ADAPTIVE PITCH-SHIFTING WITH APPLICATIONS TO INTONATION ADJUSTMENT IN A CAPPELLA RECORDINGS

Sebastian Rosenzweig

International Audio Laboratories Erlangen
Erlangen, Germany
sebastian.rosenzweig@audiolabs-erlangen.de

Simon Schwär

International Audio Laboratories Erlangen
Erlangen, Germany
simon.schwaer@fau.de

Jonathan Driedger

Chordify
jonathan@chordify.net

Meinard Müller

International Audio Laboratories Erlangen
Erlangen, Germany
meinard.mueller@audiolabs-erlangen.de

ABSTRACT

A central challenge for a cappella singers is to adjust their intonation and to stay in tune relative to their fellow singers. During editing of a cappella recordings, one may want to adjust local intonation of individual singers or account for global intonation drifts over time. This requires applying a time-varying pitch-shift to the audio recording, which we refer to as adaptive pitch-shifting. In this context, existing (semi-)automatic approaches are either labor-intensive or face technical and musical limitations. In this work, we present automatic methods and tools for adaptive pitch-shifting with applications to intonation adjustment in a cappella recordings. To this end, we show how to incorporate time-varying information into existing pitch-shifting algorithms that are based on resampling and time-scale modification (TSM). Furthermore, we release an open-source Python toolbox, which includes a variety of TSM algorithms and an implementation of our method. Finally, we show the potential of our tools by two case studies on global and local intonation adjustment in a cappella recordings using a publicly available multitrack dataset of amateur choral singing.

1. INTRODUCTION

A cappella singing is a wide-spread vocal performance practice where one or multiple singers sing together without instrumental accompaniment. Without having an instrumental reference, it becomes crucial that a cappella singers adjust their pitch relative to their fellow singers [1, 2]. Performances (in particular of amateur or semi-professional ensembles) can exhibit different kinds of intonation inaccuracies, ranging from individual, local intonation problems (e.g., singers singing a note too low or too high) to global intonation drifts over time [1, 3, 4, 5]. Figure 1 exemplifies such inaccuracies with an excerpt from an SATB (Soprano, Alto, Tenor, Bass) quartet performance, showing fundamental frequency (F0) trajectories on top of a reference derived from a musical score (visualized in gray). The figure illustrates two phenomena: first, the performance exhibits local intonation inaccuracies such as for the tenor voice (green), which sings the beginning of the first note slightly too low. Second, the performance exhibits a global intonation drift downwards over the course of the excerpt (all four F0-trajectories lay below the gray score reference at the end of the excerpt).

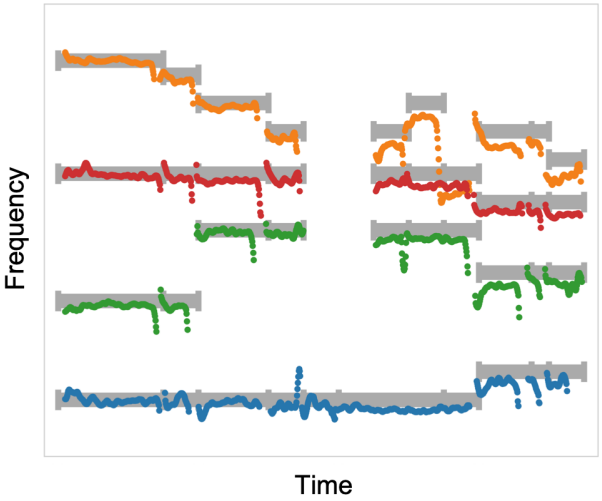

Figure 1: F0-trajectories of four-voice a cappella performance (soprano=orange, alto=red, tenor=green, bass=blue). The score reference is indicated in grey.

During postprocessing of a cappella recordings, one may want to adjust local or global intonation deviations using pitch-shifting techniques. Pitch-shifting is the task of changing an audio recording's pitch without altering its duration. Over the last decades, several conceptually different approaches have been proposed in the literature, ranging from time-domain algorithms [6, 7, 8, 9] to frequency domain approaches [10, 11]. An overview on several pitch-shifting approaches can be found in [12]. However, for adjusting local and global intonation in a cappella recordings, it is not sufficient to apply a single fixed pitch-shift to the recording, as Figure 1 demonstrates. Instead, it is necessary to apply a time-varying pitch-shift to the audio recording, which we refer to as adaptive pitch-shifting.

A naïve approach for adaptive pitch-shifting is to apply indi-

vidual pitch-shifts to small sections of an audio signal, e.g., using user-guided functionalities provided by most digital audio workstations. However, besides being labor-intensive, this approach can lead to audible "clicking" artifacts at pitch-shift transitions due to phase and other discontinuities. Previous research on adaptive pitch-shifting has been conducted in the context of audio restoration and "wow" reduction of gramophone and tape recordings [13, 14, 15]. Recently, a deep learning-based approach for adaptive pitch correction of singing performances with instrumental accompaniment has been proposed in [16]. State-of-the-art commercial tools such as Melodyne[1] or Antares AutoTune[2] offer semi-automatic functionalities for pitch correction according to different scales and tunings. However, due to the presence of global intonation drifts over time and a varying local intonation depending on the musical context, the assumption of a fixed (time-invariant) scale or tuning is problematic for a cappella music [1]. Popular open-source music processing libraries such as librosa [17] are often limited to fixed pitch-shifting functionalities. As an exception, the `C++` library Rubber Band[3] provides an interface for real-time pitch-shifting of an audio stream. Furthermore, the PyTSMod package [18] includes an adaptive pitch-shift implementation designed for monophonic audio.

In this article, we propose automatic methods and tools for adaptive pitch-shifting with applications to intonation adjustment in a cappella recordings. We base our work on an existing pitch-shifting method, which makes use of resampling and time-scale modification (TSM) [11]. As one contribution, we propose and formalize an extension to this method, which enables time-varying pitch-shifts. Furthermore, we release a Python re-implementation of a Matlab TSM toolbox [19], which we extended with an implementation of our adaptive pitch-shifting method. In order to show the potential of our method, we consider two case studies based on Dagstuhl ChoirSet [20], a publicly available dataset of a cappella performances. The first study targets the adjustment of global intonation, whereas our second study targets the adjustment of local intonation.

The remainder of this article is structured as follows. In Section 2, we review pitch-shifting via resampling in combination with TSM and introduce our adaptive pitch-shifting method. In Section 3, we give details on our Python toolbox. Finally, we address our two case studies in Section 4 and summarize our work in Section 5.

2. PITCH-SHIFTING VIA RESAMPLING AND TSM

Pitch-shifting can be seen as the complementary task to TSM [11, 12]. While TSM attempts to alter the duration of an audio recording without changing its pitch, pitch-shifting attempts to alter the pitch of an audio recording without changing its duration. In the following, we summarize existing TSM algorithms (Section 2.1), explain the basic principle of fixed (time-invariant) pitch-shifting using resampling and TSM (Section 2.2), and finally introduce our adaptive pitch-shifting method (Section 2.3).

2.1. TSM Algorithms

Over the last decades, several TSM algorithms have been proposed. In general, TSM algorithms can be subdivided into time-domain and frequency-domain approaches. Time-domain approaches typically rely on variants of the overlap-add (OLA) principle. In this case, an input signal is first decomposed into overlapping frames, which are relocated on the time axis in a second step to achieve the actual time-scale modification. Examples of time-domain algorithms are SOLA (*Synchronized* OLA) [21], TD-PSOLA (*Time-Domain Pitch-Synchronized* OLA) [6, 22, 23] or WSOLA (*Waveform-Similarity* OLA) [24]. A well-known frequency-domain approach is based on the phase vocoder technique [25, 26]. In order to obtain a time-scaled version of the input signal, the method relocates the frames of the input signal's short-time Fourier transform (STFT) [27] and applies a frequency-dependent phase correction. Recent works on TSM propose modifications of the phase vocoder technique [28] or use the phase vocoder in combination with non-negative matrix factorization [29]. While time-domain TSM methods are known to be well-suited for recordings with strong transient sound components, frequency-domain approaches typically perform well on recordings with strong harmonic sound components. This observation has been exploited by the approach in [30], which first conducts harmonic–percussive separation and then applies OLA on the percussive component and the technique based on the phase vocoder on the harmonic component. A more detailed review of several TSM methods can be found in [11].

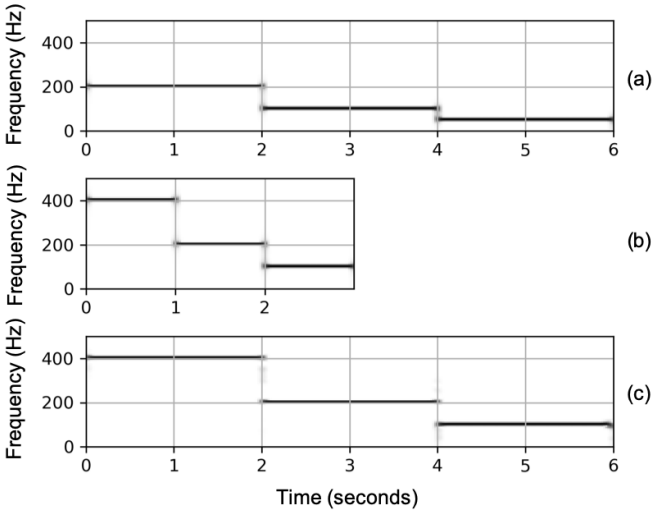

Figure 2: Pitch-shifting via resampling and TSM illustrated using power spectrograms. **(a)** Input signal. **(b)** Resampled signal. **(c)** Pitch-shifted signal after TSM application.

2.2. Fixed Pitch-Shifting

Resampling a given audio signal and playing it back at the original sampling rate changes its duration and pitch at the same time. In other words, resampling can be interpreted as a TSM procedure that additionally modifies the pitch of an audio signal. Pitch-preserving TSM algorithms, such as the ones mentioned in Section 2.1, can be used to compensate for the change in duration after resampling. Note that pitch-shifting can also be achieved by processing in reverse order (first performing TSM and then resampling) [12].

The processing steps for fixed (time-invariant) pitch-shifting

[1] `https://www.celemony.com/en/melodyne`
[2] `https://www.antarestech.com`
[3] `https://breakfastquay.com/rubberband/`

via resampling and subsequent TSM are illustrated in Figure 2. For illustrative purposes, we use a synthetic signal as input signal, which contains three sequentially played sinusoidal tones. Figure 2a shows a power spectrogram of our input signal. Let us assume our input signal is equidistantly sampled at a rate $F_{\mathrm{s}}^{\mathrm{in}}$ and we are given a fixed pitch-shift $p \in \mathbb{R}$ in cents. In a first step, we resample the given signal to have a new sampling rate $F_{\mathrm{s}}^{\mathrm{out}}$ defined by

$$F_{\mathrm{s}}^{\mathrm{out}} := F_{\mathrm{s}}^{\mathrm{in}} \cdot 2^{-p/1200}. \tag{1}$$

When playing back the resampled signal at the original sampling rate $F_{\mathrm{s}}^{\mathrm{in}}$, one can observe two effects. First, the signal's duration is scaled by a factor $\alpha_{\mathrm{RS}} \in \mathbb{R}_{>0}$ defined as

$$\alpha_{\mathrm{RS}} := \frac{F_{\mathrm{s}}^{\mathrm{out}}}{F_{\mathrm{s}}^{\mathrm{in}}} = 2^{-p/1200}. \tag{2}$$

Second, the signal is pitch-shifted by p cents. These two effects can be seen in Figure 2b for a pitch-shift of $p = 1200$ cents, which is equivalent to an octave in musical terms or a doubling of frequency in physical terms.

To compensate for the undesired time-scale modification, we then use a suitable pitch-preserving TSM algorithm to scale the signal to it's original duration. To this end, we stretch the signal with the factor $\alpha_{\mathrm{TSM}} \in \mathbb{R}_{>0}$ defined by

$$\alpha_{\mathrm{TSM}} := \alpha_{\mathrm{RS}}^{-1} = 2^{p/1200}. \tag{3}$$

For a pitch shift of $p = 1200$ cents we obtain $\alpha_{\mathrm{TSM}} = 2$. The resulting pitch-shifted signal is depicted in Figure 2c.

2.3. Adaptive Pitch-Shifting

Adaptive pitch-shifting is the task of applying a time-varying pitch shift to an audio signal. To this end, we extend the method for fixed pitch-shifting from Section 2.2. More specifically, we combine non-linear resampling with a technique referred to as non-linear TSM [11]. In the following, we explain our approach along with the example depicted in Figure 3.

Let us assume we are given an audio signal, which is equidistantly sampled at a sampling rate of $F_{\mathrm{s}}^{\mathrm{in}}$. As illustrative example, we again consider an input signal with three sequential sinusoidal tones, as visualized in Figure 3a. For the task of adaptive pitch-shifting, we model the pitch-shift p as a continuous time-varying function $p : \mathbb{R} \to \mathbb{R}$, which maps a time instance $t \in \mathbb{R}$ in seconds to a musical interval given in cents. Figure 3b shows p in our example, which consists of three parts: in the first part (0 to 2 s), the input signal should be left unshifted, in the second part (2 to 4 s), the signal should be frequency modulated, and in the third part (4 to 6 s), a frequency sweep should be applied.

In a first processing step, we perform non-linear resampling of our input signal. As explained earlier, resampling can be interpreted as a kind of pitch-modifying TSM. In this light, we first define a scaling factor function $\alpha_{\mathrm{RS}} : \mathbb{R} \to \mathbb{R}_{>0}$ that maps a time instance t to a scaling factor by

$$\alpha_{\mathrm{RS}}(t) := 2^{-p(t)/1200}. \tag{4}$$

The resulting $\alpha_{\mathrm{RS}}(t)$ for our example is depicted in Figure 3c. Subsequently, we introduce a non-linear and strictly monotonously increasing time-stretch function $\tau_{\mathrm{RS}} : \mathbb{R} \to \mathbb{R}$, which defines a mapping between time instances of an input and an output signal, by

$$\tau_{\mathrm{RS}}(t) := \int_0^t \alpha_{\mathrm{RS}}(t)\, dt. \tag{5}$$

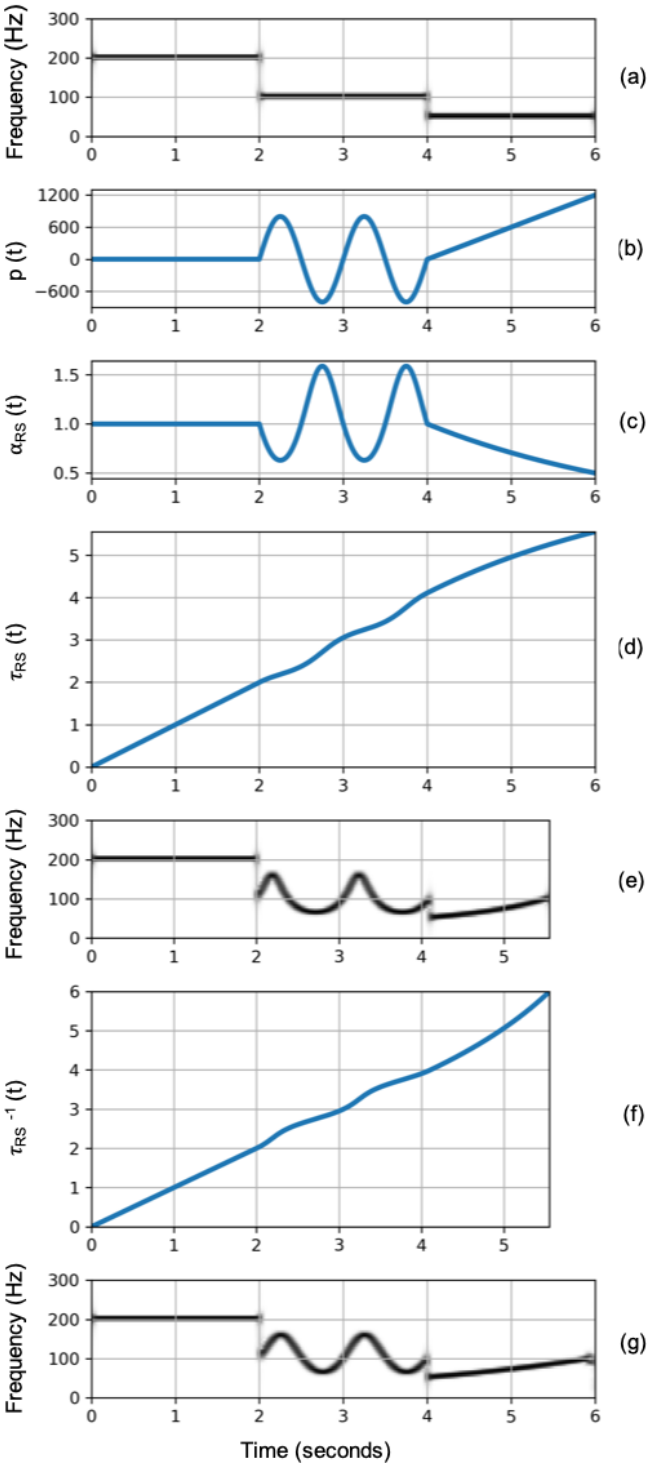

Figure 3: Adaptive pitch-shifting via non-linear resampling and non-linear TSM. **(a)** Power spectrogram of input signal. **(b)** Pitch-shift function. **(c)** Scaling factor function. **(d)** Time-stretch function. **(e)** Power spectrogram of resampled signal. **(f)** Inverse time-stretch function. **(g)** Power spectrogram of pitch-shifted signal.

The function τ_{RS} for our example is depicted in Figure 3d. As one can see, the first part of the function has a slope equal to one. As a consequence, this part of our example signal is mapped to the output signal without modification. The overall slope of the function's second part is slightly larger than one, leading to an expansion of this region in the output signal. The overall slope of the function's third part is slightly smaller than one, leading to a compression of this region in the output signal. By performing non-linear resampling according to the mapping defined by the function τ_{RS}, we obtain the signal depicted in Figure 3e.

Note that in practice, non-linear resampling can be done in many different ways [31, 32]. A comparison of resampling implementations in digital audio workstations can be found online[4]. Advanced resampling methods such as multirate filterbanks include lowpass filtering to avoid aliasing artefacts, but also require a windowing of the time-stretch function τ_{RS}. However, our goal is to adjust intonation with pitch-shifts in the order of a few cents up to roughly a semitone, where aliasing artefacts are less problematic. For the sake of simplicity, we therefore use cubic interpolation to non-linearly resample the input signal.

In a second processing step, we perform non-linear TSM on the resampled audio signal to retain the signal's original duration. To this end, we use a pitch-preserving TSM algorithm to non-linearly stretch the signal with respect to τ_{RS}^{-1}, which is depicted in Figure 3f. Further details on non-linear TSM can be found in [11, Section 7.1]. The resulting pitch-shifted audio signal is depicted in Figure 3g. As one can see, the adaptive pitch-shift p has been applied to our input signal.

3. PYTHON TOOLBOX

The release of open-source implementations along with scientific publications has become increasingly important in the field of music signal processing [12, 33]. Besides allowing for reproducing experimental results, publicly available implementations stimulate and support further research activities. In this spirit, we ported an exisiting Matlab TSM toolbox [19] to Python and expanded its functionality with our adaptive pitch-shifting method. Python is currently considered as the most used programming language in data science and machine learning. Our Python TSM toolbox is released an open source license[5].

In our re-implementation of the toolbox, we ensured that the naming conventions and usage of our Python implementation are basically the same as in the Matlab version. Table 1 provides an overview on the main algorithms, functions, and naming conventions of the Matlab and the Python toolbox. Furthermore, we tested all Python functions with respect to numerical identity to the Matlab implementations. In the following, we demonstrate the main functions of the Python toolbox using the code example in Listing 1.

```
1  # Load packages
2  import libtsm
3  import librosa
4  import numpy as np
5
6  # Load Audio File
7  fn_in = 'data/three_sinusoidals.wav'
8  x, Fs = librosa.load(fn_in, sr=22050)
9
10 # TSM Algorithms
11 alpha = 1.8  # scaling factor
12
13 y_wsola = libtsm.wsola_tsm(x, alpha)
14 y_pv = libtsm.pv_tsm(x, alpha)
15 y_hps = libtsm.hps_tsm(x, alpha)
16
17 # Fixed Pitch-Shifting (Figure 2)
18 p = 1200  # cents
19 y_psf = libtsm.pitch_shift(x, p)
20
21 # Adaptive Pitch-Shifting (Figure 3)
22 t = np.arange(0, len(x)/Fs, 1/Fs)  # sec
23 N = len(t)
24 t_1 = t[0:N//3]
25 t_2 = t[N//3:2*N//3]
26 t_3 = t[2*N//3:]
27
28 p = np.concatenate((np.zeros(len(t_1)),
       800*np.sin(2*np.pi*1*t_2),
       np.linspace(0, 1200, len(t_3))))  # cents
29
30 y_psa = libtsm.pitch_shift(x, p, t)
```

Listing 1: Code example using functions of `libtsm`.

As one can see in line 2, the TSM toolbox can be imported as a Python package `libtsm`. The toolbox includes short demo audio files, including our synthetic audio example from Section 2.2 and Section 2.3, which is loaded in lines 7–8. Lines 11–15 demonstrate the main TSM functions of the toolbox called with default settings. Note that each of the functions provides various other input arguments to tune the parameters of the algorithms. The input arguments are documented in the functions' docstrings.

Along with the TSM implementations, we added a function `pitch_shift()` to the toolbox, which implements our fixed and adaptive pitch-shifting algorithm. Lines 18–19 replicate the fixed pitch-shift by 1200 cents, as visualized in Figure 2. Adaptive pitch-shifting can be achieved using the same function by handing over two arrays of equal length, as shown in lines 22–28. The first array contains the pitch-shift values in cents, whereas the second array contains the time axis in seconds. Our example replicates the adaptive pitch-shift shown in Figure 3. A more detailed demonstration of all toolbox functions can be found in the Jupyter notebook `demo_libtsm.ipynb`, which is part of our toolbox.

4. APPLICATION: INTONATION ADJUSTMENT IN A CAPPELLA RECORDINGS

In the previous sections, we have presented a method for adaptive pitch-shifting (Section 2.3) as well as a Python toolbox with implementations of our method and a variety of TSM algorithms (Section 3). In this section, we show the potential of our method and our tools for adjusting global and local intonation in a cappella recordings.

As indicated in Section 2, the technical realization of our adaptive pitch-shifting method, in particular, the choice of a suitable resampling and TSM algorithm, depends on the acoustic properties of the input signal. In our application scenario, the versatility of the human voice imposes additional challenges on our pitch-shifting setup. Especially, an appropriate handling of fricatives, plosives, and formants is required to aviod a degradation of the audio quality. In the following, we present an extension to

[4] https://src.infinitewave.ca/

[5] https://www.audiolabs-erlangen.de/resources/MIR/2021-DAFX-AdaptivePitchShifting

Table 1: Main algorithms and implementations of the Matlab and Python TSM toolbox.

Algorithm	**Matlab Function**	**Python Function**
WSOLA/OLA [24]	`wsolaTSM()`	`wsola_tsm()`
Phase Vocoder TSM [25, 26]	`pvTSM()`	`pv_tsm()`
Harmonic–Percussive Separation TSM [30]	`hpTSM()`	`hps_tsm()`
Fixed Pitch-Shifting	`pitchShiftViaTSM()`	`pitch_shift()`
Adaptive Pitch-Shifting	–	`pitch_shift()`

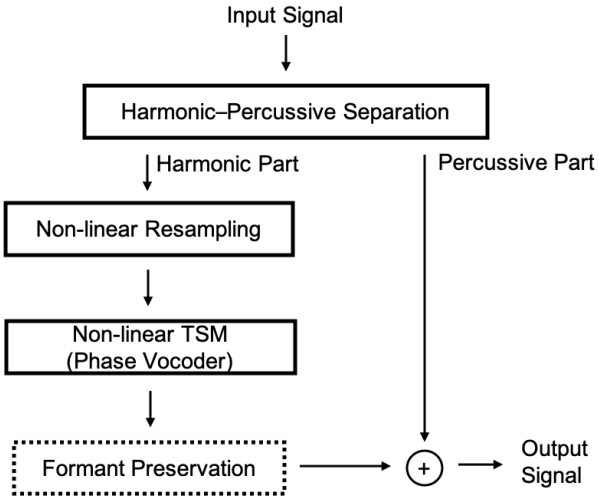

Figure 4: Overview on our intonation adjustment setup.

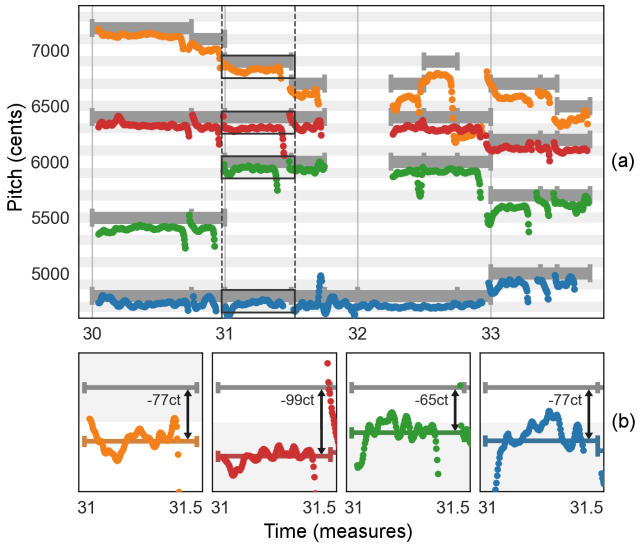

Figure 5: **(a)** Excerpt of F0-trajectories and score reference for a performance of *Locus Iste* (DCS, Quartet B, Take 3, measures 30–34). **(b)** Detailed view of the notes on the first beat in measure 31. Horizontal lines represent 12-TET pitch of the note (dark grey) and the median of the respective F0-trajectories (S=orange, A=red, T=green, B=blue).

our adaptive pitch-shifting method that accounts for these challenges. Our setup is depicted in Figure 4. Similar to the approach in [30], we first apply harmonic–percussive separation [34] on the input signal (also referred to as HPS). In a vocal recording, the percussive component typically includes fricatives, plosives, and other non-tonal background noise, whereas the harmonic component contains tonal elements. In our setup, we apply adaptive pitch-shifting only on the harmonic component. We use cubic interpolation for non-uniform resampling and the technique based on the phase vocoder for TSM. In order to avoid unnatural sounding pitch-shifted voices (sometimes referred to as the "chipmunk effect"), we include a formant preservation step [12, 22, 23] in our setup for monophonic input signals (recordings where only one voice is present). The formant preservation step first involves estimating the spectral envelopes of the original and the pitch-shifted signal from smoothed spectrogram representations. Subsequently, using the approach outlined in [35], the envelope of the pitch-shifted signal is corrected.

Note that this technical setup is only one possible way to realize adaptive pitch-shifting for our application scenario. A comparison of different pitch-shifting setups as well as a detailed evaluation of the musical quality is beyond the scope of this article and is left for future work. For an evaluation of the perceptual audio quality of the HPS-TSM approach, we refer to [30].

Given this technical setup, we show in two case studies how suitable pitch-shift functions p can be computed to achieve global intonation adjustments (Section 4.1) and local intonation adjustments (Section 4.2). Our studies are based on recordings from the *Dagstuhl ChoirSet* (DCS) [20], a multitrack dataset of amateur choral singing. The dataset includes quartet and choir performances of two choir pieces. The singers were recorded using a room microphone and several close-up microphones with little cross-talk (dynamic, headset, and larynx microphones). Furthermore, the dataset provides annotations of F0-trajectories for all singers and time-aligned score information. Figure 5a shows an excerpt of an SATB quartet performance of *Locus Iste* with a global intonation drift and several local intonation issues, which serves as a running example in our case studies. Figure 5b provides a detailed view on local intonation deviations and pitch fluctuations. Accompanying audio examples for our case studies are available online[5].

4.1. Case Study 1: Global Intonation Adjustment

In this case study, the task is to compensate a global intonation drift over the course of a performance. To this end, we first measure the

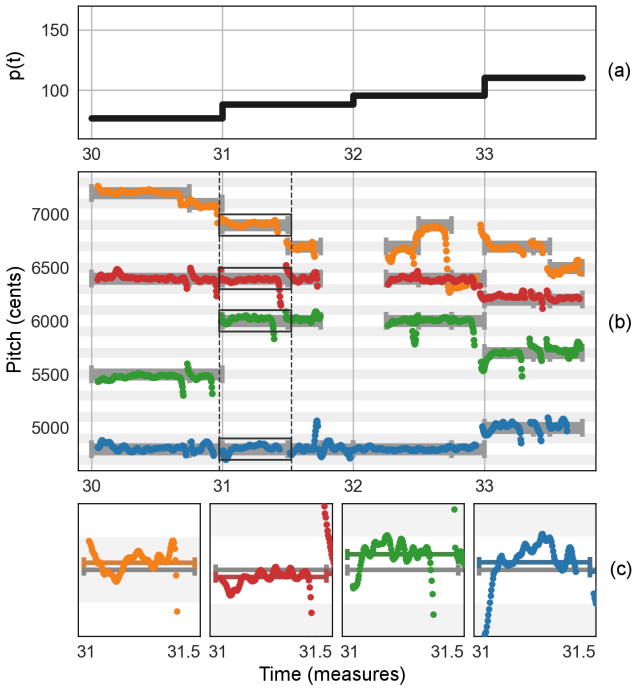

Figure 6: Global intonation adjustment in performance of *Locus Iste* (DCS, Quartet B, Take 3, measures 30–34). **(a)** Adaptive pitch-shifting function. **(b)** Globally adjusted F0-trajectories. **(c)** Detailed view of the notes on the first beat in measure 31.

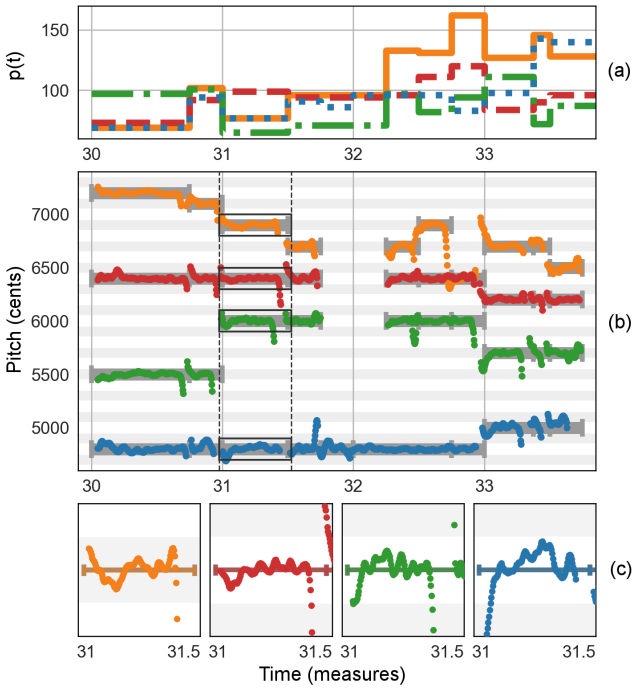

Figure 7: Local intonation adjustment in performance of *Locus Iste* (DCS, Quartet B, Take 3, measures 30–34). **(a)** Adaptive pitch-shifting functions for each voice. **(b)** Locally adjusted F0-trajectories. **(c)** Detailed view of the notes on the first beat in measure 31.

intonation drift over time and then input the inverted drift as a function p to our adaptive pitch-shifting algorithm. One way to measure intonation drift is to compute the deviations of the singers' F0-trajectories from the time-aligned musical score. Since the singers in our recordings tuned to tones played on a piano right before the performance, we compute the deviations to the notes' MIDI frequencies using 440 Hz as reference frequency for the note A4.

However, computing the deviations on a fine temporal level leads to highly fluctuating drift curves, which result in an unnatural "wobbling" in the pitch-shifted recording. Therefore, we introduce a temporal quantization of the measured intonation drifts. More precisely, we first compute the note-wise F0-median (see vertical colored lines in Figure 5b), and then average the note-wise deviations on a measure-level. After inverting the measured intonation drift curve, we obtain the pitch-shift function p, as depicted in Figure 6a for our excerpt. As one can see, p increases from roughly 77 cents to roughly 110 cents over the course of the excerpt, since the quartet drifts downwards.

The intonation adjustment can now be conducted either by applying the adaptive pitch-shift defined by the function p on each individual singer's microphone signal or on the polyphonic room microphone signal. The drift-corrected F0-trajectories for our example are shown in Figure 6b and a detailed view is provided in Figure 6c. As one can see, the drift is adjusted over the course of the four bars, whereas the local intonation is still fluctuating around the score reference. Furthermore, all note-internal pitch fluctuations are preserved.

4.2. Case Study 2: Local Intonation Adjustment

In our second case study, we show how to use adaptive pitch-shifting to adjust local intonation. As opposed to Section 4.1, we now compute an individual pitch-shift function p for each singer in the performance. To this end, we again compute the note-wise F0-median and its deviation from the aligned score reference, but this time, the temporal quantization of our measured deviations remains on a note-level. By inverting the measured deviations for the individual voices, we obtain the pitch-shift functions depicted in Figure 7a for our example. Note that adjusting local intonation to MIDI pitches in 12-TET is musically problematic in the context of western choral music [1]. In general, the task of measuring intonation in a cappella music using computational tools is subject to ongoing scientific discussions [3, 36, 37]. Therefore, the above described strategy mainly serves illustrative purposes.

The locally adjusted F0-trajectories are depicted in Figure 7b, while a detailed view is provided in Figure 7c. In contrast to the global intonation adjustment in Figure 6c, we can see that after pitch shifting, the note-wise F0-median now corresponds exactly to the 12-TET reference. Pitch variations within notes (e. g. vibrati and portamenti at the beginning of notes) are again preserved. In order to adjust these fluctuations, one would have to apply pitch-adjustments on a finer temporal level at the cost of an increasing unnaturalness of the pitch-shifted recordings.

5. CONCLUSIONS

In this work, we presented an automatic method for adaptive pitch-shifting of audio recordings based on non-linear resampling and TSM. Furthermore, we created an open source toolbox that includes implementations of various TSM algorithms and our proposed method. Finally, we showed the potential of our tools for adjusting global and local intonation in a cappella music. In future research, we plan to evaluate the perceptual quality of different intonation adjustment setups and investigate methods for computing musically meaningful pitch-shifting functions for intonation adjustments in a cappella recordings.

6. ACKNOWLEDGMENTS

This project is supported by the German Research Foundation (DFG MU 2686/12-1, MU 2686/13-1). The International Audio Laboratories Erlangen are a joint institution of the Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU) and Fraunhofer Institute for Integrated Circuits IIS. We thank Edgar Suarez, El Mehdi Lemnaouar and Miguel Gonzales for implementation support and Sascha Disch for valuable comments.

7. REFERENCES

[1] Per-Gunnar Alldahl, *Choral Intonation*, Gehrmans Musikförlag, 1990.

[2] Anke Grell, Johan Sundberg, Sten Ternström, Martin Ptok, and Eckart Altenmüller, "Rapid pitch correction in choir singers," *The Journal of the Acoustical Society of America*, vol. 126, no. 1, pp. 407–413, 2009.

[3] Johanna Devaney, *An Empirical Study of the Influence of Musical Context on Intonation Practices in Solo Singers and SATB Ensembles*, Ph.D. thesis, McGill University, Montreal, Canada, 2011.

[4] David M. Howard, "Intonation drift in a capella soprano, alto, tenor, bass quartet singing with key modulation," *Journal of Voice*, vol. 21, no. 3, pp. 300–315, 2007.

[5] Matthias Mauch, Klaus Frieler, and Simon Dixon, "Intonation in unaccompanied singing: Accuracy, drift, and a model of reference pitch memory," *Journal of the Acoustical Society of America*, vol. 136, no. 1, pp. 401–411, 2014.

[6] Francis Charpentier and M. G. Stella, "Diphone synthesis using an overlap-add technique for speech waveforms concatenation," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Tokyo, Japan, 1986, pp. 2015–2018, IEEE.

[7] Ken Bogdanowicz and Robert Belcher, "Using multiple processors for real-time audio effects," in *Audio Engineering Society Conference*. Audio Engineering Society, 1989.

[8] Keith Lent, "An efficient method for pitch shifting digitally sampled sounds," *Computer Music Journal*, vol. 13, no. 4, pp. 65–71, 1989.

[9] Azadeh Haghparast, Henri Penttinen, and Vesa Välimäki, "Real-time pitch-shifting of musical signals by a time-varying factor using normalized filtered correlation time-scale modification," in *Proceedings of the International Conference on Digital Audio Effects (DAFx)*, Bordeaux, France, September 2007, pp. 7–14.

[10] Christian Schörkhuber, Anssi Klapuri, and Alois Sontacchi, "Audio pitch shifting using the constant-q transform," *Journal of the Audio Engineering Society*, vol. 61, no. 7/8, pp. 562–572, 2013.

[11] Jonathan Driedger and Meinard Müller, "A review on time-scale modification of music signals," *Applied Sciences*, vol. 6, no. 2, pp. 57–82, February 2016.

[12] Udo Zölzer, *DAFX: Digital Audio Effects*, John Wiley & Sons, 2011.

[13] Simon J. Godsill and P. J. W. Rayner, "The restoration of pitch variation defects in gramophone recordings," in *Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, New Paltz, New York, USA, 1993, pp. 148–151.

[14] Andrzej Czyzewski, Marek Dziubinski, Andrzej Ciarkowski, Maciej Kulesza, Przemyslaw Maziewski, and Jozef Kotus, "New algorithms for wow and flutter detection and compensation in audio," *Proceedings of the Audio Engineering Society (AES) Convention*, vol. 6353, 2005.

[15] Andrzej Czyzewski, Przemyslaw Maziewski, and Adam Kupryjanow, "Reduction of parasitic pitch variations in archival musical recordings," *Signal Processing*, vol. 90, no. 4, pp. 981–990, 2010.

[16] Sanna Wager, George Tzanetakis, Cheng-i Wang, and Minje Kim, "Deep autotuner: A pitch correcting network for singing performances," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Barcelona, Spain, 2020, pp. 246–250.

[17] Brian McFee, Colin Raffel, Dawen Liang, Daniel P.W. Ellis, Matt McVicar, Eric Battenberg, and Oriol Nieto, "Librosa: Audio and music signal analysis in python," in *Proceedings the Python Science Conference*, Austin, Texas, USA, 2015, pp. 18–25.

[18] Sangeon Yong, Soonbeom Choi, and Juhan Nam, "PyTSMod: A python implementation of time-scale modification algorithms," in *Demos and Late Breaking News of the International Society for Music Information Retrieval Conference (ISMIR)*, Montréal, Canada, 2020.

[19] Jonathan Driedger and Meinard Müller, "TSM Toolbox: MATLAB implementations of time-scale modification algorithms," in *Proceedings of the International Conference on Digital Audio Effects (DAFx)*, Erlangen, Germany, 2014, pp. 249–256.

[20] Sebastian Rosenzweig, Helena Cuesta, Christof Weiß, Frank Scherbaum, Emilia Gómez, and Meinard Müller, "Dagstuhl ChoirSet: A multitrack dataset for MIR research on choral singing," *Transactions of the International Society for Music Information Retrieval (TISMIR)*, vol. 3, no. 1, pp. 98–110, 2020.

[21] Salim Roucos and Alexander M. Wilgus, "High quality time-scale modification for speech," in *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, April 1985, vol. 10, pp. 493–496.

[22] Robert Bristow-Johnson, "A detailed analysis of a time-domain formant-corrected pitch-shifting algorithm," in *Audio Engineering Society Convention*. Audio Engineering Society, 1993.

[23] Eric Moulines and Jean Laroche, "Non-parametric techniques for pitch-scale and time-scale modification of speech," *Speech Communication*, vol. 16, no. 2, pp. 175–205, 1995.

[24] Werner Verhelst and Marc Roelands, "An overlap-add technique based on waveform similarity (WSOLA) for high quality time-scale modification of speech," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Minneapolis, USA, 1993.

[25] James L. Flanagan and Roger M. Golden, "Phase vocoder," *Bell System Technical Journal*, vol. 45, pp. 1493–1509, 1966.

[26] Michael R. Portnoff, "Implementation of the digital phase vocoder using the fast Fourier transform," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 24, no. 3, pp. 243–248, 1976.

[27] Dennis Gabor, "Theory of communication," *Journal of the Institution of Electrical Engineers (IEE)*, vol. 93, no. 26, pp. 429–457, 1946.

[28] Zdenek Prusa and Nicki Holighaus, "Phase vocoder done right," in *Proceedings of the European Signal Processing Conference (EUSIPCO)*, Kos, Greece, 2017, pp. 976–980.

[29] Gerard Roma, Owen Green, and Pierre Alexandre Tremblay, "Time scale modification of audio using non-negative matrix factorization," in *Proceedings of the International Conference on Digital Audio Effects (DAFx)*, Birmingham, UK, 2019, pp. 1–6.

[30] Jonathan Driedger, Meinard Müller, and Sebastian Ewert, "Improving time-scale modification of music signals using harmonic-percussive separation," *IEEE Signal Processing Letters*, vol. 21, no. 1, pp. 105–109, 2014.

[31] Parishwad P Vaidyanathan, *Multirate systems and filter banks*, Pearson Education India, 2006.

[32] Julius Orion Smith, *Physical audio signal processing: For virtual musical instruments and audio effects*, W3K publishing, 2010.

[33] Brian McFee, Jong Wook Kim, Mark Cartwright, Justin Salamon, Rachel M. Bittner, and Juan Pablo Bello, "Open-source practices for music signal processing research: Recommendations for transparent, sustainable, and reproducible audio research," *IEEE Signal Processing Magazine*, vol. 36, no. 1, pp. 128–137, 2019.

[34] Derry FitzGerald, "Harmonic/percussive separation using median filtering," in *Proceedings of the International Conference on Digital Audio Effects (DAFx)*, Graz, Austria, September 2010, pp. 246–253.

[35] Mark Dolson, "The phase vocoder: A tutorial," *Computer Music Journal*, vol. 10, no. 4, pp. 14–27, 1986.

[36] Helena Cuesta, Emilia Gómez, Agustín Martorell, and Felipe Loáiciga, "Analysis of intonation in unison choir singing," in *Proceedings of the International Conference of Music Perception and Cognition (ICMPC)*, Graz, Austria, 2018, pp. 125–130.

[37] Christof Weiß, Sebastian J. Schlecht, Sebastian Rosenzweig, and Meinard Müller, "Towards measuring intonation quality of choir recordings: A case study on Bruckner's Locus Iste," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, Delft, The Netherlands, 2019, pp. 276–283.