types in question, knowledge of their temporal location in the recordings is of the essence. Since the algorithm needs to be tuned according to a ground truth before being able to automatically extract fragments from input files, initial annotations were carried out manually by the phonetic experts at KTH, thereby employing the ELAN software introduced before. In particular, the annotation process dealt with the eight layers discussed in Section 2.4, which - using different combinations - form the source parameters of interest and - using different values of these combinations - also yield the parameters' subsets.

It is surely understandable that the manual annotation of amounts of data as vast as the source files discussed in the previous subsection was a rather time-consuming undertaking. In particular, the researchers assigned to the task had to listen carefully to the audio files represented as wave forms, all the while studying the corresponding video files in order to further understand the actor's motions leading to specific annotation tiers (for an example consult Figure 5). It was thereby crucial to keep the amount of human errors to a minimum, since the resulting annotations serve as ground truth for both the student's and Friberg et al.'s classification algorithms.

Upon finishing the annotation process of a source file, its results were exported from ELAN into a text file. Each subset of every parameter was treated separately, thus resulting in a total of six annotation files, named after their respective parameter followed by a suffix indicating the type of subset. For example, the file *myoelastic-0.txt* contains all imitations which the myoelastic parameter is missing from, whereas *myoelastic-1.txt* includes every imitation exhibiting the same. Table 3 gives an overview of these annotation files. Note that the amount of annotations varies between parameters and subsets. In particular, the respective number is always lower in the subsets marked with "1" than in those marked with "0". This is not surprising given the fact that one parameter "holds against" two others when it comes to appearances in imitations. Theoretically assuming perfect distribution of parameters (and ruling out overlapping), each imitation exhibiting a certain parameter would be matched by two others which do not.

Table 3. Text files resulting from the manual annotation task.

#	File name	Parameter	Subset	Number annotations
1	myoelastic-0.txt	Myoelastic	lacking parameter	894
2	myoelastic-1.txt	Myoelastic	exhibiting parameter	232
3	phonation-0.txt	Phonation	lacking parameter	676
4	phonation-1.txt	Phonation	exhibiting parameter	501
5	turbulent-0.txt	Turbulent	lacking parameter	633
6	turbulent-1.txt	Turbulent	exhibiting parameter	592

After completion of the human annotation task, the next step is to study the resulting files' patterns in order to properly extract samples containing the parameters in question. Following are the first ten positive myoelastic occurrences, annotated with ELAN and exported into the file *myoelastic-1.txt*. Note that the last column, namely the path to the data file, is missing. This is simply due to the lack of space in the horizontal dimension of the document at hand, yet not of importance in the discussion of the subject matter.

```
Annot1          Annot2 Annot3     Annot4        Duration Begintime Endtime
L-open          oral   open       ventricular   126      165228    165354
lip_occlusion   velic  TM#        LS#           32       165378    165410
myoelastic_lax  oral   myoelastic LS#           331      233680    234011
lip_occlusion   nasal  TM#        aryepiglottal 83       294234    294317
L-open          nasal  open       aryepiglottal 97       294432    294529
lip_occlusion   nasal  TM#        aryepiglottal 274      295024    295298
L-open          nasal  open       aryepiglottal 16       295673    295689
lip_occlusion   nasal  TM#        aryepiglottal 115      295689    295804
L-open          nasal  open       aryepiglottal 80       296110    296190
lip_occlusion   nasal  TM#        aryepiglottal 184      296190    296374
```

Although the structure might look confusing at first, it is actually rather straightforward. Respectively, the columns denote the variables of the four annotation tiers mentioned before, in addition to the duration of the occurrences in milliseconds as well as their beginning and end. Since the thesis at hand focuses on the extraction of annotated fragments rather than their phonetic origin, a thorough discussion of the latter is not offered at this point. However, interested readers are referred to the reference literature, in particular to the publication by Helgason [8].

Regardless of these annotation values, the most crucial consideration point is the exact temporal locality of the parameters' occurrences, referenced by their start and end times in milliseconds. For example, the first such sample, barely 0.13 seconds of length, is recorded around minute 2, second 45 of the file. It will later be demonstrated that imitations as short as this one might not be able to be classified correctly. Slightly longer occurrences, however, can be processed by the program without problems.

3.3 Fragments

3.3.1 Extraction

With audio and annotation files both available, it is now possible to apply the latter to the former in order to extract annotated audio fragments. This is the first of three core steps in the program at hand, the others being the computation of features and the classification of fragments. Before starting the extraction process, aforementioned files have to be prepared using array structures in MATLAB, which are shown in Tables 4 and 5. However, for the sake of clarity, not all annotations are listed at this time. In fact, only the first ten annotations of the positive myoelastic parameter are specified (cf. Section 3.2). Note that this only serves the purpose of exemplarily demonstrating the way the data is structured in MATLAB. Naturally, the actual program utilizes all annotations in the extraction process.

Table 4. Array structure of the audio data in MATLAB.

'F01_Animals_Audio_16bit_trim'	70083979x1 double	48000
'F01_Grupp1_Audio_16bit_trim'	46232652x1 double	48000
'F01_Grupp2_Audio_16bit_trim'	56246940x1 double	48000
'F01_Grupp3_Audio_16bit_trim'	70879522x1 double	48000
'F01_Grupp4_Audio_16bit_trim'	51763049x1 double	48000
'F01_Grupp5_Audio_16bit_trim'	33739137x1 double	48000
'F02_Animals_Audio_16bit_trim'	42037970x1 double	48000
'F02_Grupp1_Audio_16bit_trim'	39044684x1 double	48000
'F02_Grupp2_Audio_16bit_trim'	44093995x1 double	48000
'F02_Grupp3_Audio_16bit_trim'	72395727x1 double	48000
'F02_Grupp4_Audio_16bit_trim'	72417002x1 double	48000
'F02_Grupp5_Audio_16bit_trim'	63629716x1 double	48000
'M01_Animals-1_Audio_16bit_trim'	30167931x1 double	48000
'M01_Animals-2_Audio_16bit_trim'	16440765x1 double	48000
'M01_Grupp1_Audio_16bit_trim'	47272392x1 double	48000
'M01_Grupp2_Audio_16bit_trim'	27684505x1 double	48000
'M01_Grupp3_Audio_16bit_trim'	31457520x1 double	48000
'M01_Grupp4_Audio_16bit_trim'	31129927x1 double	48000
'M01_Grupp5-1_Audio_16bit_trim'	5577338x1 double	48000
'M01_Grupp5-2_Audio_16bit_trim'	34274974x1 double	48000
'M02_Animals_Audio_16bit_trim'	32304912x1 double	48000
'M02_Grupp1_Audio_16bit_trim'	65360137x1 double	48000
'M02_Grupp2_Audio_16bit_trim'	97987668x1 double	48000
'M02_Grupp3_Audio_16bit_trim'	62502718x1 double	48000
'M02_Grupp4_Audio_16bit_trim'	57098498x1 double	48000
'M02_Grupp5_Audio_16bit_trim'	49257380x1 double	48000

Table 5. Array structure of an excerpt of the annotation data in MATLAB.

'F01_Grupp1'	10	'165228'	'165354'
'F01_Grupp1'	10	'165378'	'165410'
'F01_Grupp1'	10	'233680'	'234011'
'F01_Grupp1'	10	'294234'	'294317'
'F01_Grupp1'	10	'294432'	'294529'
'F01_Grupp1'	10	'295024'	'295298'
'F01_Grupp1'	10	'295673'	'295689'
'F01_Grupp1'	10	'295689'	'295804'
'F01_Grupp1'	10	'296110'	'296190'
'F01_Grupp1'	10	'296190'	'296374'

The MATLAB arrays presented above only contain the most important data in order to avoid overhead. In Table 4, the audio files' respective names are depicted in the first column. The second column contains the actual signal as a sequence of amplitudes, with its sampling rate (in Hz) offered in the third column. Conversely, the annotation data in Table 5 contains in the first column the name of the file wherein the occurrences have been found. The second column includes the length (i.e. amount of characters) of aforementioned name. The last two columns denote the start and end times of the occurrence as specified before. Note that these have not yet been converted into samples, which will however be done before extracting the fragments.

Eventually, the extraction algorithm is executed, operating as follows. First, the name of each audio file is matched against every annotation's target file identifier (denoted by the first columns in Tables 4 and 5, respectively). In order to ensure equal length of both strings, only the leading n characters of the former are considered, with n corresponding to the respective length of the latter. For example, the string *F01_Grupp1_Audio_16bit_trim* is truncated to *F01_Grupp1*, thus corresponding to aforementioned target identifier. Conversely, a string such as *M01_Animals-1_Audio_16bit_trim* is shortened to *M01_Animals-1*. Whenever the comparison yields a positive result (i.e. both strings are equal), a fragment is found and can be extracted. This is done by converting the annotation's start and end times into samples and subsequently copying the corresponding file section into a newly created array. Besides the actual audio data, both the fragment's name and its sampling rate are transferred from the annotation and audio data, respectively.

The following piece of MATLAB code executes the extraction process as described above, thereby writing all annotated occurrences of a specific subset into an array entitled "fragments". The input arrays, already introduced before, are called "files" and "annotations". Note that two nested loops are used to systematically compare each annotation with every audio file. Moreover, the functions *strncmp* and *str2double* are built into MATLAB, guaranteeing maximum speed.

```
fragments = cell(length(annotations),3);
for i = 1:length(annotations)
  for j = 1:length(files)
    if (strncmp(annotations{i,1},files{j,1},annotations{i,2}))
      fragments{i,1} = annotations{i,1};
      startTime = str2double(annotations{i,3})*(files{j,3}/1000);
      endTime   = str2double(annotations{i,4})*(files{j,3}/1000);
      fragments{i,2} = files{j,2}(startTime:endTime);
      fragments{i,3} = files{j,3};
    end
  end
end
```

After the fragments have successfully been stored in an array, they are saved as short audio files in the WAV file format, divided into separate folders according to the names of their parameters. Since two subsets are available for each source type, a value identifying the respective set by either "0" or "1" is included in the fragments' names, alongside a running identifier and the corresponding source file's description. The exact nomination procedure is described in the following MATLAB code. Note that the variable *index* identifies an offset factor used to correctly arrange the fragments. Depending on the running identifier, a variable number of prefix zeros are used. Similarly to the previous piece of code, *isempty*, *num2str* and *audiowrite* are native MATLAB functions.

```
j = 0;
for i = 1:length(fragments)
  if (~isempty(fragments{i,1});
    fileName = [num2str(i+index-j)'_'name{2}'_'fragments{i,1}extension];
    if ((i+index-j) < 10)
      fragments{i,1} = [folder '/000' fileName];
    elseif ((i+index-j) >= 10 && (i+index-j) < 100)
      fragments{i,1} = [folder '/00' fileName];
    elseif ((i+index-j) >= 100 && (i+index-j) < 1000)
      fragments{i,1} = [folder '/0' fileName];
    else
      fragments{i,1} = [folder '/' fileName];
    end
    audiowrite(fragments{i,1},fragments{i,2},fragments{i,3});
  else
    j = j+1;
  end
end
```

3.3.2 Files

With the fragments stored as WAV files, it is now possible to process and manipulate them in a myriad of ways. However, before the algorithm is discussed any further, it is perhaps of interest to once more revisit the extraction process introduced in the previous subsection. In particular, a runtime improvement is put forward which affects the program's speed as a whole.

The basic idea is that, since the annotation data is not usually altered after having completed the corresponding manual task, the extraction process need not be executed every time the program is launched. In fact, only when the annotation data changes it becomes mandatory to re-extract the fragments from their source files. This is why the student has included in the algorithm a query by means of which the users can choose for themselves whether the fragments should be extracted anew (thereby overwriting the old batch) or the results of the initial extraction be loaded instead. Naturally, this option only applies if the fragments have already been extracted before. If no previously stored fragments are found, the program proceeds without this query.

Whatever the user's choice, the loaded data is subsequently stored in a revised version of aforementioned array. The first ten annotated fragments of the positive myoelastic parameter, previously extracted by applying the annotations listed in Table 4 to their corresponding audio files, are demonstrated in Table 6. Whereas the first column contains the names of the fragment files (not including the extensions), the second and third columns feature the actual signals as well as their sampling rates (in Hz). Moreover, in order to be allocated correctly in the course of the program, the fragments' parameters and types as well as their imitators are included in columns four and five. Although this data is essential duplicated from the file name in the first column, redundancy is hereby condoned in favour of easier processing.

Table 6. Array structure of an excerpt of the fragment data in MATLAB.

'0001_1_F01_Grupp1'	6049x1 double	48000	'myoelastic-1'	'F01'
'0002_1_F01_Grupp1'	1537x1 double	48000	'myoelastic-1'	'F01'
'0003_1_F01_Grupp1'	15889x1 double	48000	'myoelastic-1'	'F01'
'0004_1_F01_Grupp1'	3985x1 double	48000	'myoelastic-1'	'F01'
'0005_1_F01_Grupp1'	4657x1 double	48000	'myoelastic-1'	'F01'
'0006_1_F01_Grupp1'	13153x1 double	48000	'myoelastic-1'	'F01'
'0007_1_F01_Grupp1'	769x1 double	48000	'myoelastic-1'	'F01'
'0008_1_F01_Grupp1'	5521x1 double	48000	'myoelastic-1'	'F01'
'0009_1_F01_Grupp1'	3841x1 double	48000	'myoelastic-1'	'F01'
'0010_1_F01_Grupp1'	8833x1 double	48000	'myoelastic-1'	'F01'

The exact process of importing the fragment files sorted by parameter and subsequently creating an array structure such as in Table 6 is explained by the following piece of MATLAB code. Once more, *fileparts*, *audioread* and *strsplit* are built-in functions. Note that the variable "parameter" needs to be selected by the user at the beginning of the program. Although this ultimately restricts the algorithm to dealing with just one parameter at a time, it is a decision consciously made in order to clearly separate the three source types from each other.

```
paths = loadPaths(['Fragments/' type]);
j = 1;
for i = 1:length(paths)
  [~,name,extension] = fileparts(paths{i});
  if (strcmp(extension,'.wav'))
    [y,fs] = audioread(paths{i});
    partsPath = strsplit(paths{i},'/');
    partsName = strsplit(partsPath{3},'_');
    fragments{j,1} = name;
    fragments{j,2} = y;
    fragments{j,3} = fs;
    fragments{j,4} = [partsPath{2} '-' partsName{2}];
    fragments{j,5} = partsName{3};
    j = j+1;
  end
end
```

Before concluding this discussion of fragments and their extraction, it is perhaps helpful to take a look at the overall distribution thereof. Thus, Tables 7-9 illustrate the quantity of positive and negative fragments for every subject individually as well as in total. Note that the amount of fragments for all subjects (located in the first column) correlate with the number of annotations in Table 3. While all imitators exhibit roughly the same quantity of fragments, the positive and negative groups deviate from each other more strongly. The reason for the latter is explained in Section 3.2.

Table 7. Number of extracted fragments for the myoelastic parameter.

	All subjects	**Female 1**	**Female 2**	**Male 1**	**Male 2**
Negative	894	243	250	224	177
Positive	232	82	30	88	32
Total	1126	325	280	312	209

Table 8. Number of extracted fragments for the phonation parameter.

	All subjects	**Female 1**	**Female 2**	**Male 1**	**Male 2**
Negative	676	211	150	183	132
Positive	501	132	135	143	91
Total	1177	343	285	326	223

Table 9. Number of extracted fragments for the turbulent parameter.

	All subjects	**Female 1**	**Female 2**	**Male 1**	**Male 2**
Negative	633	200	152	165	116
Positive	592	151	142	177	122
Total	1225	351	294	342	238

3.4 Auto-correlation

Whereas the theoretical background of the auto-correlation function has already been briefly discussed in Section 2.3, this subsection introduces an algorithmic implementation thereof. Before presenting the corresponding code, however, it is perhaps of interest to examine the algorithm's input variables, namely τ_{min}, τ_{max}, *hop* and *block*, more closely. On the one hand, the former two are calculated by means of the upper and lower limits of the frequency range which the auto-correlation is due to operate in:

$$\tau_{min} = \frac{F_s}{F_{max}} \tag{5}$$

$$\tau_{max} = \frac{F_s}{F_{min}} \tag{6}$$

On the other hand, the *hop* size defines how far the algorithm should "jump" ahead in the signal after finishing one iteration, while the *block* size specifies a window (i.e. small section of the signal) which the computation is executed for. Note that the two variables do not necessarily need to exhibit identical values. For example, while an algorithm jumps ahead 100 samples at a time, the computation is simultaneously performed on a window of 200 samples. Thus, the first iteration would see the processing of samples 1-200, with the second treating samples 101-300 and so on. In fact, it is advisable for *block* to adopt a value corresponding to two times the value of *hop*. Moreover, since both variables are specified in milliseconds rather than samples, a conversion is performed as follows:

$$hop_{samples} = hop_{ms} * F_s \tag{7}$$

$$block_{samples} = block_{ms} * F_s \tag{8}$$

All these inputs are used in the following MATLAB implementation of the auto-correlation function, which is based on a code segment taken from the book by Zölzer [25].

Its core computation corresponds to Equation 3, with the exception of signal name x being replaced by *window* and running index n by j. The variable *window* thereby denotes aforementioned section to be processed, whose length is specified by the block size. The auto-correlation is subsequently computed for each sample in this window and each value of τ at a time, thus yielding a one-dimensional result matrix running along the index of the latter. However, since the algorithm periodically jumps ahead after finishing one iteration, it becomes inevitable to include a second dimension in the output array, specified by a constantly increasing value k. In particular, this index denotes different parts of the signal (henceforth called frames), whose lengths are defined by the hop size.

```
k = 0;
for i = 1:hop:(length(y)-(block+tauMax))
  window = y(i:i+(block+tauMax));
  k = k+1;
  for j = 1:block
    for tau = tauMin:tauMax;
      results(tau,k) = results(tau,k)+(power(window(j)-window(j+tau),2));
    end
  end
end
```

A fair amount of work has been put into finding the optimal values for the input variables specified earlier. In the end, F_{max} and F_{min} are set to 200 and 20 Hz, respectively, which no imitations are believed to exceed or fall below. Hence, τ_{min} and τ_{max} amount to 240 and 2,400 when computing above equations with a sampling rate of 48 kHz. Moreover, hop and block sizes are each set to 20 and 40 milliseconds, or 960 and 1.920 samples when working with the same F_s.

Exemplarily applying these input variables to the arbitrarily chosen myoelastic file *0910_1_F01_Grupp1.wav*, an output array containing 2,400 rows (i.e. values of τ) and 158 columns (i.e. frames) is created. Note that the first 239 rows are negligible since the minimum value of τ is 240. Table 10 shows an excerpt of this resulting matrix, thereby including rows 1,780-1,790 and columns 5-9. One number per column is emphasized, denoting the smallest value thereof. Upon referencing Section 2.3, it becomes evident that, when dividing a signal's sampling frequency F_s by the index τ_0 of this value, its fundamental frequency F_0 can be obtained. These indices and frequencies (the latter denoted in Hz) are specified therafter.

Even though aforementioned results describe only a fraction of the actual data, the basic concept of the previously introduced auto-correlation function applies to every signal. In fact, the algorithm computes an output matrix for each parameter's 1,100+ fragments, thereby employing the same set of input parameters. In the next subsection, the fundamental frequencies exemplarily presented in Table 8 are subsequently utilized, amongst others, to compute a specific set of features.

Table 10. Excerpt of the results of the auto-correlation function for one imitation.

	Frame 5	**Frame 6**	**Frame 7**	**Frame 8**	**Frame 9**
τ **= 1780**	0.0796	0.0800	0.0650	0.0517	0.1159
τ **= 1781**	0.0792	0.0711	0.0618	0.0519	0.1049
τ **= 1782**	0.0800	0.0638	0.0599	0.0533	0.0953
τ **= 1783**	0.0821	0.0579	0.0591	0.0555	0.0867
τ **= 1784**	0.0855	0.0531	0.0591	0.0587	0.0789
τ **= 1785**	0.0904	0.0494	0.0603	0.0640	0.0726
τ **= 1786**	0.0965	0.0473	0.0631	0.0714	0.0683
τ **= 1787**	0.1035	0.0472	0.0673	0.0799	0.0655
τ **= 1788**	0.1116	0.0489	0.0727	0.0886	0.0633
τ **= 1789**	0.1211	0.0523	0.0794	0.0974	0.0613
τ **= 1790**	0.1319	0.0572	0.0876	0.1075	0.0601
τ_0	1781	1787	1784	1780	1790
F_0 (in Hz)	26.95	26.86	26.91	26.97	26.82

3.5 Features

3.5.1 Computation

Besides the extraction of fragments and the classification thereof, the computation of features is the second core step in the program at hand. The input data is thereby derived from the auto-correlation matrix generated in the previous subsection. However, whereas only the smallest value of each column is emphasized in Table 8, a total of four such characteristics (which the features are subsequently calculated from) are eventually employed:

1. Global minimum
2. Second global minimum
3. Local minima
4. Local maxima

Rather self-explanatory, the former two denote the smallest as well as second-smallest values of a data structure, respectively. A local minimum, however, occurs whenever a data item is neither immediately preceded nor followed by another item with smaller value. Conversely, a local maximum exhibits the highest value in its immediate proximity. Normally, multiple occurrences of the latter two are found in an arbitrary signal. In order to illustrate this concept, an overview of the different types of minima and maxima is given in Figure 6[14]. Note that their graphic representations tend to resemble valley floors and mountain peaks.

[14] http://en.wikipedia.org/wiki/Maxima_and_minima

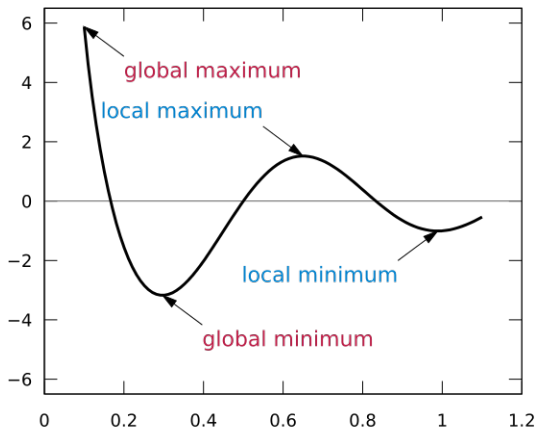

Fig. 6. Overview of the different types of minima and maxima.

Subsequently, these characteristics are applied to the auto-correlation matrix of every imitation. Much like in the above algorithm, each frame is processed in its own iteration. In particular, two types of information are derived from each characteristic, namely its values and its indices. By means of the latter, frequencies are calculated as demonstrated in Section 3.4. The former, on the other hand, are henceforth referred to as amplitudes. Note, however, that these are not in any way related to the amplitudes of a signal's waveform, which are not of importance in the algorithm at hand.

Once again referencing the data in Table 10, the global minimum of the first frame exhibits an amplitude of 0.0792 and a frequency of 26.95 Hz, which is in turn based on an index of 1,781. Furthermore, the second global minimum is described by an amplitude of 0.0796 and a frequency of 26.97 Hz, derived from an index of 1780. Continuing this procedure for all characteristics, a total of twelve features are collected for each frame, as specified in Table 11. Note that the local minima and maxima are thereby represented twice, defined by both their arithmetic mean and median. This is due to the existence of multiple values for these characteristics, which in turn calls for the consolidation thereof into a single value. Thereby, arithmetic mean and median appear to be elegant solutions.

Eventually, five statistical measures are calculated for each feature. Note that, whereas the latter exhibit separate values for each frame, the former specify aggregates thereof. In other words, all values featured in one row of Table 11 are merged, similarly to the idea presented above, by means one of the following measures:

1. Median
2. Mean
3. Mean of the absolute value of variation
4. Variance
5. Interquartile range

Table 11. Features for an excerpt of the results of the auto-correlation function.

Feature	**Frame 5**	**Frame 6**	**Frame 7**	**Frame 8**	**Frame 9**
Global minimum, frequency (in Hz)	26.95	26.86	26.91	26.97	26.82
Second global minimum, frequency	26.97	26.88	26.92	26.95	26.80
Local minima, median of frequencies	1118.00	1406.00	1404.00	1403.00	1313.00
Local minima, mean of frequencies	1264.64	1380.55	1385.09	1377.36	1341.67
Local maxima, median of frequencies	1119.00	1400.50	1397.50	1398.00	1314.00
Local maxima, mean of frequencies	1253.69	1378.80	1383.50	1377.30	1337.91
Global minimum, amplitude	0.0792	0.0472	0.0591	0.0517	0.0601
Second global minimum, amplitude	0.0796	0.0473	0.0591	0.0519	0.0602
Local minima, median of amplitudes	1.0106	0.7262	0.6787	0.6391	0.8122
Local minima, mean of amplitudes	0.8642	0.6223	0.6198	0.6271	0.7686
Local maxima, median of amplitudes	1.6984	1.9780	1.9801	1.8886	1.7507
Local maxima, mean of amplitudes	1.7447	1.9809	1.9915	1.8992	1.7926

All measures are common statistical characteristics, with the exception of *mean of absolute value of variation*, which yields an indication of a signal's continuity by computing the differences between its frames. Eventually, by applying all five measures to each of the twelve features, 60 values are derived in total. For the sake of the argument, these results are henceforth referred to as features as well. However, a representation thereof in a table is not offered at this point.

Before proceeding any further, it is nevertheless crucial to expand the above list by two features, respectively based on the median values of a signal's global and second global minimum frequencies. In particular, a Gaussian bell curve is considered, with its peak defined by the assumed fundamental frequency of each parameter. For the myoelastic and phonation types, 40 Hz and 100 Hz are arbitrarily chosen as base frequencies. Due to its fricative nature, however, the turbulent type does not exhibit such periodicity and is thus unsuitable for this type of calculation.

Eventually, the Gaussian bell curve is drawn using a bandwidth of 30 Hz and its amplitudes normalized in order to satisfy $x_i \leq 1$. Hence, considering aforementioned definition of peaks, the frequency range extends from 10 Hz and 40 Hz ($x = 1$) to 70 Hz for the myoelastic type and from 70 Hz and 100 Hz ($x = 1$) to 130 Hz for the phonation type. Subsequently, the amplitudes of the previously introduced frequency medians under the curve are conveyed as features. As an example, the Gaussian bell curve for exemplarily used imitation is depicted in Figure 7. The median values of 26.95 Hz and 26.96 Hz are thereby denoted by the same vertical red line (due to their close proximity), with the resulting amplitudes both amounting to $x_1 = x_2 = 0.4296$. These values are ultimately added to the list for the myoelastic and phonation parameters, thus yielding a total of 62 features for the former two types while retaining aforementioned 60 features for the turbulent parameter.

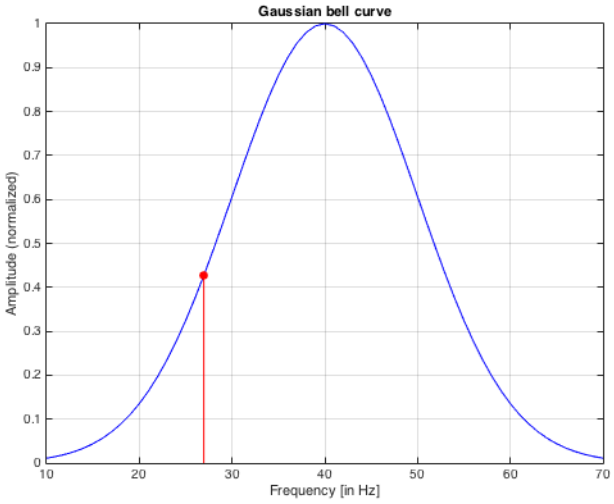

Fig. 7. Gaussian bell curve peaking on the assumed myoelastic base frequency.

Now that the selection of features is complete, it is perhaps of interest to take a closer look at how the respective algorithm is embedded in the program at hand. In particular, two important variables thereof have yet to be defined, namely $factor$ and $minimumLength$. The former, if applicable, describes the factor by which a fragment's F_s is downsampled prior to its computation:

$$F_{s\_new} = F_s / factor \tag{9}$$

For example, when considering the current sampling rate of 48 kHz for all fragments, a factor of 2 would result in $F_{s\_new} = 24kHz$, whereas a factor of 1 would leave F_s unchanged (i.e. $F_{s\_new} = F_s$). This mechanism has been implemented in order to reduce the computation time of the algorithm, based on the notion that merely every nth data point has to be considered when working with a downsampling factor of n.

Conversely, the latter denotes the minimum length that a fragment needs to exhibit in order to be eligible for feature computation. This restriction stems from the notion that at least a certain number of frames have to be computed for each signal in order to meaningfully and faultlessly apply the statistical aggregates discussed earlier (including, but not limited to, *mean of absolute value of variation* and *interquartile range*). Following a fair amount of testing, four frames are determined as the minimum amount which the program can still be executed without errors for. Subsequently consulting the MATLAB code in Section 3.4, it becomes evident that the corresponding auto-correlation loop is only correctly iterated four times or more if the following condition is satisfied:

$$length_{fragment} > (hop * 3 + block + \tau_{max}) \tag{10}$$

Inserting the previously defined values of 960 samples for *hop*, 1,920 samples for *block* and 2,400 samples for τ_{max} into this equation, a fragment length threshold of 7,200 samples (or 150 milliseconds with a sampling rate of 48 kHz) is yielded. Interestingly enough, this corresponds to three cycles of a vibration of 20 Hz, which in turn is the lowest periodic frequency any parameter can exhibit [5]. Note that an alteration of aforementioned downsampling factor effects the minimum length as well. That being said, an overview of how many fragments per parameter are respectively eligible or ineligible is offered in Table 12, implying computation with no downsampling. What's more, in Section 4 these variables are changed at will in order to demonstrate the algorithm's performance for different instances of its inputs.

Table 12. Overview of the fragments' applicability for feature computation.

length > 150 ms	**myoelastic**	**phonation**	**turbulent**
Eligible (negative/positive)	456 (307/149)	465 (243/222)	481 (284/197)
Ineligible (negative/positive)	670 (587/83)	712 (433/279)	744 (349/395)
Total (negative/positive)	1126 (894/232)	1177 (676/501)	1225 (633/592)

Eventually, the feature computation algorithm described above is executed for all qualified fragments. This process takes some time due to the complexity of the autocorrelation's input data. The following MATLAB code illustrates the loop wherein the respective function *computeFeatures* is called. Upon completion, the resulting features are stored in an array for analysis before being employed in the classification of fragments. Likewise, the humanly annotated ground truth is saved thereafter, its values amounting to either 0 or 1 depending on whether the respective parameter is absent or present in the corresponding fragment.

```
j = 0;
for i = 1:length(fragments)
  parameter = strsplit(fragments{i,4},'-');
  minimumLength = ((hop*3+block)*fragments{i,3})+(fragments{i,3}/fMin);
  if (length(fragments{i,2}) > minimumLength)
    [data,names] = computeFeatures(fragments{i,2},fragments{i,5},...
                                   factor,fragments{i,3},...
                                   fBase,fMin,fMax,hop,block);
    j = j+1;
    featuresTemp(j,1:length(data)) = data;
    groundTruthTemp(j,1) = str2double(parameter{2});
  end
end
```

3.5.2 Analysis

A fair amount of thought has been given to the possibility of creating separate feature lists for each parameter in order to achieve slightly better classification results, while simultaneously sacrificing adaptability and generic applicability of the program at hand. In the end, the student decided in favour of the latter by employing the sole overall feature list introduced above. That being said, the only exception thereto are the amplitudes under the Gaussian bell curve, which are missing for the turbulent parameter.

Even so, it might nevertheless be desirable to gain insight into each feature's individual significance before launching the classification process. Thus, all of them are correlated with the ground truth as well as with themselves by means of the following built-in MATLAB function:

```
[R,P] = corrcoef(data,'rows','pairwise');
```

Thereby, *data* denotes the input array wherefrom the correlation results are derived. For the cross-correlation between the features and the ground truth, this matrix is composed of both of their arrays, while for the correlation between the features proper, it consists solely of the former. Conversely, parameters R and P identify the correlation coefficient and its significance, respectively. In particular, each relationship between two arbitrary input values i and j is denoted by $R \in$ [-1...1], with -1 representing the strongest negative correlation, 0 representing no correlation, and 1 representing the strongest positive correlation. Furthermore, significance value P is utilised to test the null-hypothesis, which in turn claims that no relationship exists between i and j. Extending this notion, a rating system is established as part of the algorithm in order to assess the importance of each correlation, denoted by stars in ascending order of significance:

$$P(i,j) > 0.05 \longrightarrow \text{no stars} \tag{11}$$

$$P(i,j) <= 0.05 \longrightarrow \text{one star } (*) \tag{12}$$

$$P(i,j) <= 0.01 \longrightarrow \text{two stars } (**) \tag{13}$$

$$P(i,j) <= 0.001 \longrightarrow \text{three stars } (***) \tag{14}$$

Eventually, the results of above procedure are displayed in the MATLAB console. Since all features are ultimately used in the classification process, their individual significances are not pivotal at this time for a selection thereof. Nevertheless, potential issues with the prediction algorithm can be solved with greater ease by means of this analysis. In conclusion, Table 13 presents the results of both the cross-correlation between the features and the ground truth as well as the correlation between the features proper. Exemplarily, only the mean values of the myoelastic parameter's amplitude-based features are displayed. Note that the features' acronyms, which have been established in the source code upon creation thereof, are used throughout the program.

Table 13. Exemplary (cross-)correlation results for myoelastic features and ground truth.

	Ground truth	**meanA1**	**meanA2**	**meanA3**	**meanA4**	**meanA5**
meanA1	-0.25 ***					
meanA2	-0.25 ***	1.00 ***				
meanA3	0.15 **	0.82 ***	0.82 ***			
meanA4	0.30 ***	0.81 ***	0.81 ***	0.91 ***		
meanA5	0.07	-0.67 ***	-0.67 ***	-0.75 ***	-0.70 ***	
meanA6	0.31 ***	-0.61 ***	-0.61 ***	-0.73 ***	-0.54 ***	0.85 ***

3.6 Classification

3.6.1 Prediction

Eventually, the annotated audio fragments are classified by means of the features computed before. In particular, each parameter's fragments are divided into a positive and a negative group, depending on the presence or absence of the respective source type. Two types of prediction models are thereby used to forecast the results of the classification: Partial Least Squares (PLS) regression [6, 22] and Support Vector Machine (SVM). Even though a comprehensive external library entitled LIBSVM[15] exists for the latter, the corresponding native MATLAB functions are nevertheless employed for both algorithms in order to avoid unnecessary overhead. Subsequently, each prediction model is explained in detail, accompanied by excerpts of the source code whenever appropriate.

On the one hand, PLS regression is a statistical method used for the prediction of responses from a number of factors. In the program at hand, these factors correspond to the features, while the responses are equal to the classification results. Tobias offers in his paper a schematic outline of the regression process, depicted in Figure 8. Thereby, latent (i.e. underlying) variables T and U (also referred to as X- and Y-*scores*, respectively) are extracted from the sample data. Subsequently, the former are used to predict the results of the latter, which in turn the responses are constructed from [22]. Whereas for the related method of Principal Component Regression (PCR), aforementioned X-scores are chosen to explain as much of the factor variation as possible, for the PLS regression strong relationships between successive pairs of scores are of the essence. Even so, the emphasis lies on the prediction process rather than on the relationships between the extracted variables [22].

While PLS is applicable to a large amount of highly collinear factors, the selection of too many thereof can lead to a phenomenon called "over-fitting". Thereby, in spite of the overall number of factors, the amount of latent variables is insufficient for prediction, resulting in the model being perfectly suitable for the training set while failing to accurately predict new data.

[15] http://www.csie.ntu.edu.tw/∼cjlin/libsvm

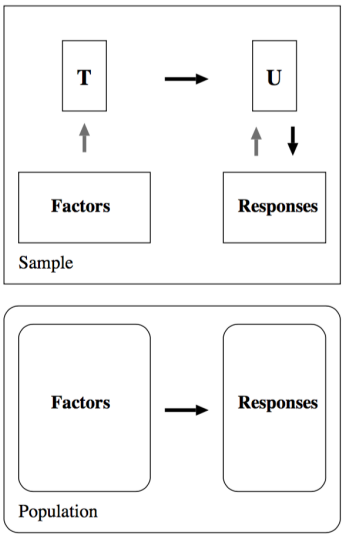

Fig. 8. Schematic outline of the Partial Least Squares regression method [22].

In MATLAB, the regression algorithm is programmed as follows. First, X- and Y-scores are extracted from the factors and responses (i.e. features and ground truth), identified by pX and pY in the following code sample. Note that the built-in MATLAB function *randperm* rearranges the input data in a random fashion.

```
zX = zscore(features.data);
p  = randperm(size(zX,1));
pX = zX(p,:);
pY = groundTruth.data(p,:);
```

Subsequently, training and validation data for both factors and responses are derived from the scores. The variables *trainingIndices* and *validationIndices*, even though introduced here, are discussed at a later time. Eventually, PLS regression is carried out using a certain number of principal components (i.e. linear combinations of input data), and its resulting parameter utilized in the calculation of the prediction matrix. The ideal value of the former is thereby determined by manual testing (cf. Section 4.2).

```
trainingX   = pX(trainingIndices,:);
validationX = pX(validationIndices,:);
trainingY   = pY(trainingIndices,:);
validationY = pY(validationIndices,:);

[~,~,~,~,BETA,~,~,~] = plsregress(trainingX,trainingY,components);
resultPLS            = [ones(size(validationX,1),1) validationX]*BETA;
```

On the other hand, SVM is a classifier model able to divide objects into two separate categories. Thereby, these items are mapped into a coordinate system, exemplarily depicted in Figure 9[16]. In order to provide the model with a classification pattern to follow, a training set is analyzed first, the objects of which have already been assigned to the two groups. The resulting pattern is subsequently stored by the algorithm, and used to predict the category of any new data item which is supplied as input thereafter. All the while, it is desirable for objects of different groups to be located far away from each other in the coordinate space in order to exhibit clear discrimination. Exemplarily, the light blue margins on both sides of the dividing lines in Figure 9 identify the gaps between items of separate categories.

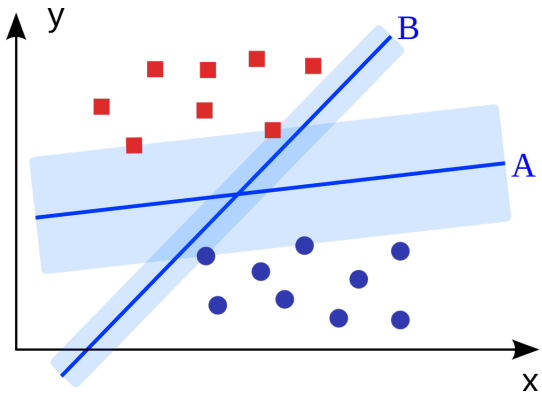

Fig. 9. Classification of objects by the Support Vector Machine method.

When taking a closer look at the source code, it becomes evident that the calculation of training and validation sets is the same for both PLS and SVM. Instead of carrying out the regression, however, a training model is derived using the corresponding arrays, and subsequently employed in the computation of the resulting matrix. Note once again that, although an external library could have been used, native MATLAB functions are preferred due to their efficiency.

```
model     = fitcsvm(trainingX, trainingY);
resultSVM = predict(model, validationX);
```

Upon completion of the prediction algorithms, both of their resulting arrays are on hand for each parameter. However, so far the prediction is merely tailored to the fragments available in the program at hand. In order to ensure flexibility and applicability in real-world scenarios, it is necessary to additionally include cross-validation. Thus, two types thereof are subsequently introduced.

[16] http://de.wikipedia.org/wiki/Support_Vector_Machine

However, before discussing this notion in more detail, it might still be of interest to take a closer look at an alternative version of the PLS regression, which is executed entirely without cross-validation. In particular, the corresponding prediction algorithm is employed in order to receive an indication as to whether or not the whole cross-validation procedure is successful. Perhaps not surprisingly, it also yields better accuracy in comparison with the models discussed previously, due to its specific adjustment to the data at hand. The following code presents the implementation of the PLS regression without cross-validation, with *cumsum* and *max* denoting built-in functions of MATLAB.

```
[zX,~,~]          = zscore(features.data);
[zY,muY,sigmaY] = zscore(groundTruth.data);
[~,~,~,~,~,PCTVAR,~,~] = plsregress(zX,zY,numberFeatures);
cumulativeSum = cumsum(PCTVAR(2,:));
for i = 1:numberFeatures
  temp(i) = 1-(1-cumulativeSum(i))*...
            (numberFragments-1)/(numberFragments-i-1);
end
[~,numberPredictors] = max(temp);
[~,~,~,~,BETA,~,~,~] = plsregress(zX,zY,numberPredictors+1);
result = ([ones(size(zX,1),1) zX]*BETA*sigmaY)+muY;
```

3.6.2 Cross-validation

In order for a prediction model to become robust and versatile, relying on a variation of alternating input data is of the essence. Thus, the corresponding algorithm is trained on a multitude of arrays before being employed in the classification of new items. This notion is called cross-validation, whereof two variations are presented in the program at hand. Both are subsequently discussed in more detail, accompanied by excerpts of the source code if necessary.

On the one hand, N-fold cross-validation is based on the idea of splitting an input data set into N mutually exclusive subsets (i.e. folds). Therefrom, $N-1$ subsets are arbitrarily selected as training data, with the remaining one used for validation. Subsequently, the latter is exchanged for another subset previously involved in training. This procedure is repeated N times, until each subset has once been employed in the validation process. Eventually, all individual results are merged, thus yielding the overall accuracy of the prediction model.

The following MATLAB code introduces N-fold cross-validation by means of selected parts of the algorithm at hand, thereby arbitrarily defining $N = 10$ as the amount of folds. In order to minimize computation errors, the whole method itself is additionally iterated 100 times, implemented by the outer loop in the code below. Ultimately, the indices of the training and validation sets introduced above are computed in the inner loop, before being employed in the creation of their corresponding arrays.

```
numberFragments   = length(groundTruth.data);
numberFolds       = 10;
numberIterations = 100;
samplesPerFold    = round(size(features.data,1)/numberFolds);
for i = 1:numberIterations
  for j = 1:numberFolds
    trainingIndices    = [1:((j-1)*samplesPerFold)...
                          (j*samplesPerFold+1):...
                          size(features.data,1)];
    validationIndices = (j-1)*samplesPerFold+1:j*samplesPerFold;
    if (j == numberFolds)
      if (validationIndices(end) < numberFragments)
        validationIndices = [validationIndices,...
                             validationIndices(end):...
                             numberFragments];
      end
      if (validationIndices(end) > numberFragments)
        lastIndex = find(validationIndices == numberFragments);
        validationIndices = validationIndices(1:lastIndex);
      end
    end
  end
end
```

On the other hand, leave-one-out cross-validation, while working similarly to aforementioned type, focuses on the imitation artists rather than on arbitrarily selected subsets. In particular, upon consideration of M subjects, the data of $M-1$ thereof is selected for training, while the remaining artist's data is used for validation. Again, this procedure is iterated M times, thereby employing each subject in the validation process once, before merging the individual results in order to yield the model's overall accuracy.

Since $M = 4$ imitators are recorded in the program at hand, the algorithm is simultaneously trained on three data sets while being validated on one. The necessary information is thereby conveyed by an array containing a unique index for each speaker. According to Friberg et al., even though leave-one-out cross-validation generally corresponds more closely to real-world scenarios than its N-fold counterpart, in the program at hand it is nevertheless less reliable due to the small number of subjects [5].

Much like before, the following excerpt of the source code depicts the implementation of leave-one-out cross-validation in MATLAB. Note that the procedure is again iterated 100 times in order to minimize computation errors. Furthermore identical to above code, validation and training indices are subsequently derived for the computation of their corresponding data matrices.

```
numberSpeakers    = max(speakerIndices);
numberIterations = 100;
for i = 1:numberIterations
  for j = 1:numberSpeakers
    trainingIndices   = [];
    validationIndices = [];
    index1 = 1;
    index2 = 1;
    for k = 1:length(speakerIndices)
      if (speakerIndices(k) == j)
        validationIndices(index1) = k;
        index1 = index1+1;
      else
        trainingIndices(index2) = k;
        index2 = index2+1;
      end
    end
  end
end
```

Ultimately, the data arrays resulting from the previously discussed prediction and cross-validation procedures contain the fragments' classification results, all of which amount to either 0 or 1 depending on the absence or presence of the corresponding source type. In particular, the following five matrices are on hand for each parameter:

1. **PLS**, 10-fold cross-validation
2. **SVM**, 10-fold cross-validation
3. **PLS**, leave-one-out cross-validation
4. **SVM**, leave-one-out cross-validation
5. **PLS**, no cross-validation

Subsequently, every output matrix is compared to its corresponding ground truth array, thereby yielding the individual overall accuracy of each prediction model. Drawing on these values, the final results of the program at hand are eventually introduced and discussed in more detail in the next section, additionally examining deviations thereof stemming from different instances of the algorithms' input variables.

4 Results

This section presents and evaluates the results of the program at hand, thereby treating each source parameter separately. In particular, the computation of the prediction models' accuracies are discussed, and ideal combinations of input variables therefor revealed. Subsequently, the latter are changed in order to further examine the behaviour of each algorithm. Specifically, the number of PLS components, imitation artists, minimum length of fragments and downsampling factor are subject to modification. Whenever necessary, cross-references to other subsections will be made.

4.1 Overall performance

When referencing the result arrays of the prediction models, it becomes evident that the classification algorithms merely yield as output 0 or 1 for each fragment, thus identifying the corresponding parameter respectively being absent or present therein. Exemplarily, Table 14 offers an overview of ten fragments that exhibit the myoelastic source type. Note that in a similar list in Table 6, imitations were merely selected from the data in running order, whereas in the table at hand, the first ten *eligible* fragments are chosen, all of which satisfy the minimum length of 7,200 samples put forward in Section 3.5. Thereby, the first column denotes their names, with the second column containing the corresponding lengths in samples. Moreover, parameter and ground truth are included in the third and fourth column, respectively. Upon input of the fragments' features, a perfect prediction model should theoretically be able to classify each imitation according to this ground truth. As a matter of fact, when consulting the results presented in the fifth column, the present classification algorithm (PLS without cross-validation) is indeed able to assign each fragment of the sample data set correctly.

Table 14. Excerpt of the (rounded) classification results for PLS without cross-validation.

Name	**Length** (samples)	**Parameter**	**Ground truth**	**Classification result**
0003_1_F01_Grupp1	15889	myoelastic	1	1
0006_1_F01_Grupp1	13153	myoelastic	1	1
0010_1_F01_Grupp1	8833	myoelastic	1	1
0011_1_F01_Grupp1	31345	myoelastic	1	1
0012_1_F01_Grupp1	103729	myoelastic	1	1
0013_1_F01_Grupp1	28993	myoelastic	1	1
0014_1_F01_Grupp1	7729	myoelastic	1	1
0015_1_F01_Grupp1	18193	myoelastic	1	1
0016_1_F01_Grupp1	155521	myoelastic	1	1
0021_1_F01_Grupp1	10177	myoelastic	1	1

However, before matching all output matrices of the prediction models with their corresponding ground truth arrays in order to receive the success rate of each, it is important to understand that, initially, the results of the PLS algorithms do not simply amount to 0 and 1 as stipulated before. In fact, they can adopt a multitude of values. Thus, in order to be successfully matched to the ground truth later on, it is necessary to round them to either one of the two binary values. Thereby, the following method is applied, with x_i denoting the PLS output for an arbitrary fragment i and y_i identifying its final result:

$$x_i > 0.5 \longrightarrow y_i = 1 \tag{15}$$

$$x_i \leq 0.5 \longrightarrow y_i = 0 \tag{16}$$

Whenever a fragment is correctly classified, a corresponding counter is incremented in MATLAB. Note that an increase of this variable is possible in every single iteration of the nested loops introduced before. Upon completion of the last iteration, the prediction models' accuracy is eventually computed by dividing the number of correctly assigned fragments by the total amount thereof. The following code sample depicts this procedure for the PLS and SVM algorithms with both types of cross-validation. Much like before, its inclusion hereby is merely of exemplary nature, and should not be mistaken for the exact source code.

```
numberCorrectPLS = 0;
numberTotalPLS   = 0;
numberCorrectSVM = 0;
numberTotalSVM   = 0;

numberCorrectPLS = numberCorrectPLS+sum(validationY == round(resultPLS));
numberTotalPLS   = numberTotalPLS+length(validationY);
numberCorrectSVM = numberCorrectSVM+sum(validationY == resultSVM);
numberTotalSVM   = numberTotalSVM+length(validationY);

accuracyPLS      = numberCorrectPLS/numberTotalPLS;
accuracySVM      = numberCorrectSVM/numberTotalSVM;
```

For the sake of completion, the MATLAB implementation of the same notion for the PLS model without cross-validation is offered below. Even though similar to the previous code sample, the actual computation of accuracy is nevertheless easier due to the absence of loops.

```
numberCorrect = sum(groundTruth.data == round(result));
numberTotal   = length(groundTruth.data);
accuracy      = numberCorrect/numberTotal;
```

Subsequently, the classification results of each prediction model are conveyed by means of their accuracy values, all of which are listed in Tables 15-17. By default, every speaker's data and no downsampling are used. Furthermore, the minimum fragment length defined before as well as the respective ideal number of PLS components (cf. Tables 18-20) are chosen as inputs. Note that, due to the random division of data items into subsets during 10-fold cross-validation, the results of algorithms employing this method are prone to change ever so slightly from iteration to iteration. However, this fluctuation can be disregarded since its range amounts to less than 0.5 %.

When referencing the following results, it becomes evident that all prediction models yield good accuracies for each parameter. In general, phonation fragments are assigned most precisely, followed by myoelastic and eventually turbulent imitations. For reasons discussed above, PLS without cross-validation is superior to every other method. However, since cross-validation is essential when classifying new data, the best results of any algorithm but the former are highlighted in red. Little over-fitting is present in the models, indicated by the relatively small differences between all results.

Table 15. Overall classification accuracies for the myoelastic parameter.

	PLS	**SVM**
10-fold cross-validation	81.72 %	81.95 %
leave-one-out cross-validation	80.26 %	79.61 %
no cross-validation	85.09 %	

Table 16. Overall classification accuracies for the phonation parameter.

	PLS	**SVM**
10-fold cross-validation	87.45 %	86.73 %
leave-one-out cross-validation	87.53 %	84.95 %
no cross-validation	90.97 %	

Table 17. Overall classification accuracies for the turbulent parameter.

	PLS	**SVM**
10-fold cross-validation	76.02 %	74.08 %
leave-one-out cross-validation	73.39 %	70.27 %
no cross-validation	80.25 %	

Additionally, Figures 10-12 depict coordinate systems in which the (exact) prediction results of PLS regression without cross-validation are mapped against the ground truth. In each, a vertical red line denotes the classification border of 0.5 introduced before. Ideally, all negative fragments (ground truth = 0) are located to the left thereof, and all positive fragments (ground truth = 1) to the right.

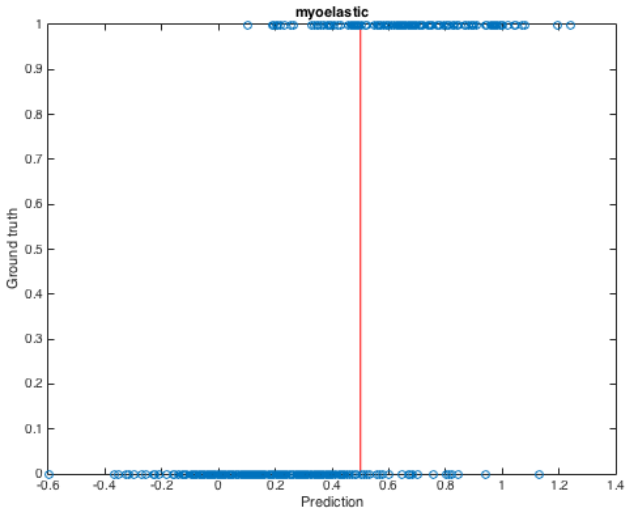

Fig. 10. Overall classification results for the myoelastic parameter.

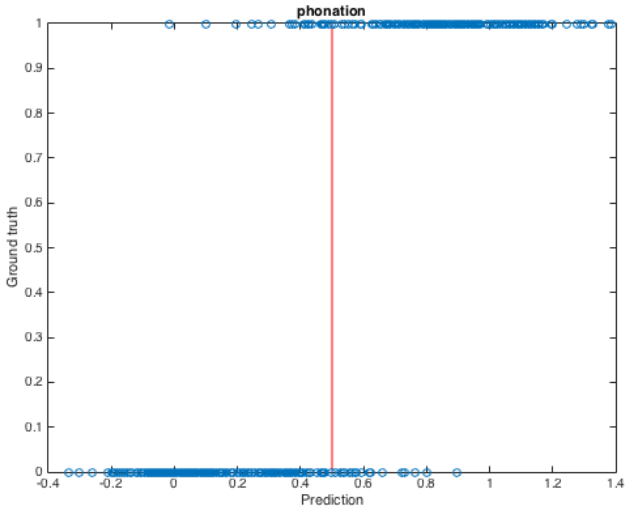

Fig. 11. Overall classification results for the phonation parameter.

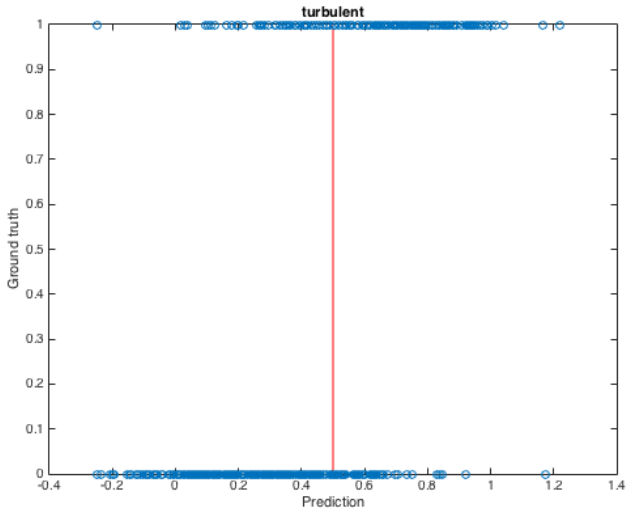

Fig. 12. Overall classification results for the turbulent parameter.

4.2 PLS components

After discussion of the prediction algorithms' overall performance, it might be of interest to examine how PLS regression with cross-validation fares when being executed with varying numbers of principal components (whilst leaving all other variables unaltered). As mentioned before, PLS components denote linear combinations of input data by means of which the prediction is carried out. The results for ten different amounts thereof are shown in Tables 18-20, all the while including both cross-validation methods. Once again, the highest classification accuracies are highlighted in red. For reasons specified above, the latter are also listed in the first columns of Tables 15-17.

In particular, every time a PLS prediction algorithm is called, the ideal number of principal components is conveyed to the respective function. The corresponding code is implemented as follows in MATLAB, with suffixes "1" and "2" denoting 10-fold and leave-one-out cross-validation procedures, respectively, and the array *speakerIndices* featuring a unique number for each subject.

```
accuracyPLS1 = predictClassification1(features,groundTruth,...
                                      components1);
accuracyPLS2 = predictClassification2(features,groundTruth,...
                                      components2,speakerIndices);
```

Table 18. Classification accuracies for the myoelastic parameter using different numbers of PLS components.

# components	10-fold cross-validation	leave-one-out cross-validation
1	69.70 %	67.54 %
2	78.15 %	76.10 %
3	80.84 %	79.17 %
4	80.60 %	78.07 %
5	81.72 %	80.26 %
6	81.29 %	78.95 %
7	81.29 %	77.85 %
8	81.14 %	78.07 %
9	80.77 %	77.19 %
10	80.51 %	77.63 %

Table 19. Classification accuracies for the phonation parameter using different numbers of PLS components.

# components	10-fold cross-validation	leave-one-out cross-validation
1	79.39 %	79.35 %
2	82.20 %	81.51 %
3	85.51 %	84.52 %
4	86.45 %	85.38 %
5	87.45 %	85.81 %
6	87.22 %	86.24 %
7	87.34 %	86.24 %
8	86.95 %	86.24 %
9	87.00 %	86.45 %
10	86.74 %	87.53 %

Table 20. Classification accuracies for the turbulent parameter using different numbers of PLS components.

# components	10-fold cross-validation	leave-one-out cross-validation
1	70.32 %	69.02 %
2	73.56 %	71.31 %
3	76.02 %	73.39 %
4	75.42 %	71.73 %
5	74.93 %	72.14 %
6	75.50 %	73.39 %
7	74.34 %	71.73 %
8	73.65 %	71.52 %
9	73.23 %	70.48 %
10	73.31 %	70.27 %

4.3 Speakers

Whereas before, no discrimination between individual speakers was made in particular, in this subsection each imitation artist's data is treated separately. Thus, potential mistakes in the prediction process can be analysed in more detail. Tables 21-23 show the respective classification results, with each column's highest applicable value highlighted in red. Note that leave-one-out cross-validation is missing, due to its impossibility of being performed when considering one subject at a time only. Interestingly enough, nearly all female speakers exhibit better classification accuracies than their male counterparts.

Table 21. Individual speakers' classification accuracies for the myoelastic parameter.

	Female 1	**Female 2**	**Male 1**	**Male 2**
PLS, 10-fold cross-validation	85.24 %	84.56 %	69.42 %	82.78 %
SVM, 10-fold cross-validation	85.91 %	85.77 %	71.98 %	79.23 %
PLS, no cross-validation	94.16 %	97.96 %	89.60 %	91.67 %

Table 22. Individual speakers' classification accuracies for the phonation parameter.

	Female 1	**Female 2**	**Male 1**	**Male 2**
PLS, 10-fold cross-validation	83.89 %	91.82 %	85.08 %	88.14 %
SVM, 10-fold cross-validation	82.01 %	90.35 %	82.48 %	90.60 %
PLS, no cross-validation	98.51 %	98.99 %	96.00 %	100.00 %

Table 23. Individual speakers' classification accuracies for the turbulent parameter.

	Female 1	**Female 2**	**Male 1**	**Male 2**
PLS, 10-fold cross-validation	82.96 %	73.84 %	68.63 %	75.31 %
SVM, 10-fold cross-validation	79.45 %	77.55 %	65.04 %	72.84 %
PLS, no cross-validation	94.16 %	94.17 %	84.96 %	87.96 %

Moreover, Figures 13-15 depict the female subjects' results of PLS regression without cross-validation in a similar fashion to Section 4.1. Thereby, the predictions of each of the two imitators are marked in different colors. Analogously, Figures 16-18 illustrate the male subjects' corresponding results.

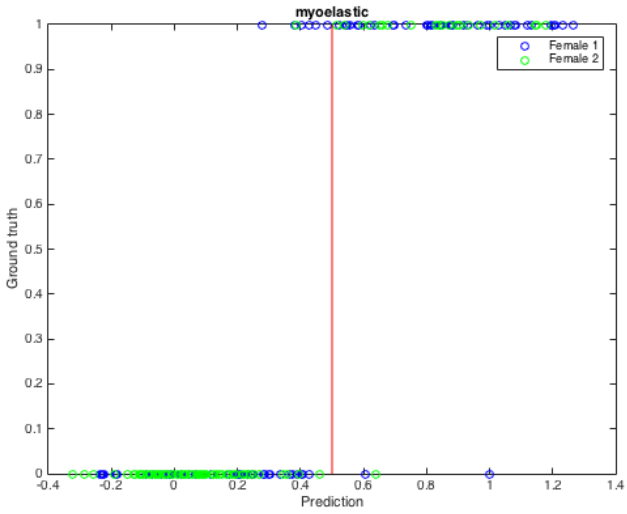

Fig. 13. Female speakers' classification results for the myoelastic parameter.

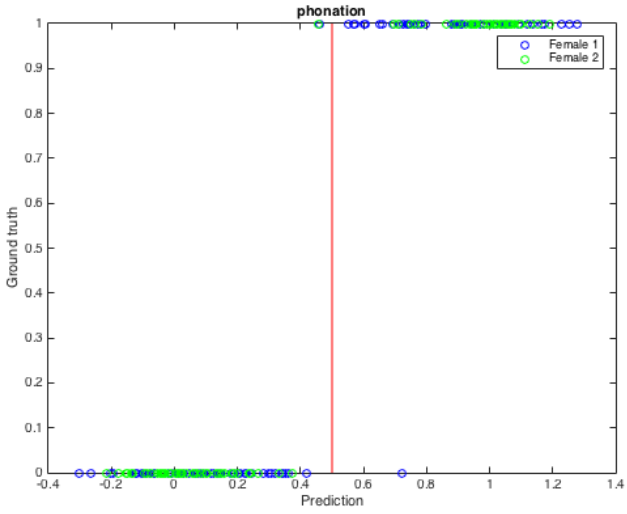

Fig. 14. Female speakers' classification results for the phonation parameter.

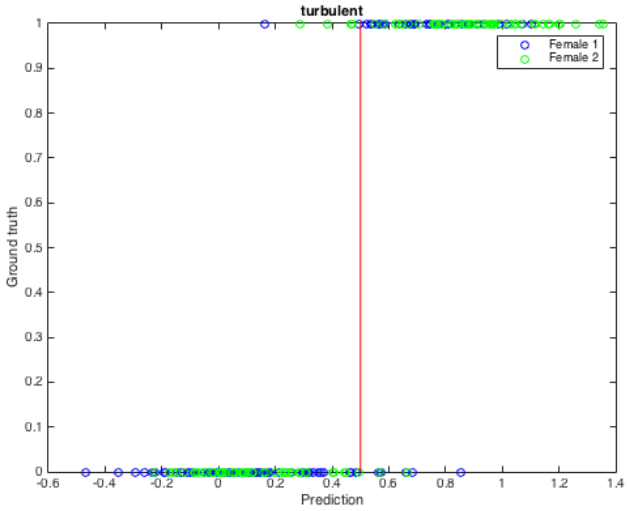

Fig. 15. Female speakers' classification results for the turbulent parameter.

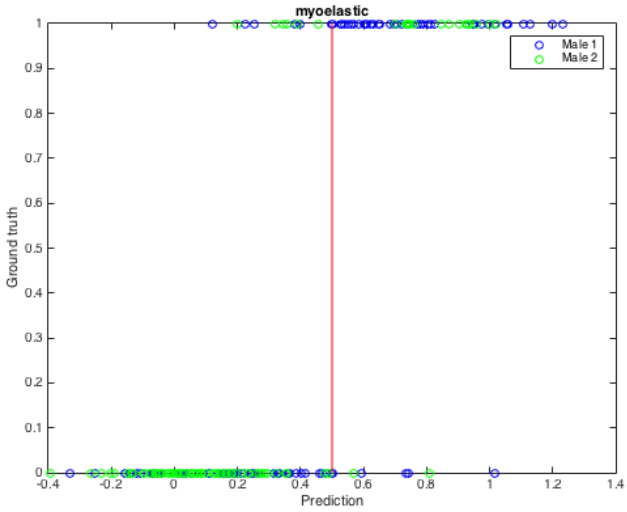

Fig. 16. Male speakers' classification results for the myoelastic parameter.

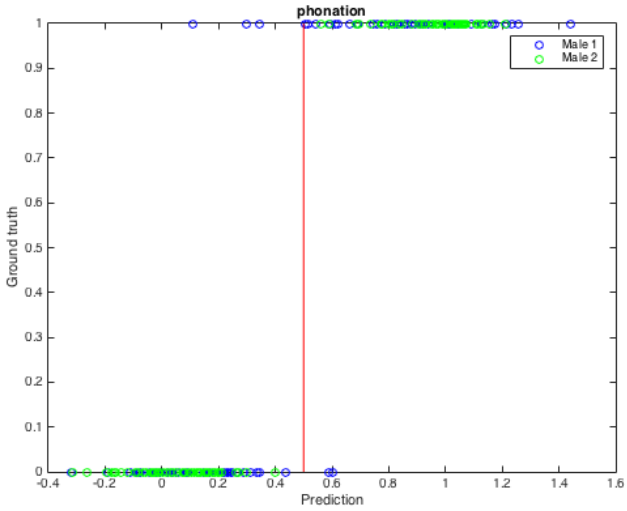

Fig. 17. Male speakers' classification results for the phonation parameter.

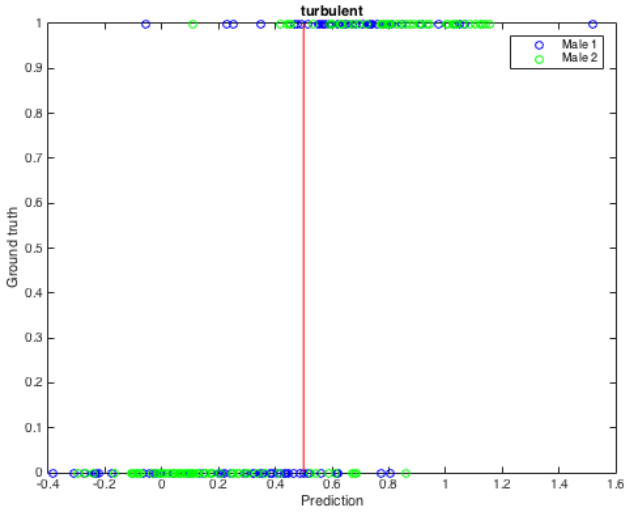

Fig. 18. Male speakers' classification results for the turbulent parameter.

4.4 Minimum fragment length

During implementation of the source code, the student has experimented with different lengths of the fragments involved in the prediction process. In general, longer imitations seem to be classified more precisely due to the wealth of information contained therein. Even though all fragments able to be processed by the algorithm should be selected as inputs, it is nevertheless interesting to compare the program's results when using alternative minimum fragment lengths. Hence, the classification accuracies for multiples of the threshold of 7.200 samples (calculated in Section 3.5) are shown in Tables 24-29. Perhaps not surprisingly, the amount of imitations is thereby inversely proportional to their minimum length.

Table 24. Classification accuracies for the myoelastic parameter using a minimum fragment length of 14.400 samples.

265 eligible fragments	**PLS**	**SVM**
10-fold cross-validation	79.21 %	82.91 %
leave-one-out cross-validation	80.00 %	82.64 %
no cross-validation	88.68 %	

Table 25. Classification accuracies for the phonation parameter using a minimum fragment length of 14.400 samples.

260 eligible fragments	**PLS**	**SVM**
10-fold cross-validation	86.32 %	86.08 %
leave-one-out cross-validation	85.00 %	82.69 %
no cross-validation	93.85 %	

Table 26. Classification accuracies for the turbulent parameter using a minimum fragment length of 14.400 samples.

269 eligible fragments	**PLS**	**SVM**
10-fold cross-validation	72.42 %	70.60 %
leave-one-out cross-validation	70.63 %	71.38 %
no cross-validation	82.53 %	

Table 27. Classification accuracies for the myoelastic parameter using a minimum fragment length of 28.800 samples.

148 eligible fragments	**PLS**	**SVM**
10-fold cross-validation	82.82 %	83.95 %
leave-one-out cross-validation	81.76 %	84.46 %
no cross-validation	95.27 %	

Table 28. Classification accuracies for the phonation parameter using a minimum fragment length of 28.800 samples.

151 eligible fragments	**PLS**	**SVM**
10-fold cross-validation	90.98 %	89.38 %
leave-one-out cross-validation	86.09 %	84.11 %
no cross-validation	98.68 %	

Table 29. Classification accuracies for the turbulent parameter using a minimum fragment length of 28.800 samples.

157 eligible fragments	**PLS**	**SVM**
10-fold cross-validation	74.24 %	75.10 %
leave-one-out cross-validation	75.16 %	73.25 %
no cross-validation	89.81 %	

Similarly to the previous subsections, Figures 19-24 depict the predictions of PLS regression without cross-validation for above minimum lengths. Each table's results are thereby plotted separately, since in this case the combination of two data sets would lead to overly cluttered graphics, caused by an excessive number of overlapping circles. Moreover, upon close inspection of all figures it becomes obvious that in the latter three, fewer fragments are mapped into the coordinate system due to the higher length threshold specified therefor.

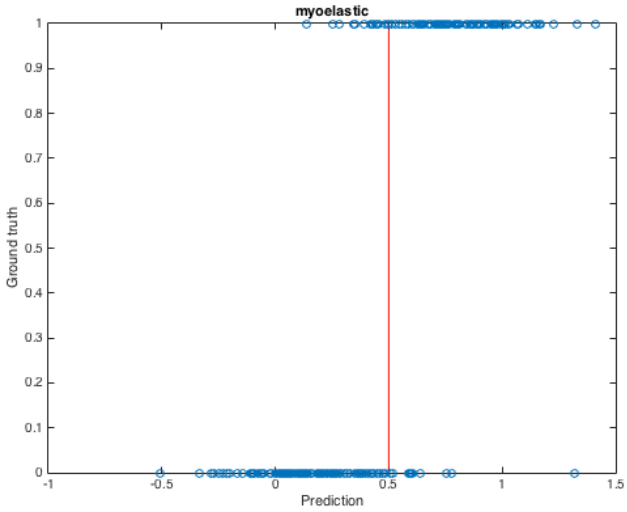

Fig. 19. Classification results for the myoelastic parameter using a minimum fragment length of 14.400 samples.

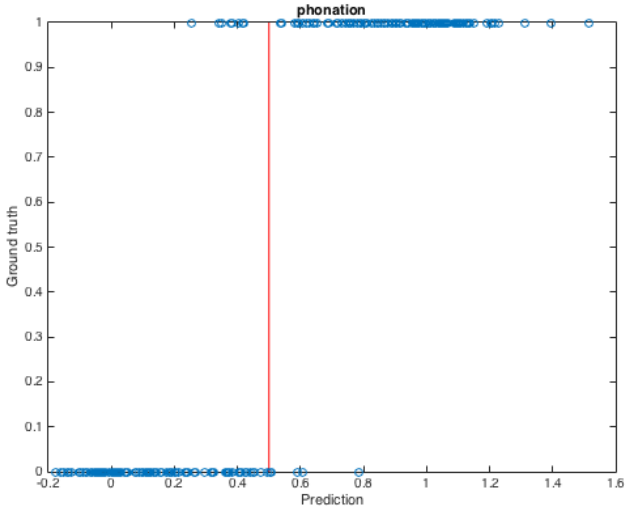

Fig. 20. Classification results for the phonation parameter using a minimum fragment length of 14.400 samples.

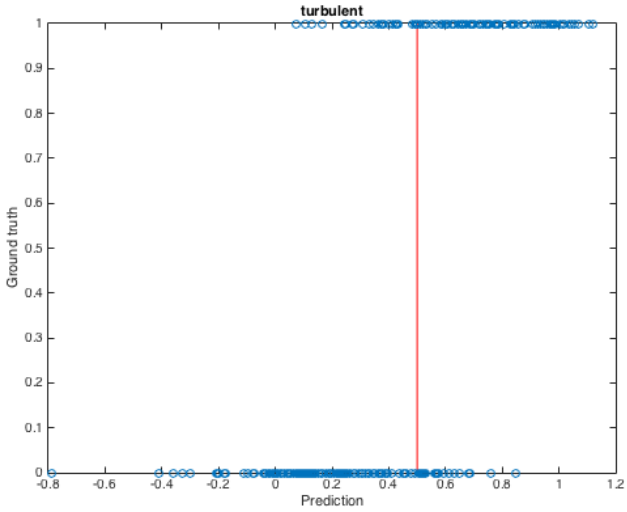

Fig. 21. Classification results for the turbulent parameter using a minimum fragment length of 14.400 samples.

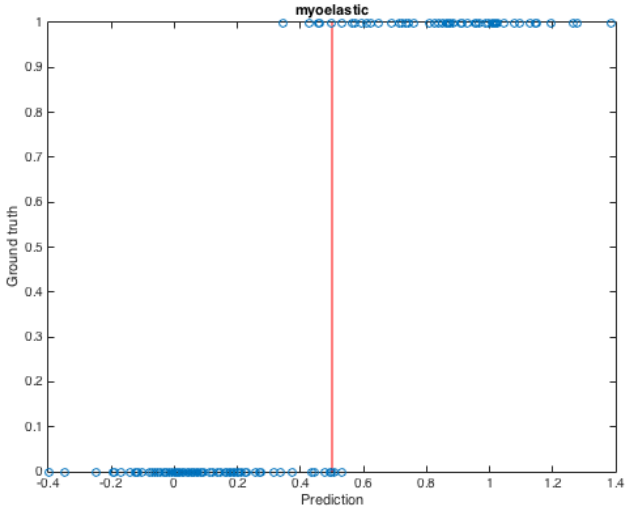

Fig. 22. Classification results for the myoelastic parameter using a minimum fragment length of 28.800 samples.

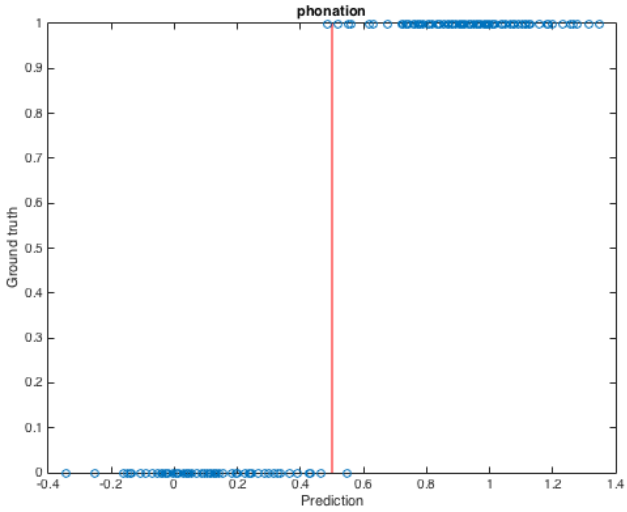

Fig. 23. Classification results for the phonation parameter using a minimum fragment length of 28.800 samples.

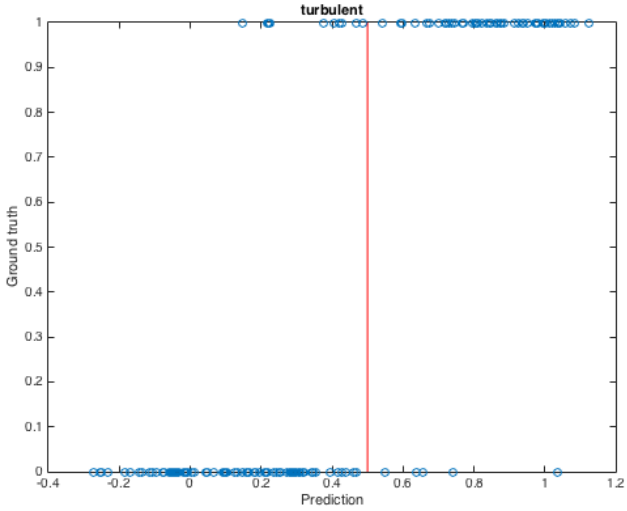

Fig. 24. Classification results for the turbulent parameter using a minimum fragment length of 28.800 samples.

4.5 Downsampling factor

Last but not least, the user-adjustable input variable $factor$, which in turn specifies the fragments' downsampling, is analyzed in more detail. For the calculation of the latter, the corresponding equation has been introduced in Section 3.5:

$$F_{s\_new} = F_s / factor \tag{17}$$

Subsequently, Tables 30-35 give an overview of the algorithm's classification accuracies when using two different downsampling factors (and thus two new sampling rates), with above F_s amounting to 48 kHz by default. Note that, while the number of fragments stays the same in each case, the respective computations are nevertheless executed faster due to less information being contained in the signals.

Table 30. Classification accuracies for the myoelastic parameter using a sampling rate of 24 kHz.

	PLS	**SVM**
10-fold cross-validation	81.80 %	81.54 %
leave-one-out cross-validation	79.39 %	80.48 %
no cross-validation	84.65 %	

Table 31. Classification accuracies for the phonation parameter using a sampling rate of 24 kHz.

	PLS	**SVM**
10-fold cross-validation	87.05 %	87.26 %
leave-one-out cross-validation	86.24 %	86.67 %
no cross-validation	91.18 %	

Table 32. Classification accuracies for the turbulent parameter using a sampling rate of 24 kHz.

	PLS	**SVM**
10-fold cross-validation	74.72 %	74.64 %
leave-one-out cross-validation	70.89 %	69.44 %
no cross-validation	80.25 %	

Table 33. Classification accuracies for the myoelastic parameter using a sampling rate of 12 kHz.

	PLS	**SVM**
10-fold cross-validation	81.46 %	81.00 %
leave-one-out cross-validation	79.82 %	77.85 %
no cross-validation	85.96 %	

Table 34. Classification accuracies for the phonation parameter using a sampling rate of 12 kHz.

	PLS	**SVM**
10-fold cross-validation	87.11 %	88.64 %
leave-one-out cross-validation	86.24 %	86.45 %
no cross-validation	93.12 %	

Table 35. Classification accuracies for the turbulent parameter using a sampling rate of 12 kHz.

	PLS	**SVM**
10-fold cross-validation	75.38 %	75.07 %
leave-one-out cross-validation	71.31 %	68.81 %
no cross-validation	80.04 %	

Eventually, Figures 25-30 show the results of PLS without cross-validation for both new sampling rates. Once again, each table's results are plotted separately in order to avoid overly cluttered figures. Interestingly enough, the prediction remains surprisingly accurate even when fragments are downsampled. Thus, by lowering F_s, major speed improvements can be achieved while potentially sacrificing a fraction of the classification accuracy. In the next section, these improvements are discussed in more detail, alongside the examination of problems arising from the algorithm.

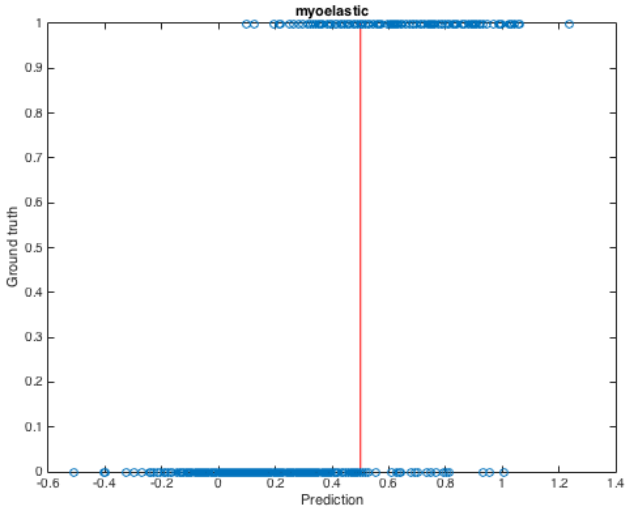

Fig. 25. Classification results for the myoelastic parameter using a sampling rate of 24 kHz.

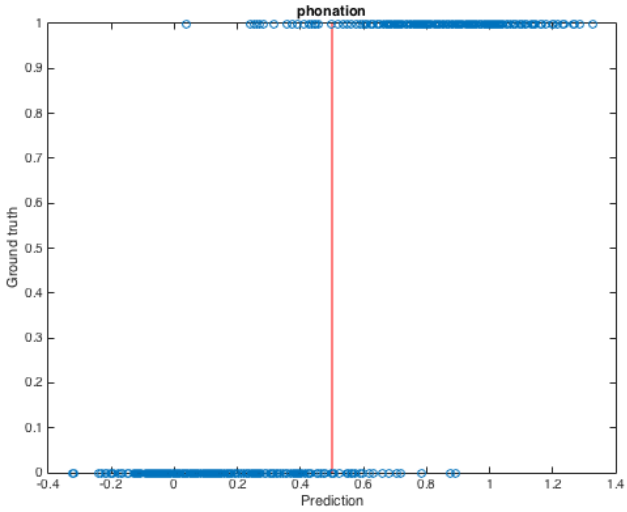

Fig. 26. Classification results for the phonation parameter using a sampling rate of 24 kHz.

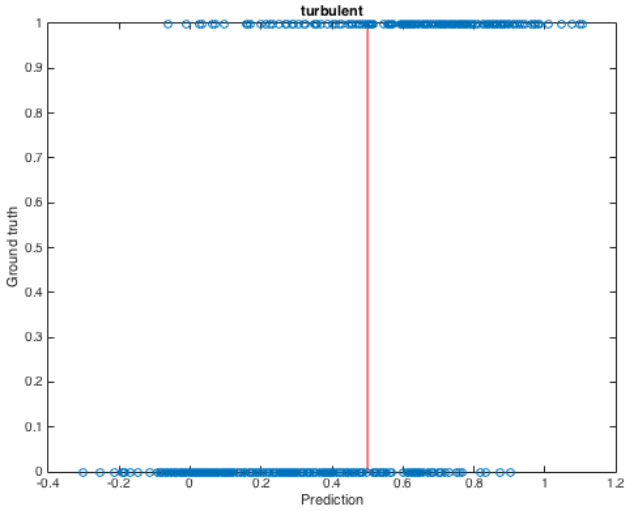

Fig. 27. Classification results for the turbulent parameter using a sampling rate of 24 kHz.

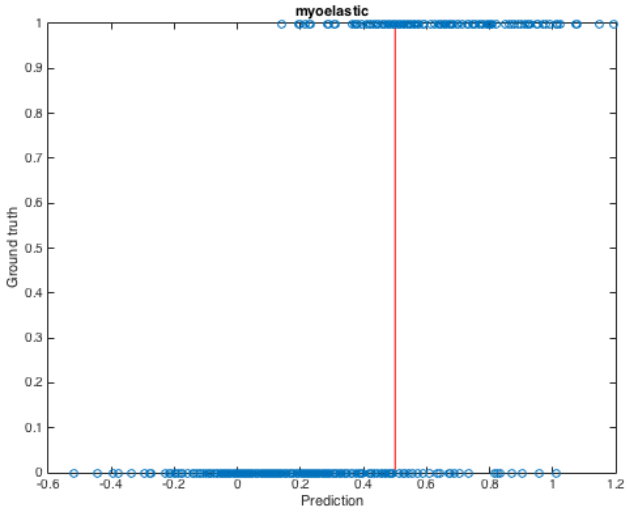

Fig. 28. Classification results for the myoelastic parameter using a sampling rate of 12 kHz.

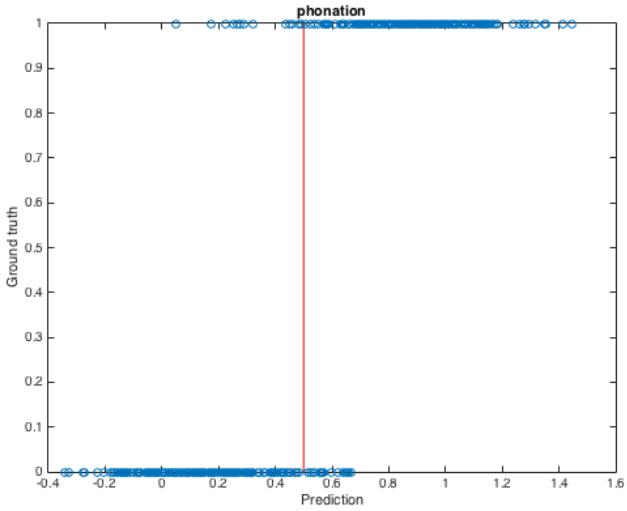

Fig. 29. Classification results for the phonation parameter using a sampling rate of 12 kHz.

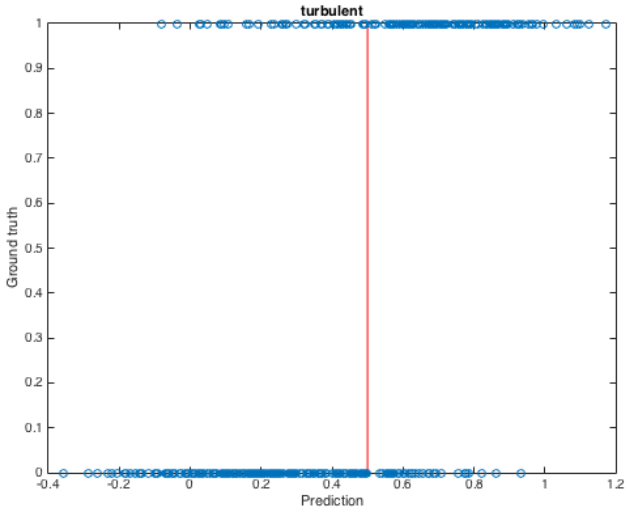

Fig. 30. Classification results for the turbulent parameter using a sampling rate of 12 kHz.

5 Discussion

While the functionality and results of the algorithm are discussed above, its problems are eventually put forward in this section, all the while additionally proposing improvements concerning accuracy and speed. Note that, even though several such suggestions are made, their implementation is beyond the scope of this thesis, and will thus be subject of future work.

5.1 Problems

In particular, an issue introduced in Section 2.4 is addressed first, namely the simultaneous appearance of two or more parameters in one signal. Even though the algorithm is able to correctly distinguish between overlapping periodic (i.e. myoelastic, phonation) and fricative (i.e. turbulent) source types, the concurrent emergence of both of the former might result in some fragments' erroneous classification, especially if one parameter's frequency is a multiple of the other (e.g. 50 Hz for myoelastic and 100 Hz for phonation). Hence, the algorithm is, in its current form, unable to achieve perfect accuracy for these parameters.

Moreover, turbulent signals have been observed to inherently include too much noise for flawless classification. Unfortunately, due to the very nature of the source type in question, which is produced by channelling an airstream through a constriction in the glottis or the vocal tract, a certain kind of background noise is inevitably contained in every such imitation. Perhaps not surprisingly, this results in the lowest classification accuracies of any parameter.

Besides technical problems, imbalances in the selection of input signals are an issue worth mentioning as well. When referencing Tables 7-9 as well as Table 12, it becomes evident that the respective amounts of positive and negative samples are far from equal, with the latter outnumbering the former in every scenario. This inequality is particularly striking for the myoelastic type, which exhibits a total of 232 positive opposed to 894 negative fragments for all speakers. Likewise, when separately examining its 456 eligible and 670 ineligible fragments (cf. Section 3.5), respective distributions of 149 positive and 307 negative as well as 83 positive and 587 negative imitations support this observation.

In addition thereto, examination of Tables 21-23 reveals fairly diverse classification accuracies for individual speakers. For example, in all cases but one, female speakers exhibit better classification accuracies than their male counterparts. Moreover, upon separate consideration of both genders, it becomes obvious that the second male imitator outperforms the first in each case, whereas for the female imitators no such clear distinction can be made. Interestingly enough, the sole perfect classification result achieved by the algorithm at hand is derived when applying PLS without cross-validation to the second male speakers's phonation predictions.

5.2 Improvements

Perhaps not surprisingly, the program's classification accuracies can only be improved when attending to the problems specified in the previous subsection. In particular, all fragments need to be analysed more closely before computing their respective features. The goal of this analysis is thereby three-fold: (i) identify overlapping myoelastic and phonation source types and clearly discriminate between them, (ii) filter out noise from turbulent parameters in order to facilitate prediction thereof, (iii) reject training samples not deemed adequate for accurate classification. Implementation of these measures is believed to result in the creation of more well-tuned prediction models, which in turn can successfully be employed to an arbitrary set of real-world data.

However, not only the algorithm's accuracy can be improved by means of follow-up efforts. In fact, speed of execution is an even more pressing issue in the program at hand. As mentioned in Section 4, the latter's default input variables are chosen in order to yield the best results for non-downsampled signals. Naturally, this decision leads to rather lengthly runtimes. Since good time performance is nonetheless essential in real-world scenarios, acceleration of the program's execution ultimately becomes a necessity, thereby condoning slightly lower classification accuracies. With this paradigm in mind, the following changes are proposed for the creation of a light-weight and fast version of the algorithm:

1. **Prediction method**: use solely PLS with 10-fold cross-validation as well as PLS without cross-validation
2. **PLS components**: lower the number of PLS components to 3 for each parameter
3. **Downsampling factor**: choose a downsampling factor of 4 (i.e. a new sampling rate of 12 kHz)

Subsequently, the accuracies of this alternative classification are shown in Tables 36-38, with the prediction values of PLS without cross-validation depicted in Figures 31-33. All the while, the number of speakers and minimum fragment length remain unchanged. Not surprisingly, the algorithm is indeed executed much more quickly. Conversely, the corresponding classification accuracies deteriorate only slightly (< 2 %) in comparison with the results presented in Section 4.1, thus highlighting the potential of this novel approach regarding runtime improvements.

Table 36. Alternative overall classification accuracies for the myoelastic parameter.

PLS components = 3, F_s = 12 kHz	**PLS**
10-fold cross-validation	80.22 %
no cross-validation	85.96 %

Table 37. Alternative overall classification accuracies for the phonation parameter.

PLS components = 3, F_s = 12 kHz	**PLS**
10-fold cross-validation	85.82 %
no cross-validation	93.12 %

Table 38. Alternative overall classification accuracies for the turbulent parameter.

PLS components = 3, F_s = 12 kHz	**PLS**
10-fold cross-validation	75.46 %
no cross-validation	80.04 %

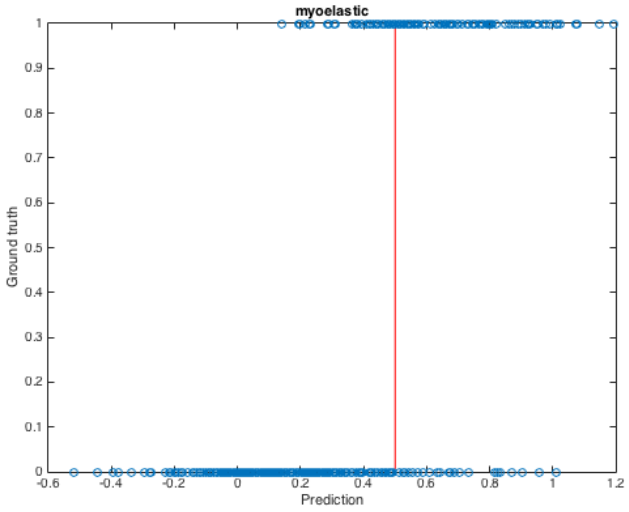

Fig. 31. Alternative overall classification results for the myoelastic parameter.

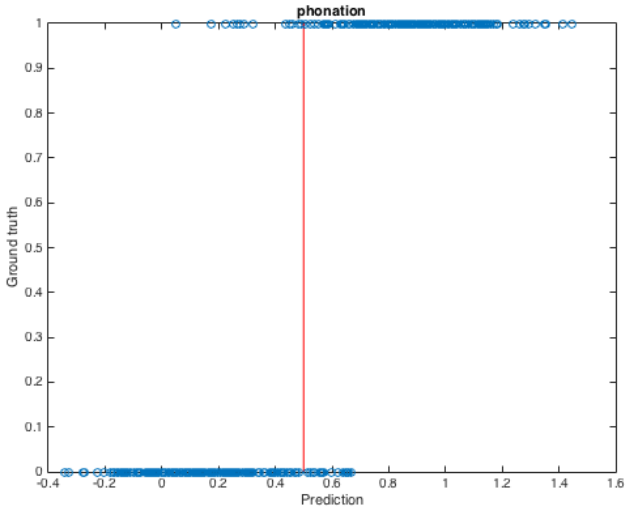

Fig. 32. Alternative overall classification results for the phonation parameter.

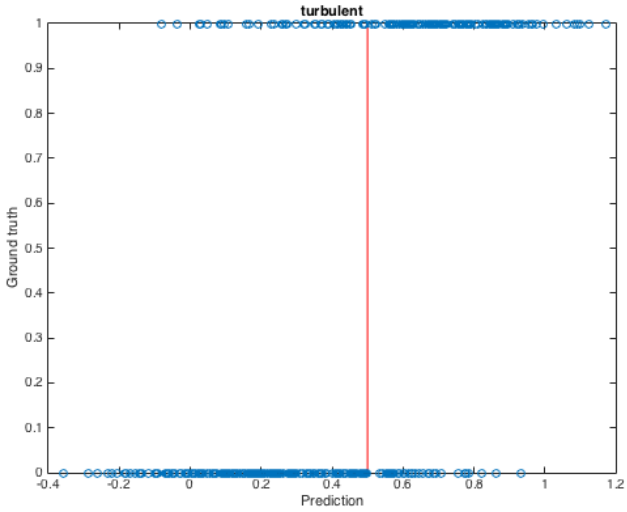

Fig. 33. Alternative overall classification results for the turbulent parameter.

6 Summary

The Master's thesis at hand dealt with the automatic extraction of articulatory parameters from audio files of everyday sounds (e.g. animal sounds, mechanical sounds) vocally imitated by actors. Thus, in order to gather the necessary information, the student joined the team at KTH Stockholm in the wake of his research, with the intent of crafting this work as well as contributing to the EU project SkAT-VG. As mentioned above, the overall goal of the latter was three-fold: (i) improve the understanding of how sounds are communicated through vocalizations and gestures, (ii) look for physical relations between vocal sounds and sound-producing phenomena, (iii) design tools for converting vocalizations and gestures into parametrised sound models [14]. Both the team at KTH and the student concentrated their efforts on the first subgoal, which will additionally be summarized in a separate paper at a later time.

In particular, the thesis was structured as follows: Section 1 introduced the general idea and included an outlook on consecutive chapters. Subsequently, Section 2 offered an overview of the subject matter's background, thereby focusing on the following topics:

1. Vocal imitations and verbal descriptions
2. Classification frameworks
3. Sound analysis and feature extraction
4. Sound sketching parameters

Naturally, special attention was given to academic work carried out as part of aforementioned research project, in addition to referencing other relevant publications. Moreover, Section 3 explained and discussed the program's methodology in more detail, all the while concentrating on a number of different notions:

1. Audio files
2. Annotations
3. Fragments
4. Auto-correlation
5. Features
6. Classification

The algorithm's execution was thereby outlined as follows: at first, the imitations of everyday sounds by four voice artists, recorded at KTH before the arrival of the student, were annotated by the department's phonetic experts with regard to three speech parameters introduced by Helgason [7], namely *myoelastic*, *phonation* and *turbulent*. Subsequently, these annotations were applied to the raw data, resulting in the extraction of a large number of short audio files (i.e. fragments), which were either marked as positive or negative, depending on their possession of the respective humanly annotated parameter or the lack thereof.

Thereafter, a set of features was created by means of the auto-correlation function introduced by Cheveigné and Kawahara [2]. Subsequently applying these features to aforementioned fragments, the latter were classified into a positive and a negative group, all the while using two types of prediction models, namely PLS regression and SVM, to forecast the corresponding results. To ensure flexibility and applicability in real-world scenarios, N-fold and leave-one-out cross-validation were additionally performed on the data sets.

Eventually, the prediction results were matched against the annotated ground truth in order yield the classification accuracies discussed in Section 4. In addition to its default combination, four different instances of the algorithms' input variables were examined:

1. PLS components
2. Speakers
3. Minimum fragment length
4. Downsampling factor

Finally, problems with the algorithm were discussed in Section 5, all the while offering improvements in order to further enhance the accuracy and speed of the program by means of future efforts.

In general, the overall results of the classification process are satisfactory, revealing good accuracies of around 82 %, 88 % and 76 % for the myoelastic, phonation and turbulent parameters, respectively. Upon comparison with the corresponding success rates of around 86 %, 94 % and 80 % derived by Friberg and his team [5] by means of the analysis of the fragments' spectrograms, it becomes evident that a promising time domain-only approach to some kinds of speech processing is put forward in the work at hand. Thus, the student believes that this Master's thesis successfully contributes to the academic research by investigating and exploiting the potential of vocal imitations, thereby facilitating the process of producing, verifying, selecting, communicating and refining ideas for sonic designers and sound artists alike.

References

1. David S. Blancas and J. Janer (2014). Sound retrieval from voice imitation queries in collaborative databases. In *Proceedings of the AES 53rd Conference on Semantic Audio*, 1-6. London, UK.
2. Alain de Cheveigné and H. Kawahara (2002). YIN, a fundamental frequency estimator for speech and music. *The Journal of the Acoustical Society of America* 111 (4), 1917-1930.
3. Mark Cartwright and B. Pardo (2015). VocalSketch: Vocally Imitating Audio Concepts. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, 43-46. Seoul, Republic of Korea.
4. Arnaud Dessein and G. Lemaitre (2009). Free classification of vocal imitations of everyday sounds. In *Proceedings of the 6th Sound and Music Computing Conference*, 213-218. Porto, Portugal.
5. Anders Friberg, T. Lindeberg, P. Helgason, M. Hellwagner, G. Laís Salomão and S. Ternström (2015). Prediction of three articulatory classes from audio. *Work Package Report*, EU project SkAT-VG (Sketching Audio Technologies using Vocalization and Gestures). Stockholm, Sweden.
6. Paul Geladi and B. R. Kowalski (1986). Partial Least Squares Regression: A Tutorial. *Analytica Chimica Acta* 185, 1-17.
7. Pétur Helgason (2014). Sound initiation and source types in human imitations of sounds. In *Proceedings from FONETIK 2014*, 83-88. Stockholm, Sweden.
8. Pétur Helgason (2015). Preliminary annotation of the database of imitations of action primitives in terms of vocal primitives. *Public Deliverable*, EU project SkAT-VG (Sketching Audio Technologies using Vocalization and Gestures). Stockholm, Sweden.
9. Soonil Kwon and L.-H. Kim (2011). Sound sketching via voice. In *Proceedings of the 5th International Conference on Ubiquitous Information Management and Communication*, Article No. 48. Seoul, Republic of Korea.
10. Guillaume Lemaitre, A. Dessein, P. Susini and K. Aura (2011). Vocal imitations and the identification of sound events. *Ecological Psychology* 23 (4), 267-307.
11. Guillaume Lemaitre and D. Rocchesso (2014). On the effectiveness of vocal imitations and verbal descriptions of sounds. *The Journal of the Acoustical Society of America* 135 (2), 862-873.
12. Guillaume Lemaitre, P. Susini, D. Rocchesso, C. Lambourg and P. Boussard (2014). Non-Verbal Imitations as a Sketching Tool for Sound Design. In *Sound, Music, and Motion*, 558-574. Springer International Publishing. Cham, Switzerland.
13. Olivier Lartillot and P. Toiviainen (2007). A MATLAB toolbox for musical feature extraction from audio. In *Proceedings of the 10th International Conference on Digital Audio Effects*, 237-244. Bordeaux, France.
14. Stefano Delle Monache, S. Baldan, D. A. Mauro and D. Rocchesso (2014). A design exploration on the effectiveness of vocal imitations. In *Proceedings of the 40th International Computer Music Conference and the 11th Sound and Music Computing Conference*, 1642-1648. Athens, Greece.
15. Geoffroy Peeters (2004). A large set of audio features for sound description (similarity and classification) in the CUIDADO project. *Technical report*, IRCAM. Paris, France.

16. Geoffroy Peeters, B. L. Giordano, P. Susini, N. Misdariis and S. McAdams (2011). The Timbre Toolbox: Extracting audio descriptors from musical signals. *The Journal of the Acoustical Society of America* 130 (5), 2902-2916.
17. Vishweshwara M. Rao (2011). Vocal Melody Extraction from Polyphonic Audio with Pitched Accompaniment. *PhD thesis*. Mumbai, India.
18. Davide Rocchesso, G. Lemaitre, P. Susini, S. Ternström and P. Boussard (2015). Sketching Sound with Voice and Gesture. *Interactions* 22 (1), 38-41.
19. Davide Rocchesso and D. A. Mauro (2014). Self-organizing the space of vocal imitations. In *Proceedings of XX CIM Conference*, 124-127. Rome, Italy.
20. Davide Rocchesso (2015). *Publishable summary*, EU project SkAT-VG (Sketching Audio Technologies using Vocalization and Gestures). Venice, Italy.
21. Sten Ternström (2015). Extensive set of recorded imitations. *Public Deliverable*, EU project SkAT-VG (Sketching Audio Technologies using Vocalization and Gestures). Stockholm, Sweden.
22. Randall D. Tobias (1995). An Introduction to Partial Least Squares Regression. In *Proceedings of the 20th Annual SAS Users Group International Conference*, 2-5. Orlando, FL.
23. Hugues Vinet et al. (2002). The CUIDADO Project. In *Proceedings of the 3rd International Conference on Music Information Retrieval*, 197-203. Paris, France.
24. Stefan Weinzierl (2008). Handbuch der Audiotechnik. *Book*, Springer-Verlag Berlin Heidelberg. Heidelberg, Germany.
25. Udo Zölzer (2011). DAFX: Digital Audio Effects, Second Edition. *Book*, John Wiley & Sons. Hoboken, NJ.