INTERSPEECH 2016
September 8–12, 2016, San Francisco, USA

Expressive Singing Synthesis based on Unit Selection for the Singing Synthesis Challenge 2016

Jordi Bonada, Martí Umbert, Merlijn Blaauw

Music Technology Group, Universitat Pompeu Fabra, Spain
jordi.bonada@upf.edu, marti.umbert@upf.edu, merlijn.blaauw@upf.edu

Abstract

Sample and statistically based singing synthesizers typically require a large amount of data for automatically generating expressive synthetic performances. In this paper we present a singing synthesizer that using two rather small databases is able to generate expressive synthesis from an input consisting of notes and lyrics. The system is based on unit selection and uses the Wide-Band Harmonic Sinusoidal Model for transforming samples. The first database focuses on expression and consists of less than 2 minutes of free expressive singing using solely vowels. The second one is the timbre database which for the English case consists of roughly 35 minutes of monotonic singing of a set of sentences, one syllable per beat. The synthesis is divided in two steps. First, an expressive vowel singing performance of the target song is generated using the expression database. Next, this performance is used as input control of the synthesis using the timbre database and the target lyrics. A selection of synthetic performances have been submitted to the Interspeech Singing Synthesis Challenge 2016, in which they are compared to other competing systems.

Index Terms: singing voice synthesis, expression control, unit-selection.

1. Introduction

Modeling expressive singing voice is a difficult task. Humans are highly familiarized with the singing voice, human's main musical instrument, and can easily recognize any small artifacts or unnatural expressions. In addition, for a convincing expressive performance, we have to control many different features related to rhythm, dynamics, melody and timbre. Umbert et al. [1] provide a good review of approaches to expression control in singing voice synthesis. Sample and statistically based speech or singing synthesizers typically require a large amount of data for generating expressive synthetic performances of a reasonable quality [2, 3, 4, 5]. Our aim is to provide a good trade-off between the expressiveness and sound quality of the synthetic performance on the one hand, and the database size and effort put into creating it on the other hand. Another motivation is the participation in the Singing Synthesis Challenge 2016. In particular, this work is a continuation of our previous contributions on the expression control of singing voice synthesis [6, 7] and on voice modeling [8, 9].

In section 2 we detail the used methodology: how databases are created, how synthesis scores are built and how samples are selected and concatenated. In section 3, we provide insights on the synthesis results and plan an evaluation of the synthesis system for rating its sound quality and expressiveness and comparing it to a performance driven case. We finally propose some future refinements.

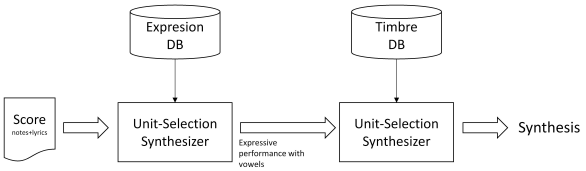

Figure 1: *Block diagram of the proposed synthesizer.*

2. Methodology

The proposed singing synthesizer generates expressive synthesis from an input consisting of notes (onset, duration) and lyrics. The synthesis is divided in two steps. First, an expressive vowel singing performance of the target song is generated using the expression database. In this step, we aim at generating natural and expressive fundamental frequency ($f0$) and dynamics trajectories. Next, this performance is used as input control of a second synthesis step that uses the timbre database and the target lyrics. The system is based on unit selection and uses a voice specific signal model for transforming and concatenating samples. The main advantages of such system are the preservation of fine expressive details found in the samples of the database, and also a significant usage of musical contextual information by means of the cost functions used in the unit selection process. The system is illustrated in Figure 1.

2.1. Databases

2.1.1. Expression database

The expression database consists of free expressive a cappella singing using solely vowels. In our experiments we just recorded 90 seconds of a male amateur singer. We asked him to sing diverse melodies so to lessen redundancy. Our main interest with this database is to capture typical $f0$ expressive gestures of the singer. One reason for using only vowels is that we can greatly reduce the microprosody effects caused by the different phonemes (e.g. decrease of several semitones in f0 during voiced fricatives). $f0$ is estimated using the Spectral Amplitude Correlation (SAC) algorithm [10]. The recordings are next transcribed into notes (onset, duration, frequency) using the algorithm described in [10], and manually revised.

It is well known that vowel onsets are aligned with perceived note onsets in singing [11]. Thus, the singer was instructed to use different vowels for consecutive notes in order to facilitate the estimation of the sung note onsets. Otherwise, it might be difficult to distinguish between scoops and portamentos, unless a noticeable dynamics or $f0$ related event clearly

http://dx.doi.org/10.21437/Interspeech.2016-872

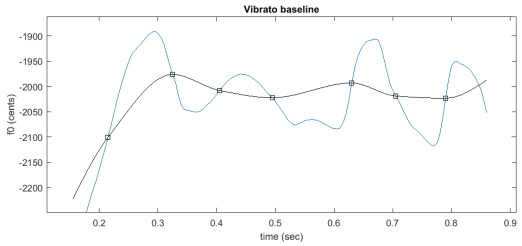

Figure 2: *Example of a vibrato baseline.*

marked the note onset. Vibratos segments were manually labeled. Then $f0$ is decomposed into a baseline function (free of modulations) and a residual. The baseline functions is estimated by interpolating $f0$ points of maximum absolute slope, where the slope is computed by the convolution of $f0$ with a linear decreasing kernel (e.g. $[L, L-1, ..., 0, ... -L+1, -L]$). In our experiments, the kernel has a length of 65 ms (13 frames of 5 ms). An example is shown in Figure 2.

2.1.2. Timbre database

The timbre database consists of monotonic singing of a set of sentences, one syllable per beat, i.e. singing the same note at a constant pace. The sentences are gathered from books, and chosen so to approximately maximize the coverage of phoneme pairs while minimizing the total length. For estimating the set of phoneme pairs and their relevance, we used a frequency histogram computed from the analysis of a collection of books. In our experiments, for the English case we recorded 524 sentences, which resulted in roughly 35 minutes.

We instructed the singer to sing the sentences using a single note, at a constant syllable rate, and with a constant voice quality. Moreover, we favored sequence of sentences with the same number of syllables. According to our experience, these constraints help to reduce the prosody effects related to the sentence meaning and to the actual words pronounced. By contrast, microprosody related to phoneme pronunciation is present and not greatly affected.

Recordings are manually segmented into sentences. All sentences are transcribed into phoneme sequences using the The CMU Pronouncing Dictionary [12]. Next, the Deterministic Annealing Expectation Maximization (DAEM) algorithm [13] is used to perform an automatic phonetic segmentation. The recordings are analyzed using the SAC and the Wide-Band Harmonic Sinusoidal Model (WBHSM) [8] algorithms for extracting $f0$ and harmonic parameters.

The last step is to estimate the microprosody, a component not considered in our previous work on expression control of singing synthesis [6, 7]. We are mostly interested in capturing $f0$ valleys typically occurring during certain consonants. With that aim, for each sentence, we estimate the difference between $f0$ and the sequence obtained by interpolating $f0$ values between vowels. We limit the residual to zero or negative values. Thus, the obtained residual is zero along vowels, and can be negative in consonants.

2.2. Expression score

The input of the system is a musical score consisting of a sequence of notes and lyrics. As described in Figure 1, the first step is to generate an expressive vowel performance of the target song using the expression database. For that it is necessary to compute the expression score. The Viterbi algorithm is used to compute the sequence of database units that better match the target song according to a cost function that considers transformation and concatenation costs. While in [7] units were sequences of three consecutive notes, here we use sequences of two notes, grouped in three unit classes: attack (silence to note transition), release (note to silence) and interval (note to note).

One requirement is that the class of selected database units and target song units has to match. Furthermore, interval units are categorized into ascendent, descendent or monotonic according to their interval. Ascendent units are not allowed to be selected for synthesizing descendent units and vice versa. The concatenation cost C_c is zero when consecutive units in the database are connected, 1 otherwise. This cost favors (when possible) to use long sequences from the database recordings. The transformation cost C_{tr} is computed as

$$C_{tr} = C_i + C_d \tag{1}$$

where C_i is the interval cost and C_d is the duration cost. The interval cost is computed as

$$C_i = |I_t - I_s| / 12 \cdot P_i \tag{2}$$

$$P_i = \left\{ \begin{array}{rcl} 1 & if & r \leq 1 \\ 1 + (r^2 - 1) \cdot w & if & r > 1 \end{array} \right\} \tag{3}$$

$$r = \frac{|I_t|}{max(0.5, |I_s|)} \tag{4}$$

$$w = \frac{min(3, max(1, 3/I_s)}{3} \tag{5}$$

where I_t and I_s are the target and source intervals expressed in semitones, and P_i is an extra penalty cost for the case where short source intervals are selected for large target intervals. The duration cost is computed as

$$C_d = C_{dn1} + C_{dn2} \tag{6}$$

where C_{dn1} and C_{dn2} are the duration costs corresponding to each note. For a given note, the duration cost is defined as

$$C_{dn} = d \cdot P_d \tag{7}$$

$$P_d = \left\{ \begin{array}{rcl} 1 & if & d \geq 0 \\ 1 - d/4 & if & d < 0 \end{array} \right\} \tag{8}$$

$$d = log_2(D_t / D_s) \tag{9}$$

where D_t and D_s are the target and source durations expressed in seconds, and P_d is an extra penalty cost for the case where source notes are compressed. An extra penalty cost of 10 is added if there is a class mismatch between the previous unit in the database song and the previous unit in the target song, also between the following ones.

2.3. Timbre score

A second synthesis step is to compute the timbre score out of the target song notes, lyrics, and the expressive vowel singing performance. The goal is to generate an expressive song combining the voice characteristics of the timbre database and the $f0$ and dynamics characteristics of the vowel singing performance. For that we need to compute the timbre score. As in the previous section, we use the Viterbi algorithm to compute the sequence of source units that best match the target song according to a cost function considering transformation and concatenation costs. In our case units are sequences of two consecutive phonemes (i.e. diphonemes).

We often expect to find one syllable per note, typically containing vowels and consonants. One important aspect is that the vowel onset has to be aligned with the note onset, hence the consonants preceding the vowel have to be advanced in time before the actual note onset. In the end, we create a map between notes and the actual phonemes sung within each note. For determining the phoneme durations we use a simple algorithm based on statistics computed from the timbre database. For each non-vowel target phoneme, we select the best unit candidates (with a pruning of 20) in the database according to the costs next defined, considering both the diphonemes that connect the previous phoneme with the current one, and those connecting the current phoneme with the following one. We estimate the mean duration of those candidates. Then, given the mean durations of each phoneme in a note, we fit the durations so that they fill the whole note. Vowels can be as long as needed. However, for ensuring a minimum presence of vowels in short notes we constrain the vowel duration to be at least a 25% of the note. In case the sum of durations of the non-vowel phonemes is more than 75%, those are equally compressed as needed.

The concatenation cost C_c is zero when consecutive units in the database are connected. Otherwise, a large cost of 15 is added when the connected phoneme is a vowel, 2.5 otherwise. This cost greatly favors (when possible) to use long sequences from the database recordings, especially for vowels. The transformation cost C_{tr} is computed as

$$C_{tr} = C_{f0} + C_d + C_{ph} \tag{10}$$

where C_{f0} is the cost related to $f0$, C_d is the duration cost, and C_{ph} is the phonetic cost. Only diphoneme samples matching the target diphoneme are allowed. C_{ph} refers to a longer phonetic context covering the previous and following diphonemes existing in the database recording and in the target score. Essentially, C_{ph} is zero if both diphonemes are matched, otherwise for each diphoneme compared a cost of 0.125 is set for matching the phonetic type (e.g. voiced plosives), 0.1875 for matching a similar phonetic type (according to a configuration parameter), 0.25 otherwise. Specifically for vowels, if the longer phonetic context is not matched, we add an extra cost of 5. This greatly favors longer phonetic contexts for vowels than for the rest of phonemes. If the timbre database is rather small, it is likely that certain diphonemes existing in the target song are missing in the database. For such cases, diphoneme candidates of the same or similar phonetic types are allowed. C_{f0} is zero for the silence phoneme, and for the rest of phonemes is computed as

$$C_{f0} = \left\{ \begin{array}{cc} \frac{|P_s - P_t|}{12} & if \quad P_s > -\infty \, and \, P_t > -\infty \\ 0 & otherwise \end{array} \right\} \tag{11}$$

where P_s and P_t are respectively the source and target note $f0$ in cents. The duration cost is computed as

$$C_d = |log_2(D_t/D_s)| \cdot P_d \tag{12}$$

$$P_d = \left\{ \begin{array}{cc} 1 & if \quad r \leq 1 \, or \, ph \notin vowels \\ 1 + (r-1) \cdot w & if \quad r > 1 \, and \, ph \in vowels \end{array} \right\} \tag{13}$$

$$r = D_t / D_s \tag{14}$$

$$w = \frac{min(6, max(1, 0.4/D_s)}{6} \tag{15}$$

where D_t and D_s are the target and source durations expressed in seconds, and P_d is an extra penalty cost for the case where short database vowels are selected for large target vowels.

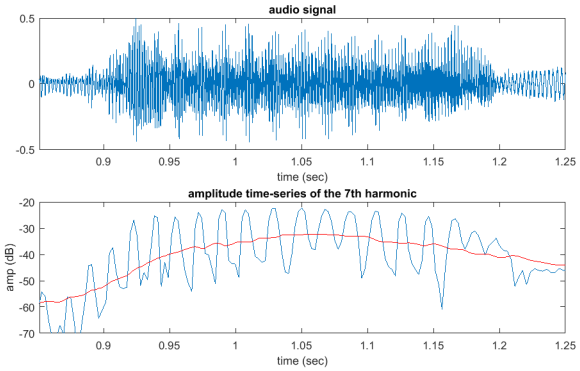

Figure 3: *LF (red) and HF (blue) decomposition of the 7^{th} harmonic amplitude time-series of a growl utterance.*

2.4. WBHSM concatenative synthesizer

The waveform synthesizer is a concatenative synthesizer and uses a refined version of the WBHSM algorithm [8] for transforming samples with high quality.

2.4.1. Analysis

This algorithm is pitch synchronous. Period onsets are determined by an algorithm that favors placing onsets at positions where harmonic phases are maximally flat [14]. Each voice period is analyzed with a certain windowing configuration that sets the zeros of the Fourier transform of the window at multiples of f0. This property reduces the interference between harmonics, and allows the estimation of harmonic parameters using a temporal resolution close to one period of the signal, thus providing a good trade off between time and frequency resolution. On the other hand, unvoiced excerpts are segmented into equidistant frames (each 5.8 ms) and analyzed with a similar scheme.

The output of the analysis consists on a set of sinusoidal parameters per period. For each period, frequencies are multiples of the estimated $f0$ (or the frame rate in unvoiced segments). Amplitude and phase values represent not only the harmonic content but also other signal components (e.g. breathy noise, modulations) that are present within each harmonic band.

Furthermore, a novelty over the original algorithm in [8] is that harmonic amplitude time series are decomposed into slow (LF) and rapid (HF) variations in relatively stable voiced segments (i.e. with low values of $f0$ and energy derivatives). Each component can be independently transformed and added together before the synthesis step. The motivation is to separate the harmonic content from breathy noise and modulations caused by different voice qualities. This method effectively allows to separate the modulations occurring in a recording with growl or fry utterances (see Figure 3), and to transform them with high quality. For each period, a spectral envelope (or timbre) is estimated from the LF component.

2.4.2. Transformation

The most basic transformations are $f0$ transposition, timbre mapping, filtering and time-scaling. Synthesis voice period (or frame) onsets are set depending on the transposition and time-scaling transformation values. Each synthesized frame is mapped to an input time. This time is used to estimate the out-

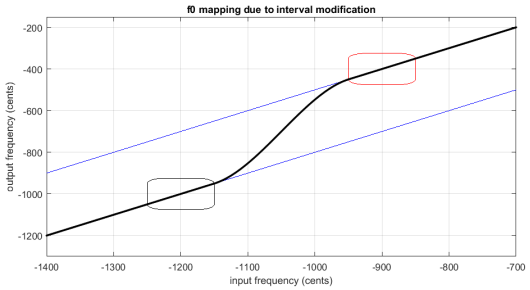

Figure 4: $f0$ *mapping function for a note unit transformation. Source interval: +3 semitones (notes at -1200 and -900 cents). Target interval: +6 semitones (notes at -1000 and -400 cents).* $f0$ *shift for first and second note (blue).*

put features (timbre, $f0$) by interpolating the surrounding input frames. Furthermore, timbre is scaled depending on the transformation parameters (timbre mapping and transposition).

The LF component of the synthesized sinusoidal amplitudes are computed by estimating the timbre values at multiples of the synthesis $f0$. The HF component is obtained by looping the input HF time-series. For each harmonic time-series, we compute the cross-correlation function between the last time used and the current mapped input time. The cross-correlations functions of the first harmonics (up to 10) are added together. If the maximum peak is above a certain threshold (3.5 in our experiments), it is used to determine the next HF position. Otherwise, the minimum value is used as the next HF position. The aim is to continue period modulations, but also to preserve noisy time-series. Both LF and HF components are added together.

Another improvement over the original WBHSM algorithm in [8] is that for voiced frames, phases are set by a minimum phase filter computed from the LF harmonic amplitudes. In addition, the (unwrapped) phase differences between consecutive voice periods are added to the synthesized phases. This helps to incorporate the aperiodic components to the synthesis sound, and improve its naturalness.

2.4.3. Unit transformation and concatenation

The first step of proposed synthesizer consists in rendering the expression score by transforming and concatenating units of the expression database. The sequence of units is set by the expression score. Units are sequences of two consecutive notes. Each unit is transformed so to match the target notes and duration. The note modification is achieved by a applying an $f0$ mapping determined by the source and target notes. Figure 4 shows the resulting mapping for a source interval of +3 semitones expanded to a target interval of +6 semitones. The $f0$ contours are shifted below the first note and above the second note, but scaled in between.

Transformed units are concatenated to produce continuous feature contours (timbre, $f0$. The concatenation process cross-fades the feature contours of the overlapping note between transformed units. Our intention is that most of the interval transition gesture of each unit is preserved during the synthesis process. While in previous works we manually set the transition segments and used them to determine the $f0$ cross-fade position, now we propose to determine it by minimizing the sum of three costs: distance to the middle of the note, distance to the note reference $f0$, and absolute $f0$ derivative. Vowel timbre cross-fading is set just at the end of the overlapping note. If vibratos are present, another novelty with respect to our previous work is that the residual (i.e. difference between $f0$ and the baseline, see Figure 2) is looped using the cross-correlation function similarly as for the HF component explained previously. This method effectively preserves the vibrato characteristics. The vibrato residual cross-fading is performed at the beginning of the overlapping note, so that mostly one vibrato is used along the note.

The second step of the synthesizer consists in rendering the timbre score by transforming and concatenating units of the timbre database (diphonemes). Feature of the overlapping phoneme are cross-faded, aiming at producing continuous transitions. Crossfading is set between the 40% and the 90% of each phoneme, except when gaps are detected and then used for cross-fading. Period onsets are synchronized in the cross-fading area. LF and HF components are cross-faded, as well as the $f0$ microprosody. Finally, a time-varying gain is applied to the synthesis performance so to match the energy contour of the input performance. The gain is estimated for vowels and interpolated in between to avoid exaggerating consonants, since the input performance consists of vowel singing.

3. Evaluation and discussion

A selection of synthetic performances submitted to the Interspeech 2016 Singing Synthesis Challenge can be downloaded from [15], including a cappella versions as well as mixes with background music. Figure 5 shows an example of the energy and $f0$ contours of a synthetic vowel belonging to the jazz standard "But not for me". Notes are also plotted. We observe that the contours are rich in details: several vibratos appear, with time-varying characteristics, even together with long scoops in the highest notes.

In the future, we plan to evaluate our system with a listening test comparing (a) synthesis with automatic expression vs (b) performance driven synthesis from the same singer and from a different singer. Possible refinements are to expand the musical context considered in unit selection and to enrich the current energy control with some parameters related to timbre (e.g. spectral slope). Another future direction is to include voice quality related expressions, such as growl or fry, in the expression database. In that direction, we show at the end of the "Autumn leaves" song from [15] that a convincing growl can be already generated by the current system.

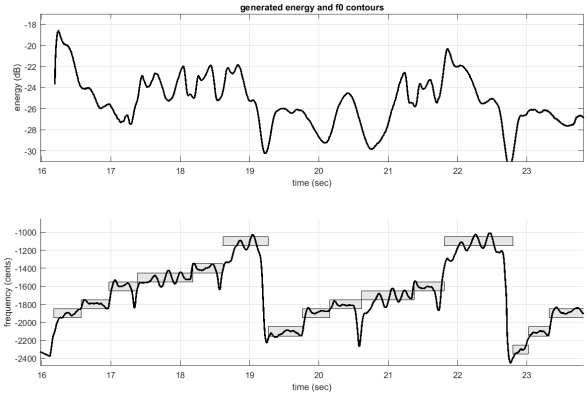

Figure 5: *Energy and* $f0$ *contours of a synthetic performance.*

4. References

[1] M. Umbert, J. Bonada, M. Goto, T. Nakano, and J. Sundberg, "Expression control in singing voice synthesis: Features, approaches, evaluation, and challenges," *IEEE Signal Processing Magazine*, vol. 32, pp. 55–73, 11/2015 2015.

[2] K. Tokuda, Y. Nankaku, T. Toda, H. Zen, J. Yamagishi, and K.Oura, "Speech synthesis based on hidden markov models," *Proceddings of the IEEE*, vol. 101, no. 5, pp. 1234–1252, May 2013.

[3] K. Nakamura, K. Oura, Y. Nankaku, and K. Tokuda, "Hidden markov model-based english singing voice synthesis," *IEICE*, vol. J97-D, no. 11, pp. 1572–1581, October 2014.

[4] M. Umbert, J. Bonada, and M. Blaauw, "Systematic database creation for expressive singing voice synthesis control," in *8th ISCA Speech Synthesis Workshop (SSW8)*, Barcelona, 31/09/2013 2013, pp. 213–216.

[5] Y. Qian, Z.-J. Yan, Y.-J. Wu, F. K. Soong, X. Zhuang, and S. Kong, "An hmm trajectory tiling (htt) approach to high quality tts," in *11th Annual Conference of the International Speech Communication Association, InterSpeech 2010*, Japan, September 2010, pp. 422–425.

[6] M. Umbert, J. Bonada, and M. Blaauw, "Generating singing voice expression contours based on unit selection," in *Stockholm Music Acoustics Conference*, Stockholm, Sweden, 30/07/2013 2013, pp. 315–320.

[7] M. Umbert, "Expression control of singing voice synthesis: Modeling pitch and dynamics with unit selection and statistical approaches," Ph.D. dissertation, Universitat Pompeu Fabra, Barcelona, 01/2016 2016.

[8] J. Bonada, "Wide-band harmonic sinusoidal modeling," in *International Conference on Digital Audio Effects*, Helsinki, Finland, 2008.

[9] J.Bonada and M. Blaauw, "Generation of growl-type voice qualities by spectral morphing," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Vancouver, Canada, 2013.

[10] E. Gómez and J. Bonada, "Towards computer-assisted flamenco transcription: An experimental comparison of automatic transcription algorithms as applied to a cappella singing," *Computer Music Journal*, vol. 37, pp. 73–90, 2013.

[11] J. Sundberg, *The Science of the Singing Voice*. DeKalb IL: Northern Illinois University Press, 1987.

[12] "The cmu pronouncing dictionary." [Online]. Available: http://www.speech.cs.cmu.edu/cgi-bin/cmudict

[13] N. Ueda and R. Nakano, "Deterministic annealing em algorithm," *Neural Netw.*, vol. 11, no. 2, pp. 271–282, Mar. 1998.

[14] J. Bonada, "Voice processing and synthesis by performance sampling and spectral models," Ph.D. dissertation, Universitat Pompeu Fabra, Barcelona, 2009.

[15] "Audio examples for the singing synthesis challenge 2016." [Online]. Available: http://www.dtic.upf.edu/~jbonada/BonSSChallenge2016.rar