TROMPA

TROMPA: Towards Richer Online Music Public-domain Archives

Deliverable 3.3

Audio Processing

Grant Agreement nr	770376
Project runtime	May 2018 - April 2021
Document Reference	TR-D3.3-Audio Processing v1
Work Package	WP3 - Automated Music Data Processing and Linking
Deliverable Type	Report
Dissemination Level	CO-Consortium
Document due date	28 February 2019
Date of submission	28 February 2019
Leader	VL
Contact Person	Álvaro Sarasúa (alvaro.sarasua@voctrolabs.com)
Authors	Álvaro Sarasúa (VL), Jordi Janer (VL), Pritish Chandna (UPF)
Reviewers	Emilia Gomez (UPF), Aggelos Gkiokas (UPF)

Executive Summary

This document presents the work performed so far for Task 3.3, *Audio processing*, within Work Package 3, as well as the plans for future work within the project. In this Task, we will research and develop techniques for audio synthesis, using existing synthesizers and devoting efforts to provide natural singing voice synthesis for the choir pilot study with multi-lingual support. In addition, we will consider sound emphasis techniques, based on sound source separation, so that users can focus their attention to different voices of the ensemble, orchestra, or choir.

This work is done meeting the requirements defined in WP2 and detailed in Deliverable 2.1, *Early Requirements*, regarding the integration of these technologies in the Choir Singers Pilot to be developed in WP6, Task 6.6. In addition, this integration will be done considering the *Data Infrastructure* model defined in Deliverable 5.1 from WP5. We also identify and consider additional possibilities for exploitation to be considered in WP7, *Dissemination and Exploitation*.

The main goals for this task are in this context to improve the state-of-the-art technologies for choir singing synthesis with multi-lingual support and to make them available for integration in the aforementioned Choir Singers Pilot. This involves recording new datasets, which will be made available for the consortium. At the same time, existing tools such as Voctro Labs' *Voiceful Editor,* which support the whole process (including data annotation) will be further developed during the project.

The challenge in singing synthesis is to go from a symbolic score to an audio signal. Scores are symbolic representations of a musical piece, and performers introduce micro-deviations in timing and intonation that a synthesis engine should be able to model. The case of choir singing adds additional requirements, since these micro-deviations also occur among different singers in the choir. Finally, multi-lingual support also adds specific requirements, since each language has its own phonetic and linguistic rules that determine how native speakers pronounce it. Because of this, speech and singing synthesis systems are language dependent and therefore a singing synthesis system needs to know the language of a piece to correctly convert the lyrics written text to the phonetic sounds of that particular language.

The singing synthesis algorithm we use in TROMPA is based on on the system called Neural Parametric Singing Synthesis (NPSS). This model is based on a modified version of the WaveNet architecture. As opposed to modeling the raw waveform, we model features produced by a parametric vocoder, separating the influence of pitch and timbre. The current approach has some limitations in terms of voice quality and realism. For example, in some cases the timbre model produce low variations in terms of dynamics, which results on a static and dull sound. Also, the vocoder output produces an artificial buzz sound on sustained harmonic sounds due to its internal algorithm design. Finally, the pitch model requires consistent data (e.g. melodies recorded in the same singing style) and it does not generalize well when the synthesis score contains melodies that differ too much from the ones in the training set. We plan to overcome these limitations taking benefit from recent advances in text-to-speech synthesis.

As for extending current models to the choir case, we identify several strategies. The first possibility is to use voice transformation to generate multiple voices from a same voice model. A similar strategy (in the sense of creating multiple different voices from the same synthesis model) is to use voice cloning. Here, the approach consists on creating several voices using by using a small amount of data by new soloist singers. Finally, there is an alternative approach base on solo-to-choir rendering. This can be achieved using spectral morphing (although this has limitations for unvoiced

parts) or using a solo-to-choir Wavenet-based vocoder. In this case, the input would be the vocoder features corresponding to a monophonic solo voice of a choir section, and the output would be a waveform of the choir sound.

We have a number of existing tools that will be used and further developed in this task. Voctro Labs' current synthesis engine, *VoSyn*, works both for speech and singing and it is integrable in a wide range of platforms. The *Voiceful Editor* desktop application is currently used for a wide range of internal tasks including data annotation, and it will be adapted and made available for the consortium for such tasks. In addition, other libraries such as *VoMix* (which works as a virtual DAW with all standard audio effects) can and will be used to apply sound effects to the rendered voices.

There are a number of existing datasets that we can use within the TROMPA project for singing synthesis, including solo performances by pop singers in English, a soprano singer in Italian, as well as choir recordings in Catalan, Spanish and German. While these are useful for the initial steps in this task, it is necessary to perform new recordings in order to cover more languages, as well as to improve the quality of the synthesizers (since this quality is directly dependant on the amount of data with which the models are trained).

All these technologies will be integrated in the Choir Singers Pilot by the end of the project. For this, we will use the Voiceful Cloud Service, which runs Voctro Labs' algorithms in the cloud providing access through a RESTful API. This integration will be explained in detail in Deliverable 5.3 - TROMPA Processing Library.

Beyond the Pilot, we identified some other possibilities for exploitation of the technologies developed in this Task. There are existing commercial applications that provide similar functionality to the one that the Choir Singers Pilot provide, but they do not integrate singing synthesis, which dramatically reduces the possibilities for extending the catalogue. Also, the synthesis algorithms developed in this task can be integrated in plug-ins for Digital Audio Workstations (DAW) so that music producers and composers can get realistic choir singing.

Version Log		
#	Date	Description
v0.1	16 February 2019	Initial version submitted for internal review
v0.2	22 February 2019	Revised version after internal review
v1.0	28 February 2019	Final version submitted to EC

Table of Contents

1 Introduction

1.1 Structure of the report

This deliverable presents the work performed so far for Task 3.3, Audio processing, as well as the plans for future work within the project. This Task deals with developing technologies for audio synthesis, using existing synthesizers and developing techniques to provide natural singing voice synthesis for the choir pilot study. In addition, this Task deals with sound emphasis techniques based on source separation of choral parts from audio recordings.

This document is structured as follows:

- This Section (1, Introduction) introduces the goals of the task and the state of the art in singing synthesis.
- Section 2, *Audio processing techniques*, explains the current audio processing techniques that are available for singing synthesis and details the strategies to meet the goals for the multilingual choir singing case.
- Section 3, *Audio Processing tools*, presents the tools currently available for audio processing, as well as other tools that can support necessary tasks such as data annotation and that will be used and extended during the project.
- Section 4, *Data resources and tools*, explains the process for gathering and annotating datasets for training choir singing synthesis models, details the existing available datasets and introduces the goals for future datasets to be created during the project.
- Section 5, *Integration with Choir Singers Pilot and further exploitation*, briefly discusses how the technologies developed in this Task will be integrated into the main tangible outcome of the project to be developed in Work Package 6, Task 6.5: the Choir Singers Pilot. It also discusses some plans for further exploitation of the technologies to be developed in this Task.
- Section 6 contains the conclusions of this report.

1.2 Task goals

This synthesis technology developed in this task will be integrated into the Choir Singers Pilot (CSP). The goal of this pilot is to assist amateur choir singers during individual performance. Users should be able to synthesize existing scores, and to sing-along with the synthesized voices.

In this context, the goals of this Task are oriented to meet the functional requirements of the pilot detailed in Deliverable 2.1-Early Requirements[1].

More concretely, the **main goals** of this Task are:

- Research and develop technologies for choir singing synthesis that improve state-of-the-art singing engines (for the case of choir) with multi-lingual support (in priority level Spanish, Catalan, English, Latin, German).
- Make the synthesis engine available as Software as a Service (SaaS) that can be accessed from other applications, supporting choral synthesis for scores in Music Encoding Initiative (MEI)[2] [31] and MusicXML[3] [36] formats.

[1] https://trompamusic.eu/deliverables/TR-D2.1-Early_Requirements.pdf
[2] https://music-encoding.org/

- Record, annotate, and make accessible for the consortium the necessary datasets for training the models.

Pursuing these objectives implies a set of necessary tasks that will also result in useful tools that will be delivered during and at the end of the project. At the same time, we will identify additional exploitation plans for the singing synthesis technologies to be developed. This will allow to shape and refine the technical requirements of these technologies in order to make them available for other possible exploitation opportunities beyond the CSP. In summary, these are the **subgoals** of this Task that derive from this:

- Develop the necessary tools for assisting the dataset creation and annotation process. More concretely, we will adapt the existing *Voiceful Editor* application (introduced in Section 3.2) to assist this process.
- Identify the requirements for adapting the technologies to other exploitation possibilities, minimizing the cost of adaptation at the end of the project.

1.3 State of the art

This section provides the technical background in the topic of singing voice synthesis. We describe the main techniques as well as the challenges faced in singing compared to speech. Finally we detail the particularities of choir singing synthesis.

1.3.1 Background on speech and singing voice synthesis

Computer-generated voices has been a topic of interest for scientists and engineers for decades. The goal is to artificially generate a speech signal, and the range of applications is vast. Today we see as a commodity voice assistants on mobile phones and other household devices.

There are differences between speech and singing voice signals, which need to be taken into account if we want to generate an artificial singing voice.

- Pitch height: in speech we tend to speak with a lower tone, to avoid fatigue of the vocal cords.
- Pitch range: in speech signals is also smaller than in singing, where a singer can have a pitch range of 2 octaves.
- Timbre: the way we arrange the different elements of the vocal tract (larynx height, mouth opening, etc.) is different when we sing and when we speak. This determines the timbre qualities of the sound.
- Timings: in speech the rate of the duration of vowels and consonant phonemes is balanced, while in singing, the duration of the musical notes determines the duration of the vowel. Therefore the ratio vowel/consonant is larger in singing than in speech.
- Linguistic/Musical information: speech signals encode linguistic information (along with expressive components), and singing signals encode both linguistic (lyrics) and musical information (score notes).

[3] https://www.musicxml.com/

It is not the aim of this report to provide a comprehensive review of Speech Synthesis methods. However, it is worth mention the evolution of state-of-the-art techniques in the recent years, and the evolution in quality that we can achieve today.

Traditionally, we made a distinction between parametric and unit-selection speech synthesis system. Parametric systems (e.g. based on Hidden Markov Models (HMM) [20] used a speech datasets to train a predictive model that infer some vocoder parameters. The vocoder converted these parameters to the actual waveform. The quality was limited by the prediction of the parameters, which tend to smooth spectral resolution, and the quality of the vocoder output.

Unit selection systems, relied also on speech datasets but in this case given an input text it concatenated samples from the dataset to render the output sound. The larger the dataset, and the longer the sentences it contained, the better quality it could be achieved. Evolution of unit-selection or sample-based synthesis engines, combined the sample selection with spectral transformations obtaining more quality and flexibility.

Modern techniques today are based on Machine Learning, and use Deep Neural Networks to infer either vocoder parameters, or directly waveform samples as in the original WaveNet algorithm [15].

Although singing voice synthesis technology is available since the advent of computer-generated speech, the human voice is still one of the most difficult instruments to model. Choir singing presents particular additional difficulties due to the strong modulations present in the audio and the lack of annotated datasets for training models. In addition, choir repertoires usually include pieces in several languages, which adds an additional requirement.

1.3.2 Challenges towards a realistic artificial singing

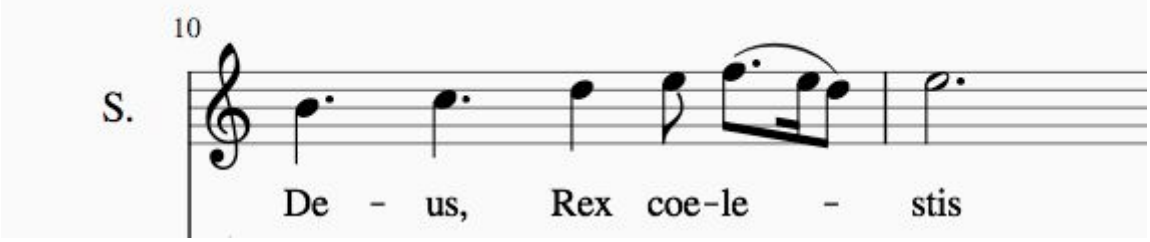
1.3.2.1 From a symbolic score to an audio signal

One of the main challenges when generating a synthetic singing from a score, is to model the micro-deviations we find in a real singer performance. Musical scores are symbolic representations of a musical piece, which follow some conventions in how the musical elements are encoded. For example, the score encodes global information of the piece (e.g. the tempo), as well as information of individual notes (e.g. duration and pitch) or characteristics affecting a musical phrase (e.g. the dynamics). The score representation is a simplified version of the music it describes.

When comparing a real singing performance recording to the information in the music score, we observe that the main differences will be the timing and pitch deviations. In the musical score, the duration of the notes is quantized and determined by the tempo (beats per minute), and the note figure (e.g. quarter note, eighth note, etc.). In a real performance, the start time and duration of the notes is not perfectly synchronized to a fixed tempo, since the singer will start a note slightly before or after the exact time, sometimes following the indications of a conductor during a live performance. These small deviations are what give the human-character and increase the expressivity of a performance.

The same type of difference can be observed for the instantaneous evolution of the pitch over time, related to the fundamental frequency value (in Hertz) of the produced voice signal. The singing voice, compared to other musical instruments with a discrete pitch (e.g. piano) is able to generate a continuous varying pitch between two semitones, e.g. between B3 and C4. As shown in Figure 1.1.a below, we see that the musical score contains eight notes. If we look at the pitch curve analyzed

from a real singing performance of this score, we see in figure 1.1.b that the purple curve (pitch) is a continuous curve, we it is difficult to identify the eight notes.



(a)

(b)

Figure 1.1. Musical score (a) and analyzed singing performance (b) of the piece. The purple curve indicates the pitch computed from the singing voice audio. Snapshot from Voiceful Editor.

1.3.2.2 Adding human interaction in the loop

Most of singing voice synthesis systems available today are meant to be used in the context of a Digital Audio Workstation (DAW) production. It means that there is a human in the loop, who provides additional manual editing to refine the timing and pitch micro-deviations of the symbolic score. The music producer is able to adjust the synthesis parameters in order to get the artistically desired result.

In terms of user interaction, these adjustments consists typically in editing a MIDI score on a GUI, in the fashion of traditional MIDI sequencers (e.g. Cubase). It may involve basic actions such as adjusting the length and pitch of individual notes; to more accurate operations such as adjusting the vibrato parameters (rate, and depth), modifying the pronounced phonemes, changing the timbre qualities or dynamics over time, etc. One example of this manual editing can be the popular commercial Vocaloid® software by Yamaha Corp. (Japan) [37], or more recently on an independent

project V-Synth by Kanru Hua [8]. The same type of GUI is implemented in the Voiceful Editor developed by Voctro Labs and used in the Choir Singers Pilot (see Section 3.1).

But, also we can employ the human interaction in form of an input audio signal, and combine a real recording and symbolic information to improve the realism in the synthesized output. This is the case of an approach that we called performance-driven synthesis [9]. In this approach we extract the musical controls (pitch curve, phonetic timings and dynamics) from a real performance, and generate more accurate symbolic controls for the synthesis engine.

One interesting trend in the area of audio processing is the fusion between audio processing at signal level and at symbolic level. When the input is an audio signal, we depart from the signal domain to the symbolic domain to do some *semantic* manipulation on the original audio signal. We find for example audio software such as Melodyne® [24], in which the user operates on a symbolic representation of *objects* (notes) generated by analyzing an audio signal (e.g. a recorded vocal track). The operations at the symbolic level are applied then to the actual recording at a signal level (audio signal processing). This type of tools have been extensively used in vocal production by studio engineers for correcting a real singing performance, adjusting pitch and timings at note level, applying effects such autotuning or harmonization on the recorded signal. There are conferences such as AES Semantic Audio[4] that have covered this topic [23]. And researchers at UPF worked in this topic in the past, targeting to polyphonic music signals such as piano recordings. In this case, the approach combined the sound transformation with source separation techniques [10].

A similar approach for the case of spoken dialogues is found in the VoCo system by Adobe® [11], where an input dialogue recording can be manipulated at symbolic level, replacing the words or phonemes and the system renders the modified audio signal combining speech recognition with synthesis and voice conversion techniques.

1.3.2.3 Particularities of choir singing synthesis

For the specific case of choir singing, we face also specific challenges. Choir singing differs from solo singing in a number of acoustic characteristics.

First of all, each of the singers in the choir has unique timbre characteristics. This means that a choir cannot be modeled using the same synthesis model to generate several voices performing the same melody. Alternatively, it becomes necessary to train different models or to apply transformations to each rendered voice in order to create different timbres for each voices.

There are other acoustic characteristics that are unique to each singer in the choir. For example, choir singers singing the same note (at unison) do not produce the exact same fundamental frequency, f_0. The small deviations in f_0 between singers producing the same note at unison performance is known as fundamental frequency dispersion [19]. This dispersion between singers affects the timbre of the resulting sound, which means that it is not possible to model a choir section by just synthesizing several singers producing the exact same pitch. Similar effects can be observed for vibrato (not all singers producing the same vibrato) and timing (not all singers starting and ending notes at the exact same time) [7]. If, as we have seen for solo singing, the challenge to produce human-like is to model the microdeviations from the score introduced by the performer, the challenge in choir singing is to model the microdeviations between different performers in the choir.

[4] http://www.aes.org/conferences/2017/semantic/

Alternatively, as we will see in Section 2.2.2, the whole choir section can be considered as the sound source to be modeled. This means, in any case, that specific recordings containing each of the sections separately must be performed in order to train new models.

1.3.3 Multilingual synthesis

In the scope of TROMPA, the goal is to sonify scores in multiple languages. Each language has its own phonetic and linguistic rules that determine how native speakers pronounce it.

Speech and singing synthesis systems are language dependent. Therefore a singing synthesis system needs to know the language of a piece to correctly convert the lyrics written text to the phonetic sounds of that particular language.

With modern synthesis techniques based on machine learning, it means that we need multilingual datasets to train multilingual models. At algorithm level, traditional synthesis systems (e.g. HMM/DNN-based Speech Synthesis System[5] (HTS) [28]) had both acoustic and linguistic models, where the latter had to be adapted specifically for each language. These linguistic components might include lexicons, linguistic rules that generate detailed linguistic information about the input text to better condition the resulting synthesis speech (e.g. labelling words with its grammar function, phonetic transcription, including exceptions, etc.). There are open source frameworks such as the Festival Speech Synthesis System[6] [32] that provide datasets and tools.

Some recent text-to-speech (TTS) research [21], took a different approach and directly use end-to-end data modelling. In this case, the term *end-to-end* means that the input of the deep neural network in the text in orthographic form, and the output is the audio signal. These recent end-to-end approaches reduce the work of developing specific linguistic adaptations for each new language, and makes systems more scalable and generalizable.

Selecting the supported languages is important in the scope of the TROMPA project and the Choir Singers Pilot (CSP) in particular. It will determine the repertoire that we will be able to use in the CSP and indirectly the target communities that can sing in those languages. In the original Description of Work document, we already introduced the need to provide a multilingual system.

Given the type of repertoire (choral music) and the geographical location (Barcelona) of the partners working on the Choir Singers Pilot, we will be targeting first Catalan and Spanish (including Latin), and work towards incorporating English and German to the system by the end of the project.

1.4 Workflow for creating a synthetic singing voice

Modern techniques for singing synthesis are based on supervised machine learning with deep neural networks (Deep Learning). These techniques require annotated datasets to train the models. This means that the data gathering and annotation are part of the process of creating a new synthetic sinting voice.

Figure 1.2 depicts this sequence at a very top-level. Generating a synthetic singing voice with modern techniques consists in:

- collecting data (singer recordings),
- annotating the data (with text and musical information), we need to model singing components: timbre, pitch and phonetic timings.

[5] http://hts.sp.nitech.ac.jp/
[6] http://www.cstr.ed.ac.uk/projects/festival/

- learning data models (with machine learning algorithms). Modern techniques based on supervised machine learning (Deep learning), which require annotated datasets
- integrating it in usable software components (APIs, libraries or applications). It includes languages supported and available software components.

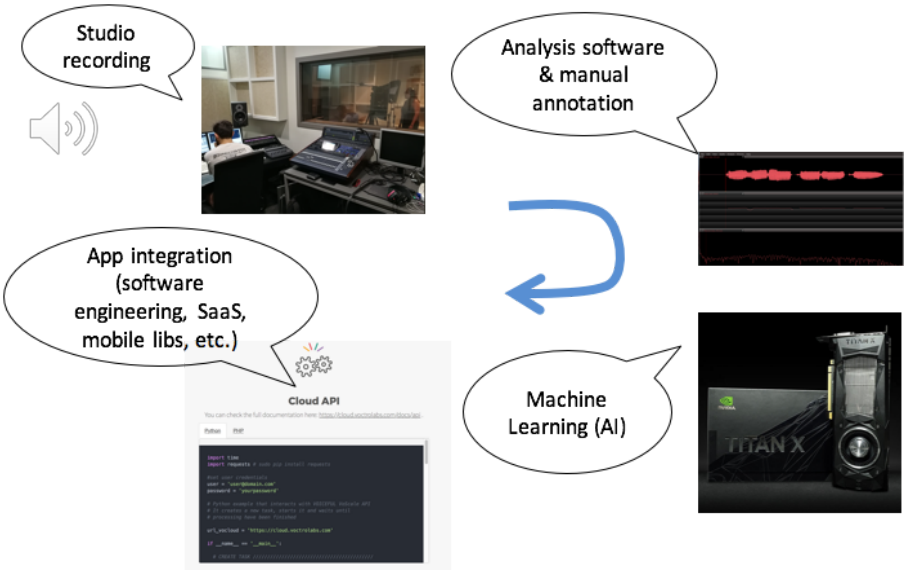


Figure 1.2. Workflow for creating a synthetic singing voice.

2 Audio Processing techniques

This section described the audio processing algorithms and research work related to the task T3.3 Audio Processing. We describe first in section 2.1 the current singing synthesis engine that will be used in TROMPA. Next we address in Section 2.2 and 2.3 the novel contributions towards generating a convincing synthetic choir singing. In Section 2.4 we describe the Audio Source Separation techniques used for separating recordings of choir singing.

2.1 Current singing synthesis algorithm (NPSS)

The singing synthesis algorithm we use in TROMPA is based on the system called Neural Parametric Singing Synthesis (NPSS), as described in the original paper by M. Blaauw and J. Bonada [3]. This model is based on a modified version of the WaveNet [15] architecture. As opposed to modeling the raw waveform, we model features produced by a parametric vocoder, separating the influence of pitch and timbre. The advantage of this approach is that it allows to conveniently modify pitch to match any target melody, it facilitates training on smaller data sizes, and it reduces training and generation times.

2.1.1 Modelling Timbre, Pitch & phonetic timing

The baseline algorithm maps time-aligned control features, $\mathbf{c} = \{c1, c2, \ldots, cT\}$, to acoustic vocoder features, $\mathbf{x} = \{x1, x2, \ldots, xT\}$, using an autoregressive model. The vocoder features can be converted to the output audio waveform for example using the WORLD vocoder[7] [12].

Figure 2.1 shows the system overview with the three modelling components: phonetic timing, pitch and timbre. These components are briefly described in the following sections as well, taking the contents from the original paper [3] written by UPF researchers.

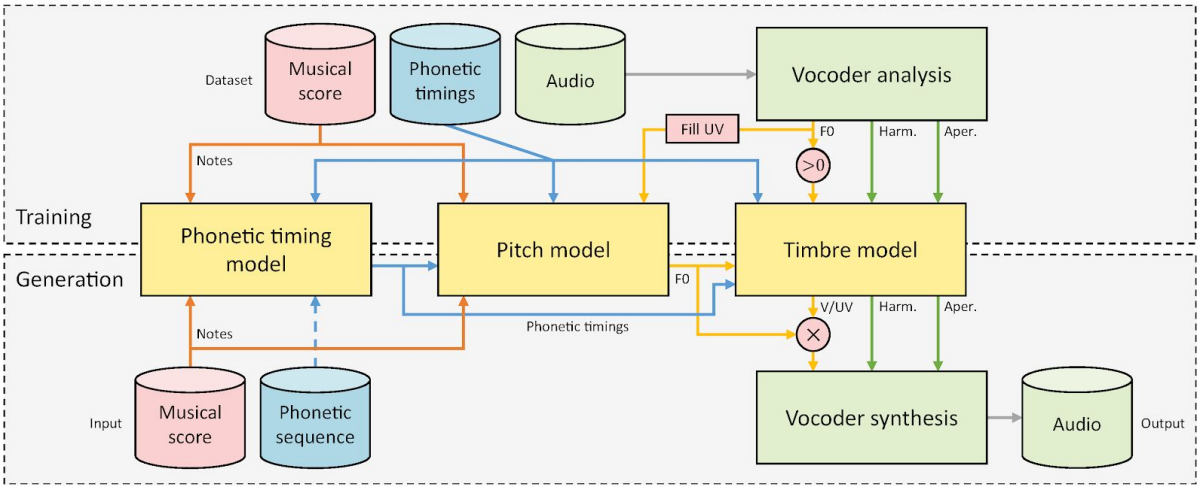


Fig 2.1: Overview of the system, depicting the different components. Figure reproduced with permission from the original paper [3].

2.1.1.1 Timbre model

This model is responsible for generating acoustic features related to the timbre of the voice. Control inputs are the sequence of timed phonemes and f_0, predicted by the timing model and pitch model respectively. The predicted timbrical acoustic features can be combined with the predicted f_0 to generate the final waveform using the synthesis stage of the vocoder.

2.1.1.2 Pitch Model

Generating expressive f_0 contours for singing voice is quite challenging. Not only is this because of its importance to the overall results, but also because in singing voice there are many factors that simultaneously affect f_0. There are a number of musical factors, including melody, various types of attacks, releases and transitions, phrasing, vibratos, and so on. Additionally, phonetics can also cause inflections in f_0, so-called microprosody. Some approaches try to decompose these factors to various degrees, for instance by separating vibratos or using source material without consonants. In our approach, however, we model the f_0 contour as-is, without any decomposition. As such, f_0 is predicted from both musical and phonetic control inputs, using a modified WaveNet architecture.

[7] https://github.com/mmorise/World

2.1.1.3 Timing model

The timing model is used to predict the duration of each phoneme in the sequence to synthesize. Unlike with TTS systems where phoneme durations are generally predicted in a freerunning manner, without tight constraints regarding when every phoneme must start and end, in singing synthesis, the phoneme durations are heavily constrained by the musical score. In the proposed system we enforce this constraint using a multistep prediction. First, the note timing model predicts the deviations of note (and rest) onsets with respect to nominal onsets in the musical score. At the same time phoneme durations are predicted by the phoneme duration model. Finally, a simple fitting heuristic is used to ensure the predicted phoneme durations fit within the available note duration, after adjusting timing.

2.1.2 Enhancements on Timbre, Pitch & Phonetic timing modelling

The current NPSS approach presents also some limitations in terms of voice quality and realism, understood as similarity to a real human voice recording. In some case we may found that:

- The timbre model produce low variations in terms of dynamics, which results on a static and dull sound.
- The vocoder output produces an artificial buzz sound on sustained harmonic sounds due to its internal algorithm design.
- The pitch model requires of consistent data (e.g. melodies recorded in the same singing style) and it does not generalize well when the synthesis score contains different melodies.

Improvements over current NPSS approach are planned to be developed as part of the Task T3.3. We can benefit from recent approaches in the area of TTS synthesis, and make the necessary adaptations for the case of singing voice and music scores input. For example, one promising approach is using a a recurrent sequence-to-sequence feature prediction network with *attention* to improve the way the neural network uses conditioning information [17]. Another interesting approach is using a fully-convolutional attention-based neural network [16], which provides faster training times.

But, in terms of sound quality and naturalness, one of the main improvements over the original NPSS algorithm described in this section, is using a neural vocoder that directly predicts the waveform from vocoder features [15]. Training a WaveNet vocoder requires significantly a large amount of data. However, only acoustic data is needed, which notably reduces the tedious work of annotating a dataset, and data from multiple speakers and languages can be combined.

2.2 Strategies for modelling choir singing (VL)

The main Research and Innovation contribution regarding the Audio Processing task, is the generation of convincing synthetic choir singing. With the review of the state of the art (Section 1), and the baseline synthesis algorithm explained in previous section, we defined four different strategies to generate choir singing. These strategies can be grouped in two categories:

- Generating multiple individual voices (2.2.1)
- Generating a single solo-to-choir sound (2.2.2)

2.2.1 Multiple individual voices from a soloist

2.2.1.1 Voice transformation approach (spectral transformations)

One possible strategy to model choir singing is to use solo-singing synthesizers to render, for each of the choir sections, several performances (as many as singers we want to model in the section). However, as we mentioned in Section 1.3.2 when discussing the challenges of choir singing synthesis, each of the singers has his/her unique timbric characteristics, and there are micro-deviations in intonation and timing between each of them.

In order to model these microdeviations between different singers, one possibility is to apply different voice transformation effects to each singer. The *VoTrans* voice transformation library developed by Voctro Labs allows to perform several voice transformation effects including pitch transposition, pitch shifting or vibrato, as well as modifying the timbre of the voice to, for example, make it sound more male or female-like [12]. This library is based on a spectral modelling of the voice signal, manipulating the signal at individual glottal pulses [6]. More recent research on transforming the expression and timbre of the singing voice, allow for example for modify the mouth opening [1], which could allow to simulate the different mouth movements of different singers in the choir.

The main problem of these approaches is the lack of realism achieved when changing the voice timbre, producing in some cases a cartoon-like voice if the transformation is too large. If the transformations are subtle, there is the risk that the voices keep sounding too similar, as in this case we cannot apply pitch transposition within singers in the same section.

2.2.1.2 Voice cloning approach (deep learning)

This approach aims to improve the limitations in the voice transformation above, especially in terms of the realism of the cloned voices. Here the approach is to generate multiple synthetic voices of the choir by using a small amount of data by a new soloist singer.

For the case of large choir synthesis, modeling a large number of timbres would require recording a full dataset for each individual voice, which would be impractical. Therefore it would be desirable to create voices from small amounts of singer data. In fact, we have seen in recent TTS research several promising results that apply voice cloning techniques to modern deep learning based models [2, 17]. Based on the previous work at UPF and Voctro Labs by Merlijn Blaauw [4], we suggest here to employ voice cloning techniques to create a large number of individual choir voices using voice cloning techniques.

The term Voice cloning, also known as *voice fitting* or *speaker adaptation*, refers to a technique that leverages data from many singers combined with a small amount of data from the target soloist singer. This allow to create a voice model that outperforms a model trained on just the adaptation target data from scratch.

In this approach, depicted in Figure 2.2, we need first to train a multispeaker model (multisinger in our case). Basically it is a single neural network able to model all voices in the dataset simultaneously, by indicating the speaker/singer identity on an additional conditioning vector. This multispeaker model can be trained with 4-6 hours of annotated recordings. The next step is to adapt the multispeaker model to the new target speaker/singer, but using only a small amount of additional data. For example, according to recent research works, with 2-3 minutes of additional data is sufficient to generate a synthetic voice that can be identified as the target speaker.

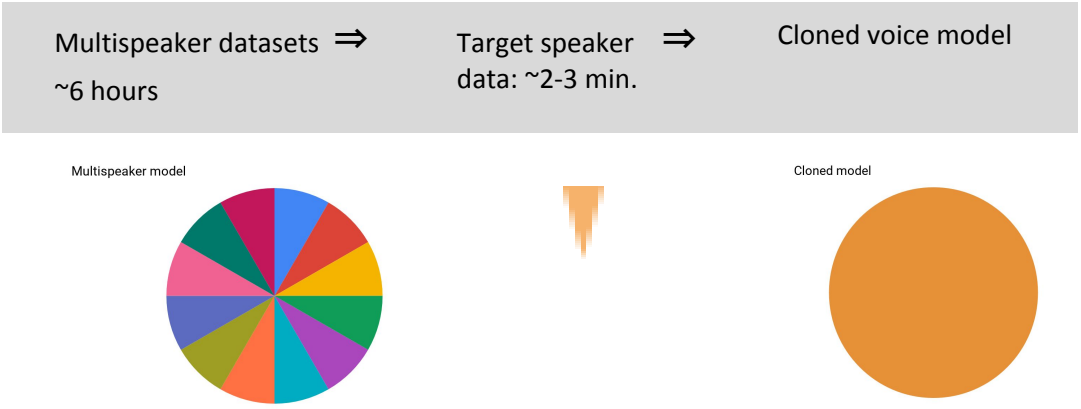


Figure 2.2. Graphical simplification of the Voice cloning approach. From a large dataset for creating the multispeaker model (left coloured circle), we can adapt a voice cloned model (right orange circle) with small amount of new target speaker data.

2.2.2 Solo-to-choir rendering

2.2.2.1 Spectral Morphing

Another strategy is to apply transformation to a solo voice in order to make it sound as a large, unison choir. This can be achieve using voice morphing [5]. The basic idea behind this strategy is to morph the solo voice with a sustained vowel of a recorded unison choir. The morph combines the pitch and smoothed timbre of the solo voice with the local spectrum of the unison choir. This strategy, however, is limited since it only works for voiced segments

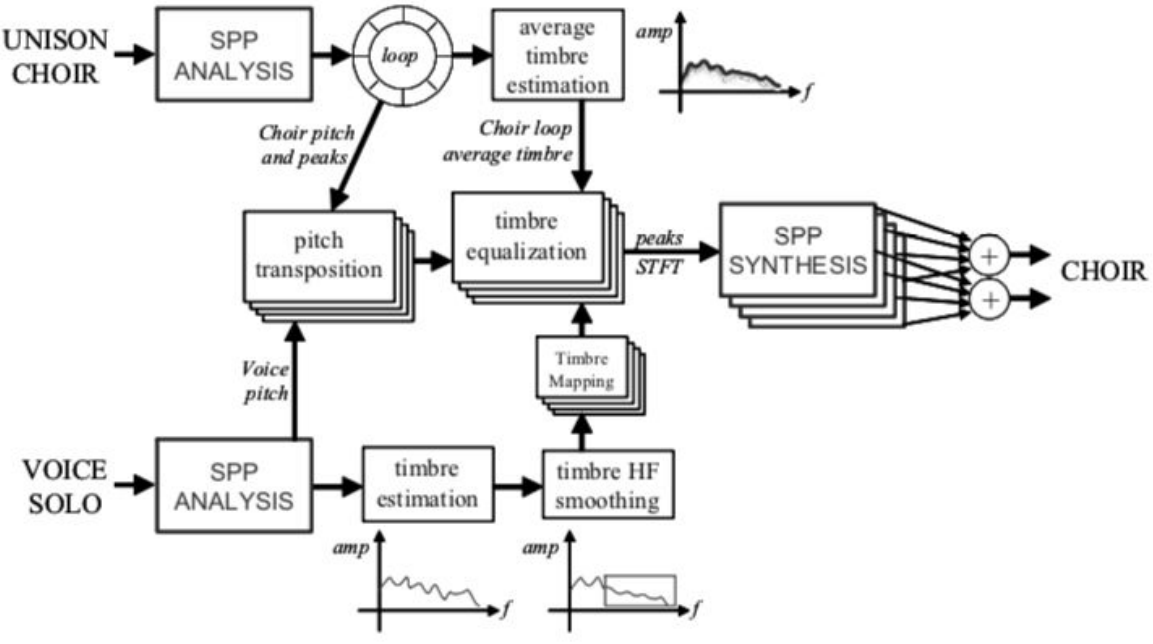


Figure 2.3. Morph strategy for generating unison choir from solo voice. Figure reproduced with permission from the original paper [5].

2.2.2.2 Solot-to-choir Wavenet-based vocoder

Finally, one of our strategies to obtain a realistic choir sound is to use a neural vocoder. In this case the input would be the vocoder features corresponding to a monophonic solo voice of a choir section, and the output will be a waveform of the choir sound.

We have not found previous approaches using a neural vocoder for the case of choir section synthesis, but the original WaveNet paper [15] already demonstrated that this type of models can cope with polyphonic musical signals such as piano.

One important aspect to take into account is that we need to record and create specific datasets. The dataset required for this type of neural vocoder models needs to be large. In our case therefore we will need to record separated choir sections, ideally collecting between 30 minutes and 1 hour of recordings per section. These requirements are further detailed in section 4 (Datasets resources and tools).

2.3 Source Separation and Emphasis

We aim to adapt existing state-of-the-art deep learning based source separation algorithms as well as propose new algorithms for source separation for the case of SATB choirs. A typical SATB choir consists of four parts, Soprano, Alto, Tenor and Bass. Each part covers a distinct range of fundamental frequencies and often with multiple singers each. Source separation in this scenario would refer to the isolation of the individual voices singing in the choir. We will start considering just one singer per voice, singing acapella, and a known number of sources (starting with 2 voices and going up to 4). This can then be extended to the case where the number of sources is unknown and further to include multiple singers per voice and also possibly the addition of noise and instrumental accompaniment.

Long term, this research will be useful for scores which are not available in the MusicXML/required format, but for which polyphonic recordings of SATB choirs are available through various medium (like YouTube). A direct application is for isolation or emphasis of a particular voice or singer in the choir performance. Indirect possibilities exist for score following, multi-modal separation, as well as choir modelling, as discussed in Section 2.2, amongst other fields.

Long term, this research will be useful for scores which are not available in the MusicXML/required format, but for which polyphonic recordings of SATB choirs are available through various medium (like YouTube). The following are potential applications, that we will explore:

1) Isolation/emphasis of a particular voice or singer in a choir performance.
2) A pre-processing step for score following.
3) Multi-modal source separation, given audio and video signals.
4) Choir modelling, as discussed in Section 2.2

3 Audio Processing tools

In this Section, we introduce the existing tools that Voctro Labs contributes to the project and which are being further developed and exploited in the TROMPA project. First, we talk about the current singing synthesis system (*VoSyn*); then, we introduce *Voiceful Editor*, a tool used internally for a wide

range of tasks that will be made available for the consortium for data annotation; finally, we explain how *VoMix*, a library for effect rendering and mixing, can be used to support the TROMPA use cases.

3.1 Annotation and Testing tool: Voiceful Editor

Voiceful Editor is a desktop application developed by Voctro Labs which at the moment is being used internally for a wide range of tasks related to audio annotation and testing. The application allows to load different data streams (audio, annotations, musical scores) in several formats, create and/or edit annotations and perform different operations on them to generate new data streams.

The motivation for using (and developing) *Voiceful Editor* within this Task of the project is two-fold:

1. As an annotation tool, it will be used for data preparation and annotation.
2. As a testing tool, it will be used for testing the synthesis algorithms developed during the project that will be finally integrated into the CSP.

In this sense, the main goal is to further develop the application to make it available for other project partners for the data preparation and annotation tasks.

3.1.1 Annotation features

There are different functionalities of *Voiceful Editor* that are useful to simplify and semi-automate the necessary data annotation step to be done for new datasets. As stated in Section 2, it is necessary to segment and phonetically transcribe the recordings in order to train the models.

For example, the application allows to perform pitch analysis on an audio file. This analysis generates a new data stream that is visualized in the application as a time series with pitch values in Hz along time (the sampling rate depending on the parameters of the analysis). It is possible to edit this time series, as well as to export it to a text file, which again is possible to load later in the application. It is also possible to perform onset analysis, which generates an annotation layer with timestamps where onsets are detected in the audio file.

In Section 4.1, where we explain the workflow for creating and annotating a dataset, we explain how *Voiceful Editor* is used in this context.

3.1.2 Synthesis algorithm testing

Voiceful Editor will also be used to test the synthesis algorithms. Since the application allows to load symbolic musical scores and to generate synthesized versions from them, we can replicate the same scenario to be implemented in the CSP.

Currently, the application works with soloist voices in English and Spanish. By the end of the project, it will support multiple languages and choral scores, generating one audio file for each voice.

3.2 Voice Model Creation Framework (VoSyn)

VoSyn is Voctro Labs' library for voice generation. It works both for speech and singing and it is integrable in a wide range of platforms. *VoSyn* works with several voice models and new ones can be created and used with it. In this section, we start by explaining how a new voice model can be created, we discuss the supported languages and detail the integration and software components of the application.

3.2.1 Creating a singing synthesis voice model

In section 1.3 we have introduced the workflow for creating a synthetic singing voice with modern techniques based on Deep Learning. In this section, we give more detail about the developed audio processing software tools and the steps to be followed. Creating a new synthetic singing voice implies the following steps:

1. **Record a singer** singing a set of musical scores.
2. **Annotate the recordings:** Segment and phonetically transcribe the recordings. Note level transcription and segmentation can be obtained from the musical scores, provided that the singer does not excessively deviate from the written score.
3. **Extract acoustic features:** from the recordings (pitch and spectral data) and combine with the supervised annotations to create the dataset.
4. **Training process:** Train the models to learn to produce the acoustic features from phonetic and musical input sequences.
5. **Rendering engine:** usable implementation of a neural network to loads the trained models and generate an audio output from an input score (lyrics and melody).

3.2.2 Supported languages

As described in the Description of Work, this task T3.3. (Audio Processing) in WP3 shall include multilingual support for the Choir Singers Pilot. Iin priority level, we plan to support: Spanish, Catalan, English, Latin, German.

Supporting a new language requires phonetic transcription tools for the new language. One resource of data and publicly available tools is the Festival framework, and its lightweight C++ implementation Flite[8] [29]. Other resources in Spanish is the TALP-UPC-Saga[9] [30], which includes phonetic transcription of all Spanish variants under a GNU license.

Currently the singing synthesis engine supports English, Spanish, Catalan and Italian. The English version is publicly available online on the Voiceful Cloud API[10] [33]. Language support for Spanish, Catalan and Italian (including Latin) is still under development and running on internal prototypes.

3.3 VoSyn library

The *VoSyn* library is available as a C++ SDK that implements the configurable neural networks for synthesis. Regarding the training process, this is done using Python scripts.

3.3.1 Technical details

3.3.1.1 Training scripts

The training scripts consists on Python scripts using frameworks such as TensorFlow. It includes training scripts (source code) and example datasets for creating data models of singers. This work was done in collaboration with the MTG researchers in the scope of the CASAS[11] project.

[8] http://www.festvox.org/flite/
[9] https://github.com/TALP-UPC/saga
[10] http://www.voiceful.io/demos.html
[11] https://www.upf.edu/web/mtg/casas

3.3.1.2 Synthesis engine library (VoSyn API)

The synthesis engine consists on a C++ implementation implementing configurable neural networks. It consists of a custom C++ implementation of a multilingual voice synthesis engine developed by Voctro Labs. It is based on the algorithm described in Neural Parametric Singing Synthesis, (See section 2.1).

This rendering engine is able to load the voice models exported by the Python/TensorFlow scripts described in step 5 above.

The performance on a off-the-shelf computer is >6x faster than real time. It means that to render a 1-minute score, it will take 10 seconds of CPU single-thread processing. Note that we could take advantage of parallel processing by rendering independently different musical phrases when separated by rests/silences in the score.

The C++ implementation is cross-platform and can be deployed either on server (Linux) or client side applications (Linux,MacOS, Windows, Android, iOS).

3.4 Additional Mixing and Rendering tools (VoMix)

3.4.1 Mixing and mastering

As we have seen, the case of synthesis for choir singing implies synthesizing each of the choir sections (or singers) separately. Rendering the whole choir, thus, implies mixing the choir parts (Soprano, Alto, Tenor, Bass or SATB) with their corresponding levels and pannings. In addition, generating pleasant-sounding choir voices will require applying sound effects, particularly reverb.

VoMix is a library that can be used for both purposes (applying effects and mixing), since it works as a virtual DAW with all standard audio effects. For the particular case of choir singing in TROMPA, it is relevant to mention that *VoMix* implements convolution reverb. Convolution reverb allows to simulating the reverb of a real physical space by convolving the input signal with the impulse response of the space being modeled.

There are many available impulse responses that can be used for this. For example, the The Open Acoustic Impulse Response Library[12] [22] contains a catalogue of freely-available impulse responses. There are also many commercial impulse response libraries such as Chris Hein's *Churches*[13] [25], which are suitable for the choir use case.

3.4.2 Effect-chain designer tools

The *VoMix FX designer* tool is a desktop application that allows to create effects chains to be used with the *VoMix* library. A screenshot of the application is depicted in Figure 3.1. The application allows to create several tracks that consist on a canvas to which sound effects can be added and whose parameters can be edited. The order in which the effects are placed in the canvas (from top to down) is the order in which they will be applied in the audio. In addition, there is a Master track whose effects are applied over the mix of the rest of the tracks. The application allows to load audio files to each of the tracks so that the result of the effect chain can be heard. Clicking "play" on the Master applies the effect chains to each of the tracks, mixes them with the desired panning and volume values and applies the effects in the Master track to the mix.

[12] http://www.openairlib.net/
[13] http://www.chrishein.net/web/IR_Churches.html

All effect chains can be exported to a JSON-formatted text file which can be later loaded in the application or used in *VoMix*. Figure 3.2 shows a real example of how a preset for an effect chain with a high-pass filter and reverb looks like in the JSON format.

This tool will be used within the TROMPA project in order to generate presets for mixing the different voices in the choir, as well as to create presets for effects to be applied on the generated synthetic voices (e.g. compression, reverb).

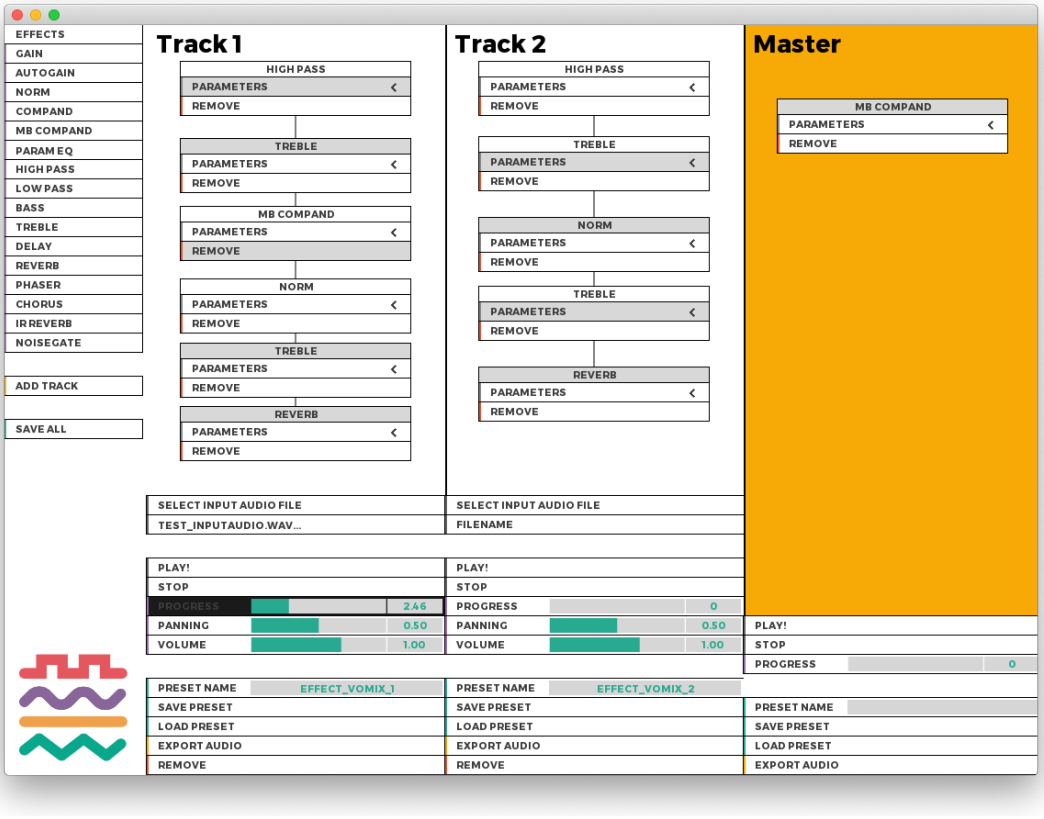


Figure 3.1. VoMix effect chain designer tool.

```
{
    "FXChain": [{
        "effect": "HIGHPASS",
        "params": {
            "freq": 150,
            "width": 0.71
        }
    }, {
        "effect": "REVERB",
        "params": {
            "hfDamping": 50,
            "preDelay": 0.0,
            "reverberance": 50,
            "roomScale": 50,
            "stereoDepth": 100,
            "wetGain": 0.0,
            "wetOnly": false
        }
```

```
    }],
    "panning": 0.5,
    "volume": 1
}
```


Figure 3.2. A VoMix preset in JSON format. This preset would apply a high-pass filter and algorithmic reverb.

4 Data resources and tools

4.1 Data gathering and annotation workflow for the choir case

Recording data for creating choir singing synthesis models has specific requirements. As we have seen, there are different possible strategies to follow in order to train the models, from modelling individual singers to whole choir sections. Accordingly, recording data for this purpose should be done in a way that is flexible for testing different strategies.

An ideal dataset for training choir singing synthesis algorithms would consist of isolated recordings from each of the choir singers, recorded with the whole choir singing simultaneously in a realistic environment. In practical terms, this is hardly feasible. Even when using directional microphones, there is a leakage between different microphones. This is particularly problematic if sound from one choir section is captured by microphones in another section. For this reason, a practical solution is to record each choir section separately. In order to overcome the possible problem of synchronization between each recording, it is possible to record the conductor in the first recording session and display the video in the subsequent sessions.

Our previous experience with solo singing recordings suggests that it is best to record pieces that are in the repertoire of the singer / choir. This reduces the number of necessary takes and guarantees better results in terms of performance (timing, intonation). It is important to make sure that the score being used for the recordings is the same that is available, in digital symbolic format, for posterior data annotation.

In order to get quality audio, recordings must be done in a studio or acoustically appropriate space with short reverberation time.. To minimize leakage between tracks, it is desirable to use cardioid directive microphones.

Any Digital Audio Workstation (DAW) that easily allows to perform several multi-track takes is appropriate for this kind of recordings. Additionally, it is convenient to be able to allow singers to listen to references (e.g. piano) for intonation purposes. This feature is also common among popular DAW software programs.

After recording the data, it must be annotated in order to be ready for training the models. As we introduced in Section 3.1, we use *Voiceful Editor* for this purpose. Figure 4.1 shows a snapshot of the application being used for annotating a solo singing performance of “Happy Birthday”.

The upper panel shows the waveform of the audio of the performance. The annotation task consists on aligning the musical score and the phonetic transcription to the performance. *Voiceful Editor* allows to load MusicXML scores (in the Figure, the score is represented with a piano roll visualization in the second panel). The start and end times of the notes must be moved by drag-and-dropping the vertical dashed lines. The same applies for the phonetic transcription, which in the Figure is depicted in the upper panel. Each segment in the phonetic transcription corresponds

to a phoneme. This transcription can be generated from scratch or imported from a .lab file containing the start time, end time and label for each phoneme[14]. Finally, these data can be exported to new MusicXML and .lab files.

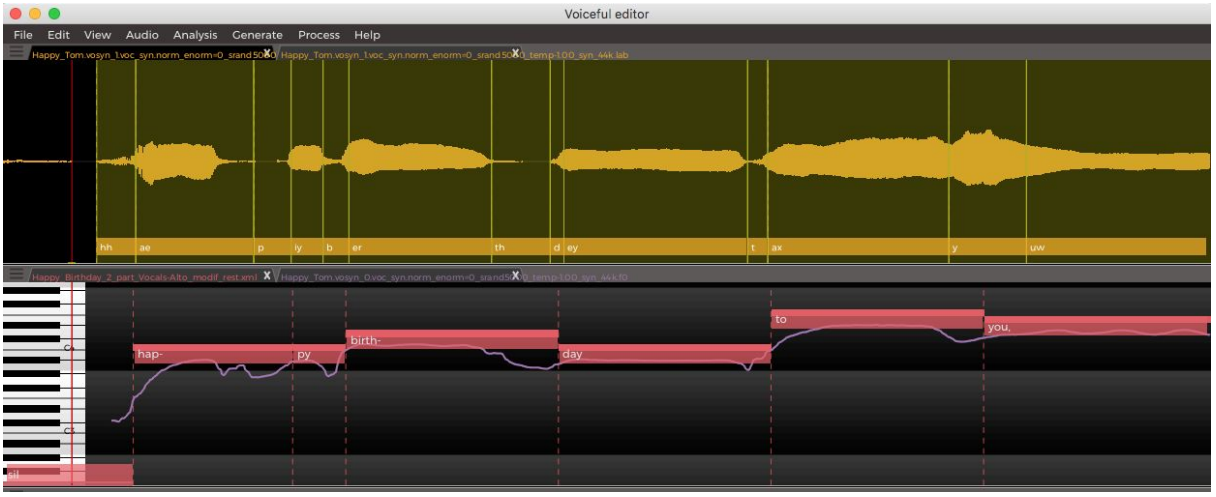


Figure 4.1. Data annotation in *Voiceful Editor.*

4.2 Datasets

For the creation of choir singing synthesis, we need recordings from real singers. We will focus on the most common case of choral music repertoire, which consists of four sections SATB (Soprano, Alto, Tenor and Bass).

As pointed out in Section 2.2 (Strategies for modeling choir synthesis), we will need at least recordings from four different singers: two female (Soprano, Alto) and two male (Tenor, Bass). But ideally we want to obtain a wider variety of voice timbres to reproduce the sound of a choir with more fidelity. Therefore, we aim to work with larger datasets with several singers per section.

Next we detail the characteristics of existing datasets that we can use within the TROMPA project. And after, the characteristics of the new datasets that we plan to prepare during the project. The goal to record new datasets is two-fold: first, to have a dataset that fulfills the quality and technical needs of multilingual choir synthesis; and second to provide open data for further research in the area of Music Information Processing.

4.2.1 Existing datasets

We compiled five datasets of singing recordings that are of interest for generating choir singing synthesis. These datasets have been recorded in the scope of previous projects. They can be used in the scope of the TROMPA project for R&D purposes, and the corresponding partner (UPF and VOCTRO) have obtained the corresponding permission by the singers.

The UPF can provide two datasets of choir recordings, while Voctro Labs can provide three datasets of soloists. These datasets will allow us to carry out the initial experiments on choir singing synthesis. With the results of these first experiments, we can determine the requirements in terms

[14] LAB files are used in the HTS synthesis system [26]. The monophone labels (mono) are just the phoneme sequence. Both formats have the start and end times of each phoneme, in ten-millionths of a second, so to get times in seconds, add a decimal point before 7 digits from the end.

of the new datasets to be recorded in 2019-2020. Table 4.1 summarizes the characteristics of the existing datasets:

Dataset Name	Choir Singer	Language	Size (minutes)	Sections	Recording type	Score type
UPF-CASAS-1-Bruckner	Cor Anton Bruckner (BCN)	Catalan/ Spanish		SATB (12 singers)		Pseudo-singing (2 pitches)
UPF-CASAS-2-ESMUC	ESMUC Choir (BCN)	German	~20 minutes	SATB (12 singers)	a) Single singers (with leakage mics) b) Monophonic Section	3 classical pieces
VOCTRO-Soprano-IT	P.Shots	Italian	31 minutes	Soprano	Single singer	12 classical pieces
VOCTRO-Pop-Female	Ayesha	English	90 minutes	Mid-range pop	Single singer	Pseudo-singing (3 pitches)
VOCTRO-Pop-Male	Randy	English	120 minutes	Mid-range pop	Single singer	40 pop songs

Table 4.1. Existing datasets available for the project.

Figure 4.2 shows the recording session with the ESMUC Choir, conducted by Lluís Vila, that took place at UPF (Barcelona) in November 29, 2018. The dataset recorded in this session is the *UPF-CASAS-2-ESMUC,* which will be used for research purposes within the TROMPA project.

Figure 4.2. Multichannel recording session of the ESMUC choir on November 29 2018.

4.2.2 Future datasets

During the TROMPA project, we will record new datasets for choir singing synthesis. The repertoire of the datasets to record is still to be determined. As mentioned above, it is generally best to record pieces that are already in the repertoire of the choir being recorded. The main goal to bare for future recordings is mainly to **support new languages** while at the same time keep collecting more data for all languages, since the achievable quality directly depends on the amount of data for model training.

Regarding the addition of new languages, something to consider is that professional choirs are able to trained to sing in several languages from the most common repertoires. For example, the UPF-CASAS-2-ESMUC contains german repertoire, while the native languages of most singers in the choir are Catalan and Spanish.

5 Integration with Choir Singers Pilot and further exploitation

As stated in the introduction, the synthesis technology developed in this Task will be integrated into the Choir Singers Pilot (CSP). This pilot covers two main use cases, one for **conductors**, and one for **singers**. Conductors will be able to create repertoires of choral pieces through TROMPA's **Contributor Environment** and singers will be able to practice their parts singing along with the synthesized voices of these repertoire voices. The functional requirements defined in Deliverable 2.1 - *Early Requirements*, establish the basis for this integration.

In order to integrate the synthesis into the functionality of the CSP, we will extend the existing Voiceful Cloud API, which already allows to synthesize singing voice using a RESTful API. The documentation for this Cloud API is available online[15] [34]. The plans to extend the Voiceful Cloud API for this integration will be explained in detail in the upcoming Deliverable 5.3: *TROMPA Processing Library*.

5.1 Additional developments for exploitation

In work package 7 (WP7 Dissemination and Exploitation), we will address the exploitation plans related to the choir singing synthesis technologies developed in this task T3.3. Audio Processing. Task 7.6 (Exploitation) deals with looking for sustainable business around classical music. Yearly reports will help determining the best strategy for exploitation, the type of product and its technical requirements. In that sense, we foresee additional technology developments linked to the opportunities drawn from the exploitation plan and the feedback received from the CSP (Choir Singers Pilot).

The architecture of the CSP shall be suitable for a future exploitation of the singing synthesis technologies in form of a SaaS (Software-as-a-Service). We have identified several commercial apps for choir singers (e.g. Singerhood[16], Carus[17]). These apps provide a reduced catalogue of scores with associated audio recordings, typically a single voice per section (SATB). We can offer to the

[15] https://cloud.voctrolabs.com/docs/api
[16] https://www.singerhood.com/
[17] https://www.carus-verlag.com/en/digital-media/carus-music-the-choir-app/

developer companies an options to automatically generate a choir synthesis version of any score using the API. However, this might require to develop additional functions to the Cloud API for a more efficient workflow adapted to the client needs. Nonetheless, we will explore another technical approach to make the libraries available for integration in the client side for browser-based applications. We will explore emscripten / webassembly[18] [27] implementations for the synthesis and analysis algorithms using the Web Audio API[19] [35], which would allow to replicate the whole use case of the CSP on the client side. We have already developed and tested webassembly implementations of the voice transformation algorithms for real-time voice transformation [13].

Also, we could foresee also the integration of the choir singing engine integrated as a production tool or plugin for a DAW offline (e.g. Cubase, ProTools) or online (e.g. Soundtrap); or score composition applications (e.g MuseScore, Finale). In these cases, the host application would need to interact directly with the API of the C++ library. Specific developments might be required depending on the host application, for example related to the audio buffering.

An additional application to bare in mind for choir singing synthesis is 3D-audio rendering. Virtual and Augmented Reality applications are increasingly common and popular, and choir singing performances are suitable to be listened in such setups, since spatialization plays a major role in perception of such performances.

6 Conclusion

This Document details the framework under which Task 3.3, Audio Processing, is being developed within the TROMPA project. As we have seen, choir singing synthesis is a challenging task that will imply extending many already-existing libraries and tools.

We have shown how new voice models can be created and explained what are the particularities that choir singing introduces. Also, we have detailed the existing tools that we will use and extend to support these tasks, which include data annotation.

There are existing datasets that we can use, but it is necessary to build new datasets in order to cover several languages. Also, the more data we collect, the better the quality of the voice models that we create will be.

All this work will be integrated in the Choir Singers Pilot by the end of the project. This application will benefit from choir singing synthesis to support choir singers for individual practice. We have already detailed how this integration will be done, using the Voiceful Cloud Service.

Finally, it is important to consider other possibilities for exploitation of the developed technologies beyond the pilot. We have identified some options that will be kept in mind while refining the requirements of the developed technologies.

[18] https://emscripten.org/
[19] https://developer.mozilla.org/en-US/docs/Web/API/Web_Audio_API

7 References

7.1 Written references

[1] Ardaillon, L., & Roebel, A. (2017). A mouth opening effect based on pole modification for expressive singing voice transformation. In Interspeech. Stockholm, Sweden.

[2] Arik, S. Ö., Chen, J., Peng, K., Ping, W., & Zhou, Y. (2018). Neural Voice Cloning with a Few Samples. CoRR, abs/1802.0.

[3] Blaauw, M., Bonada, J., Blaauw, M., & Bonada, J. (2017). A Neural Parametric Singing Synthesizer Modeling Timbre and Expression from Natural Songs. Applied Sciences, 7(12), 1313. http://doi.org/10.3390/app7121313

[4] Blaauw, M., Bonada, J., & Daido, R. (2019). Data Efficient Voice Cloning for Neural Singing Synthesis. Retrieved from http://arxiv.org/abs/1902.07292

[5] Bonada, J. (2005). Voice Solo to Unison Choir Transformation. In Audio Engineering Society Convention 118.

[6] Bonada, J. (2008). Wide-Band Harmonic Sinusoidal Modeling. In International Conference on Digital Audio Effects. Helsinki, Finland.

[7] Cuesta, H., Gómez, E., Martorell, A., & Loáiciga, F. (2018). Analysis of Intonation in Unison Choir Singing. In 15th International Conference on Music Perception and Cognition (ICMPC). Graz (Austria).

[8] Hua, K. (2018). Modeling Singing F0 With Neural Network Driven Transition-Sustain Models. Retrieved from http://arxiv.org/abs/1803.04030

[9] Janer, J., Bonada, J., & Blaauw, M. (2006). Performance-driven control for sample-based singing voice synthesis. In Proc. of DAFx (Vol. 6, pp. 41–44).

[10] Janer, J., Gorlow, S., & Arimoto, K. (2014). OBRAMUS: A System for Object-Based Retouch of Amateur Music. Audio Engineering Society Convention 137, 1–9.

[11] Jin, Z., Mysore, G. J., Diverdi, S., Lu, J., & Finkelstein, A. (2017). Voco: Text-based insertion and replacement in audio narration. ACM Transactions on Graphics (TOG), 36(4), 96.

[12] Mayor, O., Bonada, J., & Janer, J. (2011). Audio Transformation Technologies Applied to Video Games. In 41st AES Conference: Audio for Games. London, UK: AES.

[13] Mayor O., Janer J., Parra H., & Sarasúa Á. (2018). VOICEFUL: Voice Analysis, Transformation and Synthesis on the Web. In Web Audio Conference. Berlin.

[14] Morise, M., Yokomori, F., & Ozawa, K. (2016). WORLD: A vocoder-based high-quality speech synthesis system for real-time applications. IEICE Transactions on Information and Systems, E99D(7), 1877–1884. http://doi.org/10.1587/transinf.2015EDP7457

[15] Oord, A. van den, Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kavukcuoglu, K. (2016). WaveNet: A Generative Model for Raw Audio. http://doi.org/10.1109/ICASSP.2009.4960364

[16] Ping, W., Peng, K., Gibiansky, A., Arik, S. Ö., Kannan, A., Narang, S., Miller, J. (2017). Deep Voice 3: 2000-Speaker Neural Text-to-Speech. CoRR, abs/1710.0.

[17] Shen, J., Pang, R., Weiss, R. J., Schuster, M., Jaitly, N., Yang, Z., Wu, Y. (2017). Natural TTS Synthesis by Conditioning WaveNet on Mel Spectrogram Predictions. CoRR, abs/1712.0.

[18] Taigman, Y., Wolf, L., Polyak, A., & Nachmani, E. (2017). Voice Synthesis for in-the-Wild Speakers via a Phonological Loop. CoRR, abs/1707.0.

[19] Ternström, S. (1993). Perceptual evaluations of voice scatter in unison choir sounds. Journal of Voice, 7(2), 129–135. http://doi.org/10.1016/S0892-1997(05)80342-X

[20] Tokuda, K., Yoshimura, T., Masuko, T., Kobayashi, T., & Kitamura, T. (2000). Speech parameter generation algorithms for HMM-based speech synthesis. In ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings (Vol. 3, pp. 1315–1318). IEEE. http://doi.org/10.1109/ICASSP.2000.861820

[21] Wang, Y., Skerry-Ryan, R. J., Stanton, D., Wu, Y., Weiss, R. J., Jaitly, N., Saurous, R. A. (2017). Tacotron: A Fully End-to-End Text-To-Speech Synthesis Model. CoRR, abs/1703.1.

7.2 Web references

[22] Arts and Humanities Research Council. Open Acoustic Impulse Response Library (OpenAIR). Retrieved February 25, 2019: http://www.openairlib.net/

[23] Audio Engineering Society. 2017 AES International Conference on Semantic Audio. Retrieved February 25, 2019: http://www.aes.org/conferences/2017/semantic/

[24] Celemony GmbH. Melodyne. Retrieved February 25, 2019: https://www.celemony.com/en/start

[25] Chris Hein. Impulse Response Churches Collection. Retrieved February 25, 2019: http://www.chrishein.net/web/IR_Churches.html

[26] Columbia Speech Lab. Preparing Data for Training an HTS Voice. Retrieved February 25, 2019: http://www.cs.columbia.edu/~ecooper/tts/data.html

[27] Emscripten Contributors. Emscripten documentation. Retrieved February 25, 2019: https://emscripten.org/

[28] HMM/DNN-based speech synthesis system (HTS). Retrieved February 25, 2019: http://hts.sp.nitech.ac.jp/

[29] Language Technologies Institute at Carnegie Mellon University. CMU Flite: Speech Synthesizer. Retrieved February 25, 2019: http://www.festvox.org/flite/

[30] Mariño, J., & Nogueiras, A. SAGA. Retrieved February 25, 2019: https://github.com/TALP-UPC/saga

[31] MEI Technical Team. Music Encoding Initiative. Retrieved February 25, 2019: https://music-encoding.org/

[32] The Centre for Speech Technology Research, University of Edinburgh. The Festival Speech Synthesis System. Retrieved February 25, 2019: http://www.cstr.ed.ac.uk/projects/festival/

[33] Voctro Labs. Voiceful Cloud API Demos. Retrieved February 25, 2019: https://www.voiceful.io/demos.html

[34] Voctro Labs. Voiceful Cloud API Documentation. Retrieved February 25, 2019: https://cloud.voctrolabs.com/docs/api/index.html

[35] W3C Audio WG. Web Audio API. Retrieved February 25, 2019: https://developer.mozilla.org/en-US/docs/Web/API/Web_Audio_API

[36] W3C Music Notation Community Group. MusicXML. Retrieved February 25, 2019: https://www.musicxml.com/

[37] Yamaha Corp. Vocaloid. Retrieved February 25, 2019: http://www.vocaloid.com/

7.3 Trademarks and copyrights

Voiceful is a trademark property of Voctro Labs (European and US registered trademarks).

7.4 List of abbreviations

Abbreviation	Description
API	Application Programming Interface
CSP	Choir Singers Pilot
DAW	Digital Audio Workstation
f_0	Fundamental frequency
GUI	Graphical User Interface
HMM	Hidden Markov Models
HTS	HMM/DNN-based Speech Synthesis System
JSON	Javascript Object Notation
MEI	Music Encoding Initiative
NPSS	Neura Parametric Singing Synthesis
REST	Representational State Transfer
SaaS	Software as a Service
SATB	Soprano, Alto, Tenor and Bass
TTS	Text-To-Speech
XML	eXtensible Markup Language
Partner	**Description**
UPF	University Pompeu Fabra
VL	Voctro Labs