Morphing Musical Instrument Sounds with the Sinusoidal Model in the Sound Morphing Toolbox

Marcelo Caetano

Morphing Musical Instrument Sounds with the Sinusoidal Model in the Sound Morphing Toolbox*

Marcelo Caetano[1,2] [0000−0002−5119−8964]

[1] Aix-Marseille Univ, CNRS, PRISM "Perception, Representations, Image, Sound, Music", Marseille, France marcelo.caetano@prism.cnrs.fr
https://www.prism.cnrs.fr/

[2] Schulich School of Music & CIRMMT, McGill University, Montreal, Quebec, Canada marcelo.caetano@mcgill.ca
https://www.cirmmt.org/

Abstract. Sound morphing stands out among the sound transformation techniques in the literature due to its creative and research potential. The aim of sound morphing is to gradually blur the categorical distinction between the source and target sounds by blending sensory attributes. As such, the focus and ultimate challenge of most sound morphing techniques is to interpolate across dimensions of timbre perception to achieve the desired result. There are several sound morphing proposals in the literature with few open-source implementations freely available, making it difficult to reproduce the results, compare models, or simply use them in other applications such as music composition, sound design, and timbre research. This work describes how to morph musical instrument sounds with the sinusoidal model using the sound morphing toolbox (SMT), a freely available and open-source piece of software. The text describes the audio processing steps required to morph sounds with the SMT using a step-by-step example to illustrate the need for and the result of each step. The SMT contains implementations of a sound morphing algorithm in MATLAB ® that were designed to be as easy as possible to understand and use, giving the user control over the result and full customization.

Keywords: Sound morphing · Musical instruments · Sinusoidal model · Musical timbre.

1 Introduction

Sound morphing has found creative, technical, and research applications in the literature. In music composition, sound morphing allows the exploration of the sonic continuum [32]. Notable examples include Jonathan Harvey's *Mortuos Plango, Vivos Voco* [18], Michael McNabb's *Dreamsong* [22], and Trevor

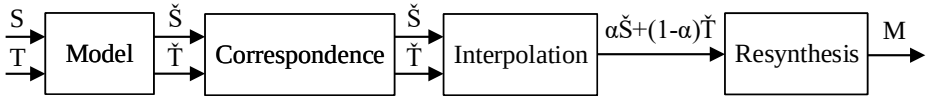
* This project has received funding from the European Union's Horizon 2020 research and innovation program under the Marie Skłodowska-Curie grant agreement No 831852 (MORPH)

Wishart's *Red Bird* [32]. These morphs were achieved by hand. Ideally, given the input sounds, sound morphing should allow automatically setting input parameters that control the morph to achieve the desired result [28]. Sound morphing is also used in audio processing, sound synthesis, and sound design. Tellman *et al.* [30] proposed a sound morphing technique based on sinusoidal modeling (SM) that is intended to improve the performance of a sample-based synthesizer by morphing between sounds of the same instrument to obtain intermediate pitches, dynamics, and other effects. Fitz *et al.* [12] use an SM called *Loris* to morph sounds. Sound morphing techniques have been used to investigate different aspects of timbre perception. Grey and Gordon [16] investigated the perceptual effect of exchanging the shape of the spectral energy distribution between pairs of musical instrument sounds. More recently, Carral [10] used spectral morphing to determine the just noticeable difference in timbre for trombone sounds. Siedenburg *et al.* [27] investigated the acoustic and categorical dissimilarity of musical timbre with morphing. However, these results are difficult to reuse, re-purpose, and build upon because seldom do we find freely available or open-source implementations of the morphing algorithms used in the literature.

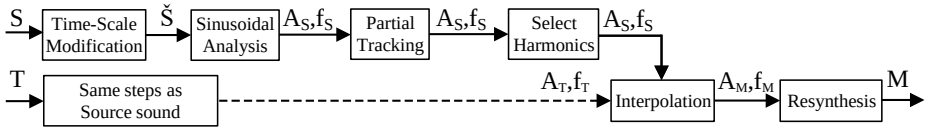
Currently, there are commercial morphing implementations available, such as Symbolic Sound's *Kyma* [3], SoundMorph's *Time Flux* [4], Melda Production's *MMorph* [5], and Zynaptic's *Morph* [6]. These commercial products typically have stable and bug-free implementations that can be controlled via a graphical user interface (GUI). However, besides the price, disadvantages such as little flexibility (i.e., control) and scarce technical information prevent their wider adoption in academic circles. A notable exception is *Kyma*, an implementation of the SM dubbed Loris [12] integrated in a full-fledged sound design environment. However, composers and researchers alike need to be able to understand the algorithms employed and control several parameters of the transformation. There also exist closed-source implementations based on algorithms whose technical details can be found in publications. Ircam's *Diphone Studio* [7] uses the SM to morph between sounds. Trevor Wishart's *Sound Loom* [8] also allows morphing sounds. These are controlled via a GUI and the manuals typically contain little technical information because composers are the target user.

There are freely available open-source morphing implementations, such as Google Magenta's *NSynth* [9], Mike Brookes's *Voicebox* [10], and Hideki Kawahara's *STRAIGHT* [11] and *SparkNG* [12]. However, these find limited use in musical instrument sound morphing. NSynth uses a neural network synthesizer trained on

[3] http://kyma.symbolicsound.com/
[4] https://www.soundmorph.com/product/24/timeflux
[5] https://www.meldaproduction.com/MMorph
[6] http://www.zynaptiq.com/morph/
[7] http://anasynth.ircam.fr/home/english/software/diphone-studio
[8] http://www.trevorwishart.co.uk/slfull.html
[9] https://magenta.tensorflow.org/nsynth-instrument
[10] http://www.ee.ic.ac.uk/hp/staff/dmb/voicebox/voicebox.html
[11] https://github.com/HidekiKawahara/legacy_STRAIGHT
[12] https://github.com/HidekiKawahara/SparkNG



(a) General sound morphing procedure.



(b) The audio processing steps with the SMT.

Fig. 1: Overview of the morphing procedure in the SMT

a dataset with sounds from commercial sample libraries instead of recordings from acoustic musical instruments. Voicebox, STRAIGHT, and SparkNG were optimized for speech and their performance with musical instrument sounds remains untested. Dedicated sound models usually perform poorly on other acoustic sources.

This article describes the sound morphing toolbox (SMT), which contains MATLAB ® implementations of modeling and transformation algorithms used to morph musical instrument sounds. The SMT is open-source and freely available [13], making it highly flexible, controllable, and customizable by the user. The contribution of this work is the use of a practical example to illustrate the audio processing steps in the SMT to less technically inclined users (such as composers or researchers without the technical background) so these users understand the impact of technical decisions (i.e., parameter values) in the final result. This manuscript is an extended version of [7] presented at CMMR 2019 [14]. The figures have been updated along with the text to provide more detailed information about the SMT and the algorithms within. The next sections take the reader through the audio processing steps involved in morphing with the SMT, which are illustrated with figures and citations to the reference implementations. Section 2 presents an overview of the SMT and the source and target sounds used throughout the rest of the text. Section 3 shows the time-scaling algorithm, Section 4 describes the sinusoidal model used, Section 5 describes parameter interpolation, followed by resynthesis in Section 6 and finally morphing in Section 7.

2 Overview

Figure 1 shows an overview of sound morphing with the SMT. Figure 1 (a) shows the general morphing procedure and Fig. 1 (b) shows the audio processing steps

[13] https://github.com/marcelo-caetano/sound-morphing
[14] https://cmmr2019.prism.cnrs.fr/

in the SMT. The SMT automatically morphs between a source sound S and a target sound T by setting the morphing parameter α that varies between 0 and 1. Only S is heard when $\alpha = 0$, whereas only T is heard when $\alpha = 1$. Intermediate values of α correspond to morphed sounds M with different combinations of S and T. For example, setting $\alpha = 0.5$ produces a morph that is halfway between S and T. Fig. 1 (a) shows that, firstly, S and T are modeled to obtain a parametric representation $\check{S}$ and $\check{T}$. Next, correspondence between $\check{S}$ and $\check{T}$ is established, followed by interpolation and resynthesis.

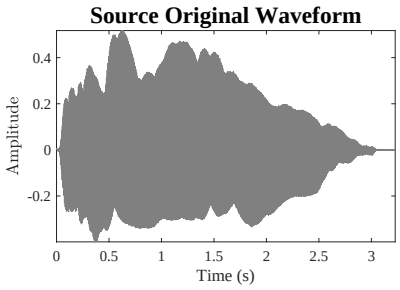
In Fig. 1 (b), we see a representation of the audio processing operations behind these steps in the SMT. First, both S and T are time-scaled to the same duration. Next, the SMT performs sinusoidal analysis of $\check{S}$ and $\check{T}$, producing the sets of parameters $\{A_S, f_S\}$ and $\{A_T, f_T\}$, namely, the amplitudes A and frequencies f of the sinusoids corresponding to $\check{S}$ and $\check{T}$. Correspondence in the SMT requires *partial tracking* and only the *harmonics* are interpolated because S and T are assumed to be nearly harmonic musical instrument sounds. The SMT establishes correspondence between harmonics of the same order and interpolates the amplitudes A and frequencies f using α to obtain $\{A_M, f_M\}$, which are used to synthesize the morphed sound M.

2.1 Source and Target Sounds

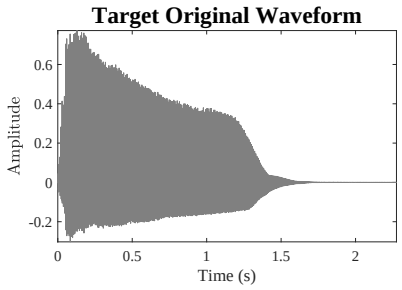
In what follows, the signal processing steps in the SMT corresponding to Fig. 1 (b) are explained and illustrated with an example for $\alpha = 0.5$. Figure 2 shows S on the left column and T on the right column used throughout the rest of the text. The top row of Fig. 2 shows the waveforms, the middle row shows the spectrograms, and the bottom row shows a zoomed-in segment of the waveform to highlight the periodicity of S, a C#3 note played *forte* on an accordion, and of T, a C3 note played *fortissimo* on a tuba. The text contains instructions to listen to sounds corresponding to specific figures. Listen to *source_orig.wav* for Fig. 2 (a) and to *target_orig.wav* for Fig. 2 (b).

3 Time-Scale Modification (TSM)

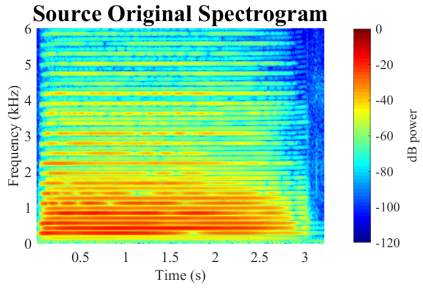
The first step is to use time-scale modification (TSM) [11] to establish temporal correspondence between S and T, which simply guarantees that both have the same duration. In the SMT, the TSM algorithm implemented is *synchronized overlap-add with fixed synthesis* (SOLA-FS) [19]. SOLA-FS uses local waveform similarity with an adaptable analysis step and a fixed synthesis step (see [19] for details). Figure 3 illustrates the result of performing TSM with the SOLA-FS algorithm by comparing the original waveforms against two transformations, namely a time stretch by a factor of two and a time compression by the same factor, which can be expressed as $1/2$. The purpose of Fig. 3 is to show that the waveforms are preserved whether the duration is doubled or halved. More importantly, with the exception of the sound duration and other time-varying attributes such as the attack time, *tremolo* and *vibrato*, the overall sound quality



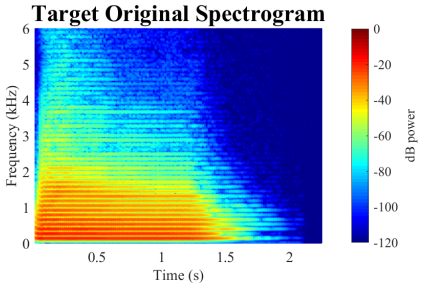
(a) Waveform.



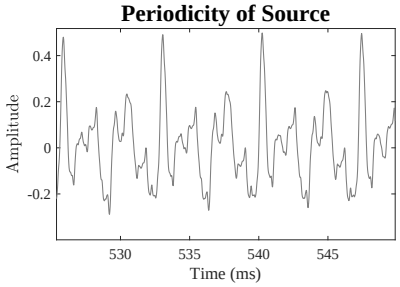
(b) Waveform.



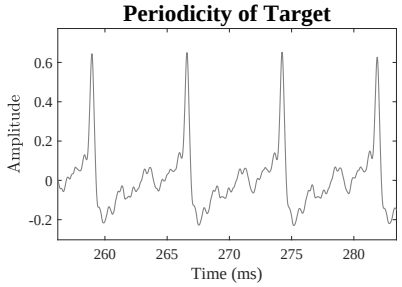
(c) Spectrogram.



(d) Spectrogram.

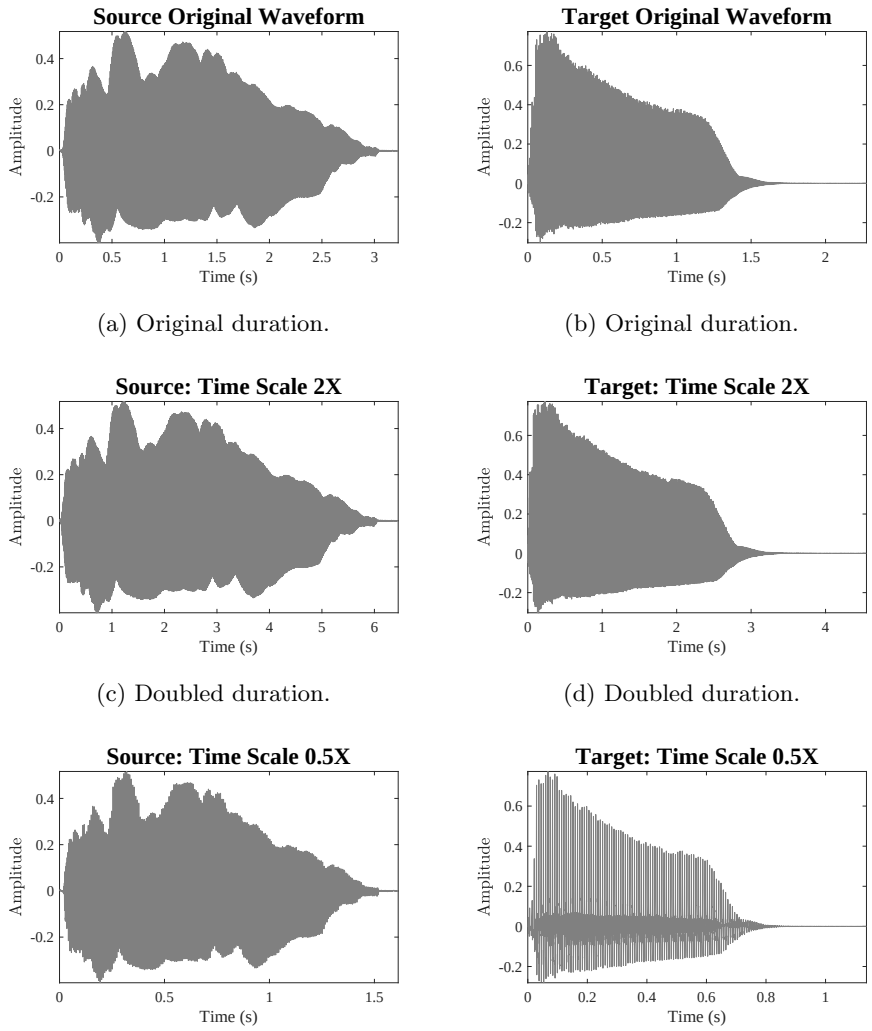


(e) Periodicity.



(f) Periodicity.

Fig. 2: Source (Accordion C#3 *forte*) and Target (Tuba C3 *fortissimo*) sounds used throughout the text to exemplify the audio processing steps in the SMT.



(a) Original duration.

(b) Original duration.

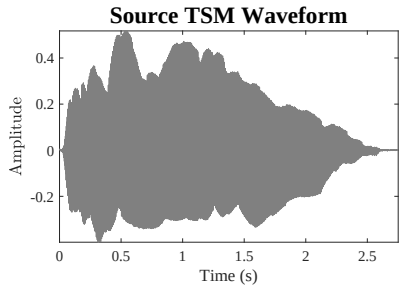
(c) Doubled duration.

(d) Doubled duration.

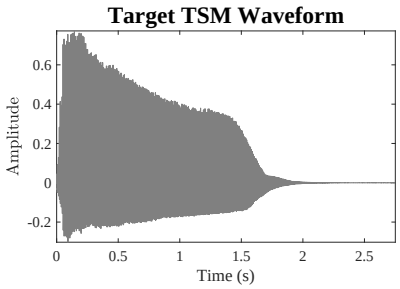
(e) Halved duration.

(f) Halved duration.

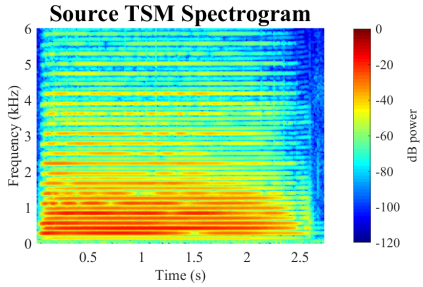
Fig. 3: Illustration of the SOLA-FS time-scale modification algorithm.



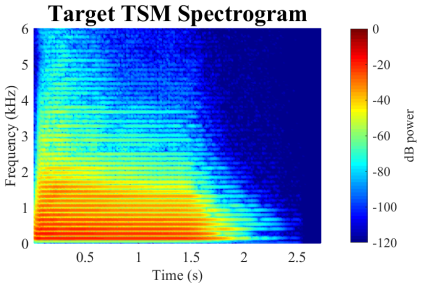
(a) Time-scaled source sound.



(b) Time-scaled target sound.



(c) Time-scaled source sound.



(d) Time-scaled target sound.

Fig. 4: Time-scaled versions $\check{S}$ of the source sound and $\check{T}$ of the target sound.

is preserved after the transformations. Listen to the sounds *source_tsm2X.wav* and *target_tsm2X.wav* for the doubled sounds, and *source_tsm05X.wav* and *target_tsm05X.wav* for the halved sounds.

In the SMT, the morphing parameter α sets the final duration of the morphed sound (see [6] for details). Here, $\alpha = 0.5$ so the duration of M will be halfway between that of S and T, as shown in Fig. 4. Figures 4 (a) and 4 (b) show $\check{S}$ and $\check{T}$ respectively, which are S and T time-scaled. Note that the duration of both $\check{S}$ and $\check{T}$ is the same. Listen to *source_tsm_alpha.wav* and *target_tsm_alpha.wav* and compare with the original sounds. The next step is to use sinusoidal modeling (SM) to represent the oscillatory modes of $\check{S}$ and $\check{T}$.

4 Sinusoidal Modeling (SM)

Currently, the SMT represents musical instrument sounds with the SM, which models a waveform as a sum of time-varying sinusoids parameterized by their amplitudes A, frequencies f, and phases θ [21,26]. The time-varying sinusoids, called partials, represent how the oscillatory modes of the musical instrument change with time, resulting in a flexible representation with perceptually meaningful parameters. The parameters completely describe each partial, which can

be manipulated independently. So the original waveform is represented by the set of time-varying A, f, and θ for each partial, greatly reducing the amount of information required to represent it. The estimation of parameters is called sinusoidal analysis and the process of recreating a waveform from the parameters of sinusoidal analysis is called sinusoidal resynthesis. After the analysis step, sound transformations can be performed as changes of the parameter values prior to resynthesis. In what follows, the mathematical formalization of the SM is described.

The waveform of a sound is represented as $s(n)$, where n is the sample index and there are L samples in total. Then, sound $s(n)$ is divided into frames with a window function $w(n)$ with D samples, so D is the length of the frame and usually $D \leq L$. Inside each frame m, the waveform $s(n,m) = w(n-m)\,s(n)$ is $s(n)$ seen through $w(n-m)$, where m is the integer number of samples by which the center of the window $w(n)$ is shifted. The SM [21,26] assumes that, inside each frame m, $s(n,m)$ can be modeled as

$$s(n,m) = w(n-m)\sum_{q=1}^{Q} A_q(n,m)\,\cos\theta_q(n,m) + e(n,m)\,, \tag{1}$$

where Q is the number of sinusoids, $A_q(n,m)$ is the time-varying amplitude and $\theta_q(n,m)$ is the time-varying phase of sinusoid q, and $e(n,m)$ is the *modeling error* or *residual*. Assuming that $s(n,m)$ is relatively stationary inside each frame m, eq.(1) can be written as

$$s(n,m) = w(n-m)\sum_{q=1}^{Q} A_q\,\cos(\omega_q\, n + \phi) + e(n,m)\,, \quad \text{with } \omega_q = 2\pi\frac{f_q}{f_s}, \tag{2}$$

where f_q is the frequency in Hz and f_s is the sampling frequency in samples per second. Therefore, inside stationary frames, each time-varying sinusoid q can be approximated by a stationary sinusoid with constant amplitude A_q and linear phase $\omega_q n + \phi$.

4.1 Sinusoidal Analysis

The aim of sinusoidal analysis is to use a set of time-varying sinusoids to represent the waveforms $\check{S}$ and $\check{T}$ when added together (hence the name *additive synthesis* [29]). In the SMT, sinusoidal analysis uses spectral modeling [21,26], which comprises *peak picking* and *parameter estimation*. Prior to sinusoidal analysis, the short-time Fourier transform (STFT) is calculated as the discrete Fourier transform (DFT) of each frame m of $s(n,m)$ [24]. For each of these frames, the peaks of the magnitude spectrum (peak picking) are associated with underlying sinusoids whose parameters are estimated (parameter estimation) and later connected across frames in the *partial tracking* step. The DFT $s(n,m)$ from eq.(2) is

$$S(k,m) = \sum_{q=1}^{Q} A_q\, W(\omega_k - \omega_q) + E(\omega)\,, \tag{3}$$

where $W(\omega_k)$ is the DFT of the window $w(n)$ and $E(\omega)$ is the DFT of the modeling error $e(n,m)$. Therefore, in the frequency domain, each sinusoid appears as the DFT of the window $w(n)$ scaled in amplitude by A_q and shifted in frequency by ω_q. The magnitude of $W(\omega_k)$ has a main lobe used to estimate A_q and ω_q and side lobes that introduce spectral distortion and estimation bias [17].

An important requirement to estimate A_q and ω_q using eq. (3) is *spectral resolution* [17,26]. Spectral resolution is the requirement that each individual sinusoid appear as an independent spectral peak. Two sinusoids with frequencies f_1 and f_2 can be individually resolved in the magnitude spectrum if the frame size D obeys

$$D \geq B\frac{f_s}{f_1 - f_2}, \tag{4}$$

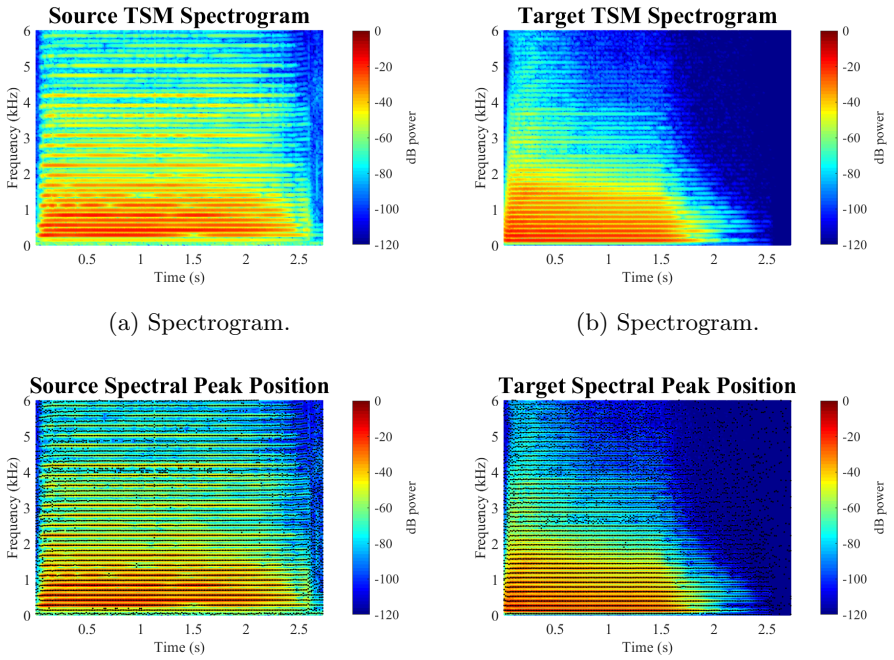
where D is the length of the analysis window in samples, f_s is the sampling frequency in samples per second, and B is the bandwidth of the window, or simply the width of the main lobe in samples (see [17] for more information and tabulated values for several commonly used analysis windows). In practice, D must be large enough to ensure spectral resolution. Nearly harmonically related sinusoids have the property that $f_h \approx hf_0$, where f_0 is the fundamental frequency in Hertz and h is an integer that defines the partial number. Thus, adjacent harmonics are separated by f_0 and the condition in eq. (4) becomes

$$D \geq B\frac{f_s}{f_0}. \tag{5}$$

For the Hann window, $B = 4$ bins [17]. In the SMT, the fundamental frequency f_0 is estimated with SWIPE [9]. In the example, C3 $\approx$ 131 Hz and C#3 $\approx$ 138 Hz and $f_s = 44.1$ kHz, so $D = \max\{1279, 1347\}$ samples. The size of the DFT was $N = 2048$ (the power of two immediately greater than D, achieved by zero padding), and the hop size $H = {}^{D}/_{2}$ (50% overlap).

Peak Picking In practice [21,26], A_q and ω_q are estimated from peaks in the magnitude spectrum $|S(k)|$, where the frame index m has been omitted to simplify the notation. A peak is a local maximum of the magnitude spectrum [26], defined as a sample of the DFT spectrum whose magnitude is greater than both its immediate neighbors. At DFT bin k, $|S(k)|$ is a peak if $|S(k-1)| < |S(k)| > |S(k+1)|$. Figure 5 illustrates the peak-picking algorithm in the SMT [21,26]. The top row shows the spectrogram of $\check{S}$ and $\check{T}$ and the bottom row shows the position of the spectral peaks (i.e., their frequencies f) as dots on top of the spectrogram. Inside each frame m, the SMT returns the $P_{\max}$ spectral peaks with the highest amplitude, so $P_{\max}$ sets the *maximum number of peaks* per frame. In Fig. 5 (c) and Fig. 5 (d), $P_{\max} = 80$.

Peak Selection After peak picking, the information used to represent the magnitude of the STFT is greatly reduced from N frequency bins per frame to $P_{\max} = 80$ peaks per frame. This can be visually confirmed by comparison of



(a) Spectrogram.

(b) Spectrogram.

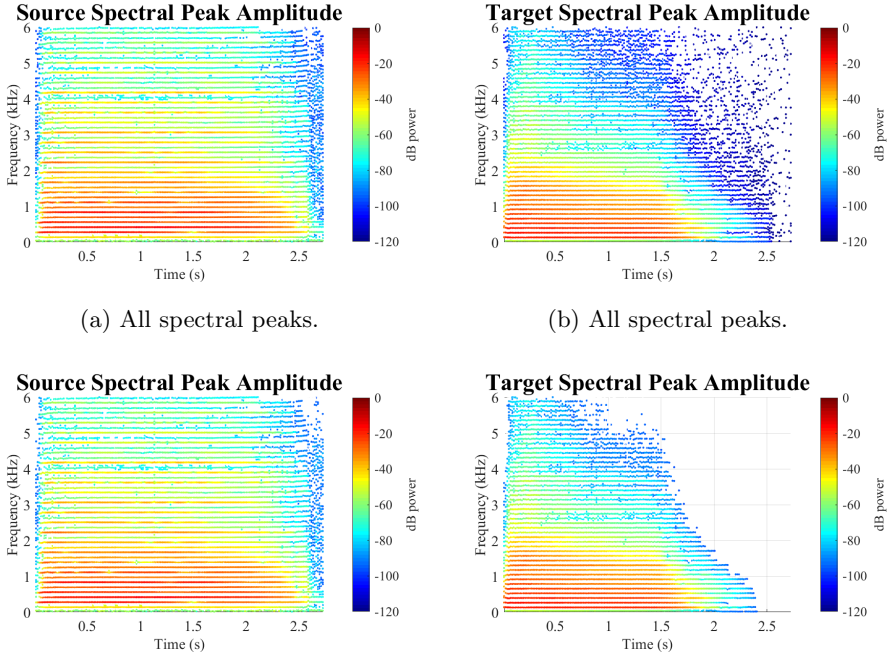
(c) Peak picking.

(d) Peak picking.

Fig. 5: Illustration of the peak picking algorithm.

Fig. 5 with Fig. 6. The spectrogram shown in Fig. 5 (a) and Fig. 5 (b) is reduced to the spectral peaks shown in Fig. 6 (a) and Fig. 6 (b) as the peak frequencies f with their corresponding amplitudes A. The spectrogram of Fig. 5 illustrates the magnitude of the STFT. When both the magnitude and the phase (or equivalently the real part and the imaginary part) of the STFT are used, the STFT can be inverted to recover the original waveform (i.e., the forward and inverse STFT form an identity transform pair [24]). When only the spectral peaks are kept, the rest of the information is lost. Fig. 6 only illustrates the peaks of the magnitude spectrum, but Sec. **Parameter Estimation** below explains how the corresponding phase values are retrieved. Section 6 explores in more detail how to resynthesize a waveform using only spectral peaks with the SM. At this point, we will assess the perceptual impact of removing all this information from the STFT. Listen to *source_sin_allpeak_synthPI.wav* and *target_sin_allpeak_synthPI.wav* and compare with the original sounds.

Additionally, the SMT allows to further reduce the information in the representation with two different thresholds that set the minimum amplitude level (in dB) of the selected peaks. Across all frames, peaks below the *absolute threshold* ϱ are removed. Inside each frame, the *relative threshold* ρ removes peaks whose amplitude is ρ below the maximum level of the frame. In Figs. 6 (c) and 6 (d),



(a) All spectral peaks.

(b) All spectral peaks.

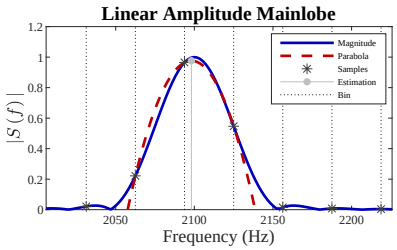
(c) Selected spectral peaks.

(d) Selected spectral peaks.

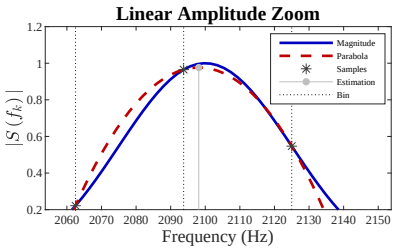
Fig. 6: Illustration of the peak selection step.

$\rho = -76$ dB and $\varrho = -96$ dB. Visual comparison of Figs. 6 (a) and 6 (b) shows the difference. Note, however, that there is virtually no perceptual difference. Listen to *source_sin_thres_synthPI.wav* and *target_sin_thres_synthPI.wav* and compare first with *source_sin_allpeak_synthPI.wav* and *target_sin_allpeak_synthPI.wav* and then with the original sounds.

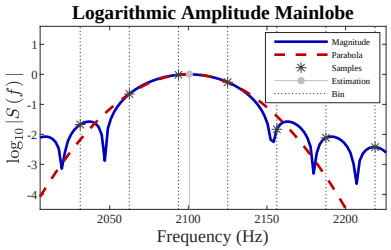
Parameter Estimation In the SMT, the values of the parameters of the SM (A, f, and θ) are estimated using either *nearest-neighbor* estimation [21] or refined by interpolation. The estimation of the amplitudes A and frequencies f is refined by *quadratic interpolation* [29,26] of the peaks of the magnitude spectrum over a *linear* [26], a *logarithmic* [29], or a *power* [31] scale. The estimation of the phase θ uses *linear interpolation* [29] over the unwrapped phase spectrum. Figure 7 illustrates quadratic interpolation, which fits a parabola to each spectral peak in the magnitude spectrum to refine the estimation of A and f. Linear interpolation is shown in Fig. 8 as fitting a straight line to the unwrapped phase spectrum to refine the estimation of θ at the refined frequency estimation f.



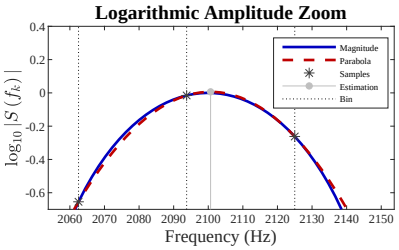
(a) Linear scaling.



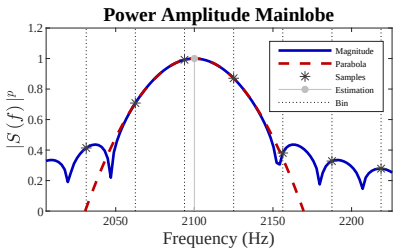
(b) Linear scaling.



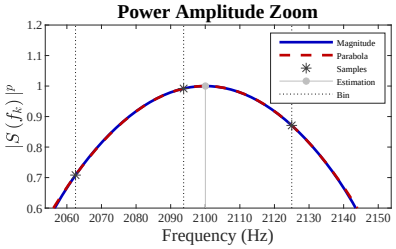
(c) Log scaling.



(d) Log scaling.



(e) Power scaling.



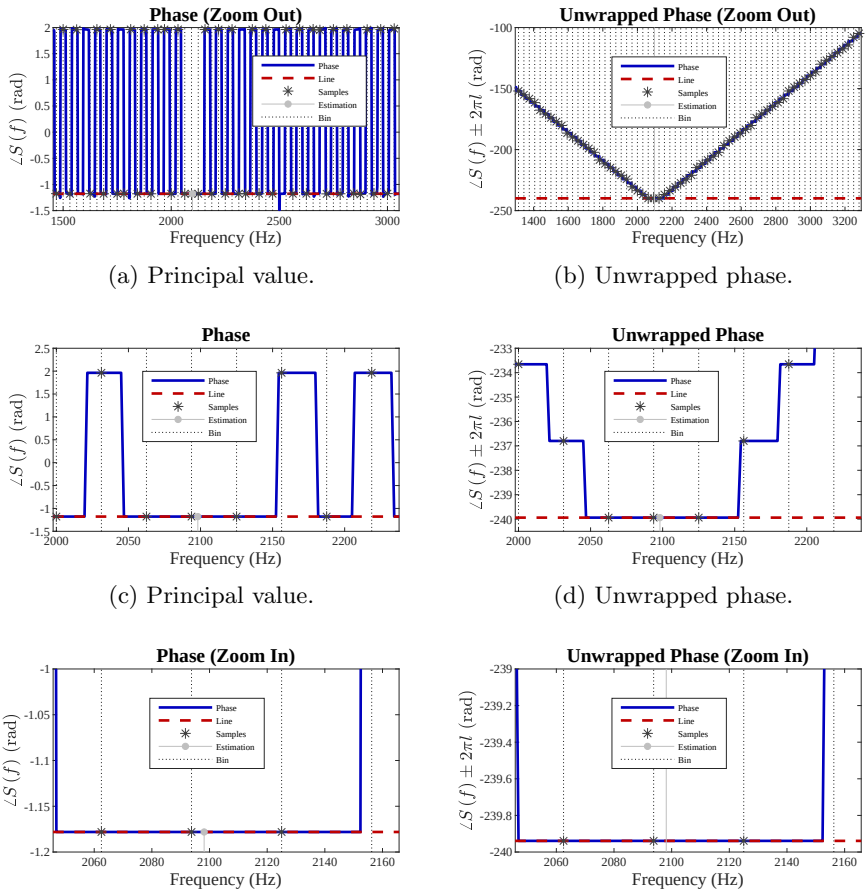
(f) Power scaling.

Fig. 7: Illustration of quadratic interpolation of amplitude and frequency. The figure shows the effect of scaling the magnitude spectrum in the estimation of the amplitudes and frequencies of the underlying sinusoids.

Magnitude Scaling Figure 7 shows the main lobe of the Hann window [17] (zoomed-in on the right) modulated by a sinusoid with $A_q = 1$ and $f_q = 2100\,\text{Hz}$ under linear magnitude scaling in 7 (a) and 7 (b), log magnitude scaling in 7 (c) and 7 (d), and power magnitude scaling in 7 (e) and 7 (f). The column on the left of Fig. 7 shows that the shape of the main lobe of $W(\omega_k)$ changes radically under these different scalings. The column on the right emphasizes how the parabolic fit changes under the different scales. In Fig. 7, the solid line represents the main lobe of the window (in a continuous frequency representation such as the discrete-time Fourier transform), the vertical dotted lines mark the position of the frequency bins k, and the DFT samples are illustrated as * where the bins k cross the main lobe. The spectral peak will be notated as $\hat{S}(k)$ to simplify the notation. Nearest-neighbor estimation [21] uses $A_q = \hat{S}(k)$ and $f_q = \frac{k}{N} f_s$, where N is the size of the DFT. In practice, the frequency bin corresponding to the spectral peak is used, as illustrated in Fig. 7 by *. The DFT samples are discrete frequency values $f_s/N\,\text{Hz}$ apart whereas the analyzed frequency f_q is continuous. In practice, f_q can be anywhere between two samples of the DFT, resulting in *spectral leakage*, which is basically the appearance of a spectral peak in more that one bin of the DFT (the lobes in Fig. 7 are typical illustrations of spectral leakage). Consequently, nearest-neighbor estimation leads to estimation errors of up to half a bin [26,31] with the DFT. The SMT uses quadratic interpolation of spectral peaks to address the bias inherent in nearest neighbor estimation [1].

As shown in Fig. 7, quadratic interpolation fits a parabola (dashed line in Fig. 7) to $[k-1, k, k+1]$ and $\left[\hat{S}(k-1), \hat{S}(k), \hat{S}(k+1)\right]$. Quadratic interpolation improves the accuracy of stimation of both A_q and f_q by using the vertex of the parabola as refined estimation. Linear scaling uses $\hat{S}(k) = |S(k)|$, log scaling uses $\hat{S}(k) = \log_{10}|S(k)|$, and power scaling uses $\hat{S}(k) = |S(k)|^p$. Figure 7 (c) shows that the parabola fits the log-scaled main lobe better than the linear case in Fig. 7 (a). However, there is still frequency and amplitude estimation error. Figure 7 (e) visually confirms the finding that power scaling improves the fit over log scaling [31]. However, power scaling is currently limited to $D = N$ with N being a power of two, which limits the selection of D. The example in this article uses log magnitude scaling and selection of D according to eq. (5).

Phase Unwrapping Figure 8 shows the phase spectrum of the modulated zero-phase Hann window from Fig. 7. The left column of Fig. 8 shows the principal value of the phase $-\pi < \vartheta \leq \pi$ and the right column of Fig. 8 shows the unwrapped phase $\theta = \vartheta \pm 2\pi l$, where $l \in \mathbb{N}$. The solid line in Fig. 8 represents the phase spectrum (principal value and unwrapped), the vertical dotted lines mark the position of the frequency bins k, the DFT samples are illustrated as *, and the dashed line shows the linear fit. As Fig. 8 (a) shows, the principal value of the phase is discontinuous because it is confined to the interval $-\pi < \vartheta \leq \pi$. Phase unwrapping, shown in Fig. 8 (b), corrects the phase ϑ by adding multiples of $\pm 2\pi$ whenever the discontinuity ξ is $\xi \geq \pi$. Note that there is no linear phase component because the analysis window is zero phase [25], so the effect of phase unwrapping is equivalent to keeping track of complete cycles. Figure 8 (c) and



(a) Principal value. (b) Unwrapped phase.

(c) Principal value. (d) Unwrapped phase.

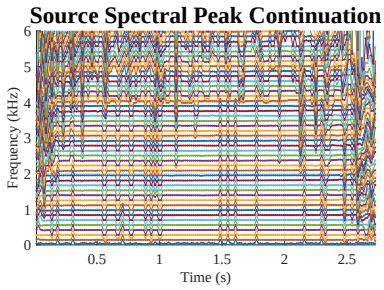
(e) Principal value. (f) Unwrapped phase.

Fig. 8: Illustration of linear interpolation of phase. The figure shows the effect of unwrapping the phase in the estimation of the phases of the underlying sinusoids.

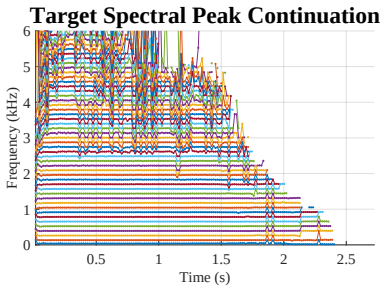
Fig. 8 (d) show the phase spectrum around the estimated frequency $f_0 \approx 2100\,\text{Hz}$ and both Fig. 8 (e) and Fig. 8 (f) zoom in to emphasize the fit of the straight dashed line. Linear interpolation does not seem to visually improve the estimation of phase over nearest-neighbor estimation. However, phase unwrapping is important for an accurate representation of the phase θ.

At this point, the sounds $\check{S}$ and $\check{T}$ are represented by the set of parameters A, f, and θ resulting from sinusoidal analysis. Each sound was divided into overlapping frames, and then the DFT of these frames is calculated (i.e., the STFT). For each frame m, peaks of the magnitude spectrum are associated with underlying sinusoids whose parameters $A_q(m)$, $f_q(m)$, and $\theta_q(m)$ are estimated. Across the frames m, sinusoids with time-varying amplitude $A_q(n)$ and phase $\theta_q(n)$ can be synthesized and added to create the sinusoidal component of $\check{S}$ and $\check{T}$ separately as described in [21]. Section 6.1 shows the resultant sinusoidal component as well as the modeling error $e(n)$ from eq. (1), referred to as resynthesis by parameter interpolation (PI). It is important to note that only the time-varying amplitudes and phases are used in PI resynthesis. The frequencies are used to reconstruct the time-varying phase with the aid of the estimated $\theta_q(m)$ used as anchors [21]. However, as Fig. 1 illustrates, morphing requires a single set of intermediate parameters that would result in a perceptually intermediate sound upon resynthesis. Morphing with the SM is achieved by interpolating A and f from $\check{S}$ and $\check{T}$ across time. However, the anchors $\theta_q(m)$ cannot be interpolated because the interpolated time-varying phase would not correspond to the interpolated time-varying frequency [3,4,23], so the mismatch between temporal variation of phase and frequency would result in audible artifacts. Therefore, the estimated phase values θ from $\check{S}$ and $\check{T}$ are discarded in the SMT. Section 6.2 provides details on how the morph M is resynthesized via phase reconstruction by frequency interpolation [20].

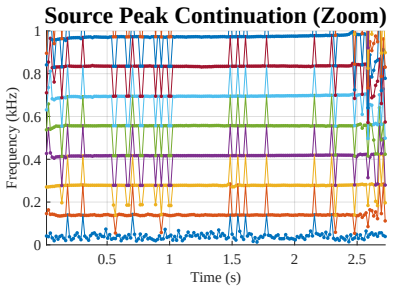
Partial Tracking The spectral peaks returned from the peak-picking step (after further parameter estimation) do not result in a set of *continuous* partials because there is no mechanism to ensure temporal continuity. Figure 9 shows the final peaks returned from the parameter estimation step connected by lines. Inside each frame, spurious spectral peaks appear and later disappear (due mainly to interference by nearby peaks and sidelobe interaction), resulting in the discontinuous "spectral lines" in Figs. 9 (c) and 9 (d). The SMT uses a *partial tracking* algorithm to convert the discontinuous spectral lines seen in Fig. 9 into the continuous partial tracks shown in Fig. 10. The partial tracking algorithm implemented in the SMT is based on the peak continuation algorithm proposed by McAuley and Quatieri [21], so it simply collects peaks within a frequency threshold Δ_p into continuous tracks. In Fig. 10, $\Delta_p = f_0/4$, where f_0 is the fundamental frequency of $\check{S}$ or $\check{T}$. Note the difference between Figs. 9 and 10, especially the zoomed-in panels on the bottom. After partial tracking, the partials present continuous temporal trajectories, seen as fairly straight horizontal lines across. Once again we might want to assess the perceptual impact of the partial tracking algorithm. Listen to *source_sin_partrack_synthPI.wav* and *tar-*



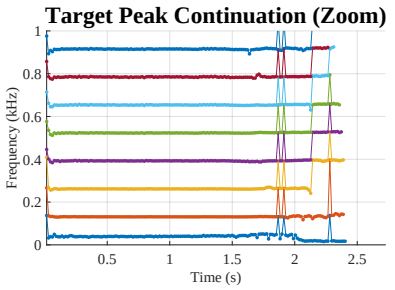
(a) Spectral peak continuation.



(b) Spectral peak continuation.



(c) Spectral peak continuation.



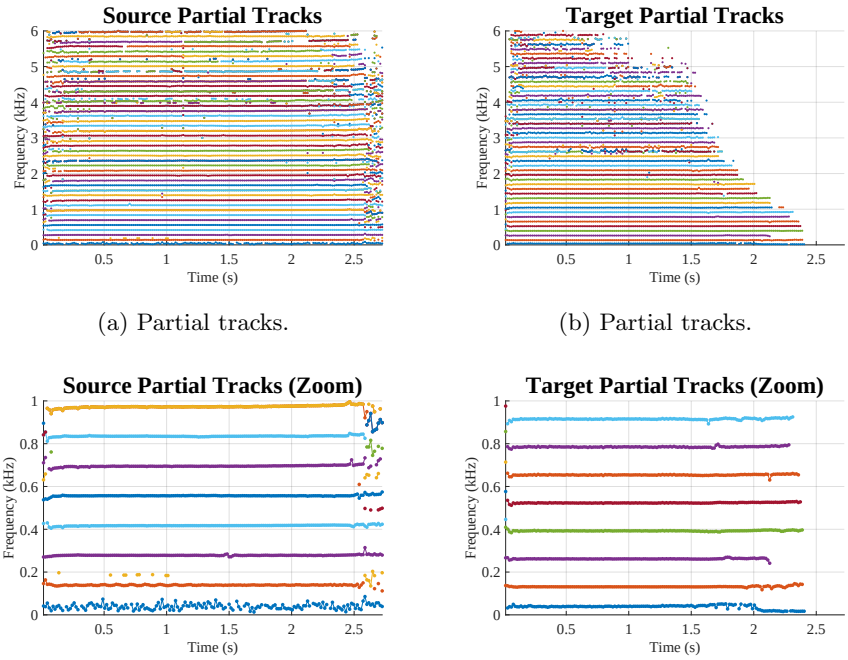
(d) Spectral peak continuation.

Fig. 9: Spectral peak continuation prior to partial tracking. The top row shows the spectral peaks connected by lines to illustrate the temporal discontinuity. The bottom part shows a zoomed-in part of the top row.

get_sin_partrack_synthPI.wav to hear the result of resynthesis using the original phase θ and all the partial tracks. Compare with *source_sin_thres_synthPI.wav* and *target_sin_thres_synthPI.wav*, which use essentially the same information (i.e., the selected spectral peaks) but not yet organized into partial tracks.

4.2 Harmonic Selection

The next step after partial tracking is to select the harmonics of the fundamental frequency f_0. The *harmonic selection* step eliminates mainly the partials resulting from spurious frequency peaks while keeping the harmonically related partials, called *harmonics*. In the SMT, harmonics are the partials whose median frequencies over time are harmonically related to the fundamental f_0 within an interval Δ_h. This nearly harmonic relation can be expressed as $f_h = hf_0 \pm \Delta_h$, where f_h is the harmonic of order h. Figure. 11 shows the result of harmonic selection on the partial tracks from Fig. 10 with $\Delta_h = 10\,\text{Hz}$. Listen to *source_sin_harm_synthPI.wav* and *target_sin_harm_synthPI.wav* to hear the result of resynthesis using the original phase and only the harmonics.



(a) Partial tracks.

(b) Partial tracks.

(c) Partial tracks.

(d) Partial tracks.

Fig. 10: The result of partial tracking. The top row shows the spectral peaks from Fig. 9 reorganized as partial tracks. The partial tracking algorithm ensures temporal continuity, as illustrated by the zoomed-in regions on the bottom row.

5 Interpolation

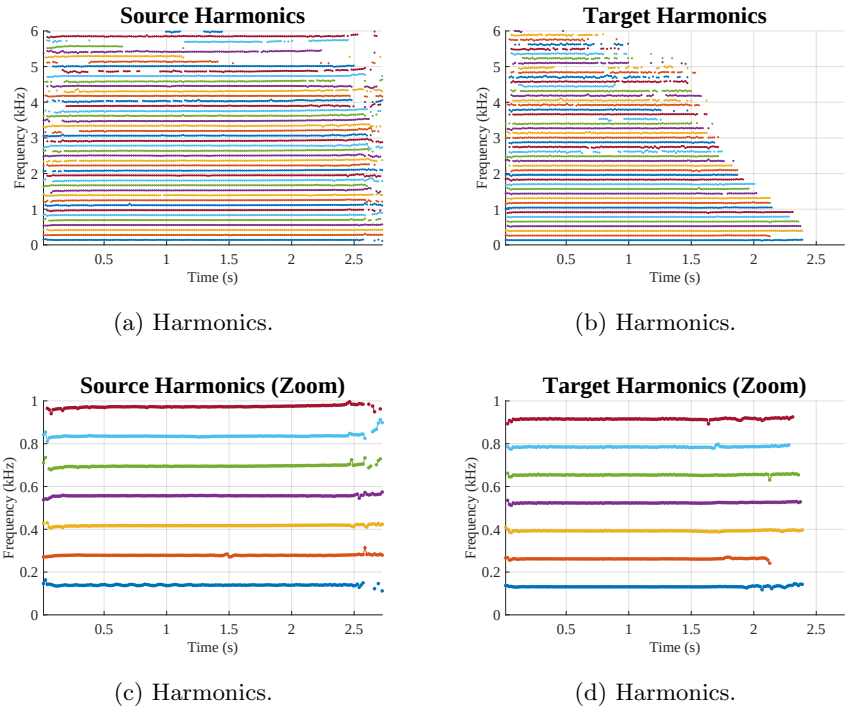
Prior to interpolation, the SMC establishes correspondence between harmonics using the harmonic number h. Then, the frequencies are interpolated in cents (see [6] for details) and the amplitudes can be interpolated linearly or in decibels. The interval in cents c between two frequencies f_1 and f_2 is $c = 1200 \log_2 (f_1/f_2)$, so an intermediate frequency f_α is given by

$$f_\alpha = f_1 2^{(1-\alpha) \log_2 \left(\frac{f_2}{f_1}\right)}. \tag{6}$$

Equivalently, f_α can be obtained as

$$f_\alpha = f_2 2^{\alpha \log_2 \left(\frac{f_1}{f_2}\right)}. \tag{7}$$

Both equations (6) and (7) yield the same value for f_α, so either one may be used. Naturally, they can also be combined as in [7]. Similarly, for the amplitudes, the interval in dB between A_1 and A_2 is $\mathrm{dB} = 10 \log_{10} (A_1/A_2)$ and an intermediate



(a) Harmonics. (b) Harmonics.

(c) Harmonics. (d) Harmonics.

Fig. 11: Only the partials whose frequencies are nearly harmonically related to the fundamental remain. Compare with Fig. 10. See text for details.

amplitude A_α is given by

$$A_\alpha = A_1 10^{(1-\alpha)\log_{10}\left(\frac{A_2}{A_1}\right)} \tag{8}$$

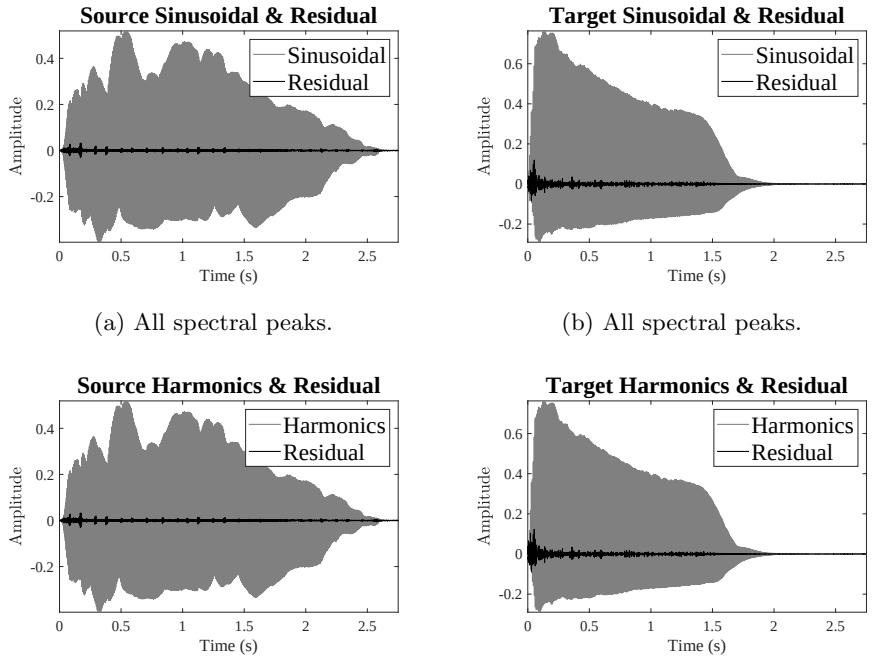
or

$$A_\alpha = A_2 10^{\alpha\log_{10}\left(\frac{A_1}{A_2}\right)}. \tag{9}$$

Again, these expressions are equivalent and can also be combined as in [7]. Linear interpolation of amplitudes is achieved as $A_\alpha = \alpha A_1 + (1-\alpha)A_2$. In the example, logarithmic interpolation of amplitudes from eq. (8) was used. After all frequencies and amplitudes have been interpolated, the set of interpolated harmonics is resynthesized to obtain the final morph.

6 Resynthesis

Currently, the SMT has three resynthesis methods implemented, namely overlap-add (OLA) [15,14], parameter interpolation (PI) [21], and phase reconstruction by frequency integration (PRFI) [20]. Both OLA and PI require the phases θ



(a) All spectral peaks.

(b) All spectral peaks.

(c) Only harmonics.

(d) Only harmonics.

Fig. 12: Source sound is Accordion C#3 *forte* and Target Sound is Tuba C3 *fortissimo*

along with A and f to re-synthesize a waveform that is similar to the original both objectively and perceptually [29]. PRFI reconstructs the phase by integrating the frequency tracks across time [20], resulting in a waveform that is objectively different from the original but perceptually similar [29]. As previously stated in Section 4.1, the sinusoidal morphing procedure currently implemented in the SMT only interpolates A and f, so PRFI is used to synthesize the morph M. PRFI results in waveforms that look different than the original but sound similar. The next section explores further the consequences of PI resynthesis with the original phase and PRFI resynthesis with phase reconstruction.

6.1 Resynthesis with the Original Phase

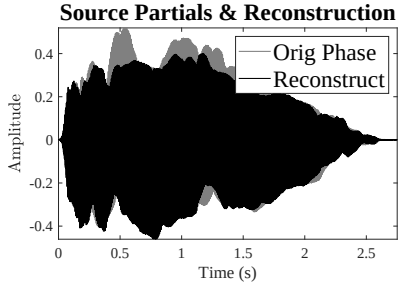
Figure 12 shows a comparison of the waveforms of S (on the left) and T (on the right) resynthesized with the original phase θ (PI resynthesis) using all spectral peaks (top row) or only the harmonics (bottom row). Each panel of Fig. 12 shows both the sinusoidal component (in gray) and the residual (in black). The residual is simply the subtraction of the sinusoidal component from the original waveform. Thus, the residual results from the information in the original wave-

form missed by the sinusoidal component. The energy present in the residual is an indication of how well the sinusoidal model captures the oscillatory behavior of S and T, where the lower the residual energy the better the model. The modeling residual is commonly assumed to be noise that was not captured by the sinusoidal component because sinusoids are not a compact representation of noise [8]. For musical instrument sounds, the residual commonly captures noise from the sound production mechanism such as the hammer striking the strings on the piano, the plectrum plucking the strings on the harpschord, bowing on the violin, blowing into the flute, among many other vibration mechanisms. Mechanical noise is intrinsic to the sounds produced by acoustic musical instruments, so the residual is commonly modeled as filtered white noise and added back into the sinusoidal component. Caetano *et al.* [8] investigated if there is oscillatory energy left in the residual from sinusoidal analysis and concluded that the residual is not perceptually equivalent to filtered white noise, further noting that the differences may lie in the phase spectrum.

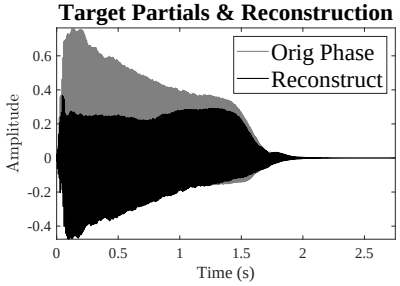
Figure 12 reveals that, visually, the difference between using all spectral peaks and only the harmonics is barely noticeable. Naturally, the perceptual difference is also important, so listening to the sounds might reveal perceptual differences that are not visible in the waveforms shown in Fig. 12. The sinusoidal component of Fig. 12 (a) corresponds to *source_sin_thres_synthPI.wav* and the residual to *source_res_thres_synthPI.wav*. The original residual *source_res_thres_synthPI.wav* is much softer than the sinusoidal component, so *source_res16dB_thres_synth.wav* is the same sound normalized to -16 dB RMS. Similarly, for Fig. 12 (b), the sinusoidal component is *target_sin_thres_synthPI.wav* and the residual is *target_res_thres_synthPI.wav*, normalized in *target_res16dB_thres_synth.wav*. Finally, for the harmonic resynthesis of Fig. 12 (c), listen to the sinusoidal component *source_sin_harm_synthPI.wav* and the residual *source_res_harm_synthPI.wav* (normalized in *source_res16dB_harm_synthPI.wav*). For Fig. 12 (d), listen to the sinusoidal component *target_sin_harm_synthPI.wav* and to the residual component *target_res_harm_synthPI.wav* (normalized in *target_res16dB_harm_synthPI.wav*).

6.2 Resynthesis via Phase Reconstruction

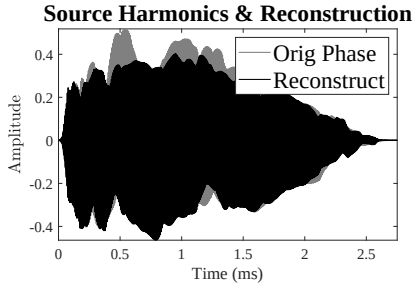
Figure 13 shows a comparison of the resynthesized waveforms of S and T with the original phase θ (PI resynthesis) and via phase reconstruction (PRFI resynthesis). The left column shows the source sound $\check{S}$ and the right column shows the target sound $\check{T}$. The top row shows both $\check{S}$ and $\check{T}$ resynthesized using all spectral peaks and the bottom row shows resynthesis using only the harmonics. In all panels, the grey waveform uses the original phase θ (PI resynthesis) and the black waveform uses PRFI. Figure 13 illustrates the role of the original phase in the SM. For example, Fig. 13 (a) and Fig. 13 (b) show that the resynthesized waveform is different when the original phase θ is discarded and a new phase $\hat{\theta}$ is reconstructed by integrating the time-varying frequencies of the partials [20]. For sound morphing, it is more important to determine if PRFI resynthesis results in a waveform that is perceptually different from the one using the original phase. Listen to *source_sin_thres_synthPI.wav* for the resynthesis



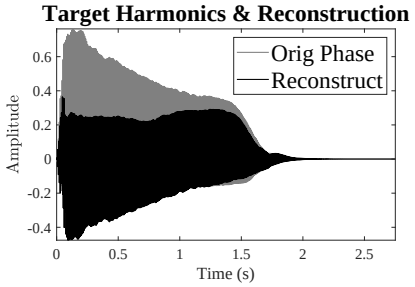
(a) All spectral peaks.



(b) All spectral peaks.



(c) Only harmonics.



(d) Only harmonics.

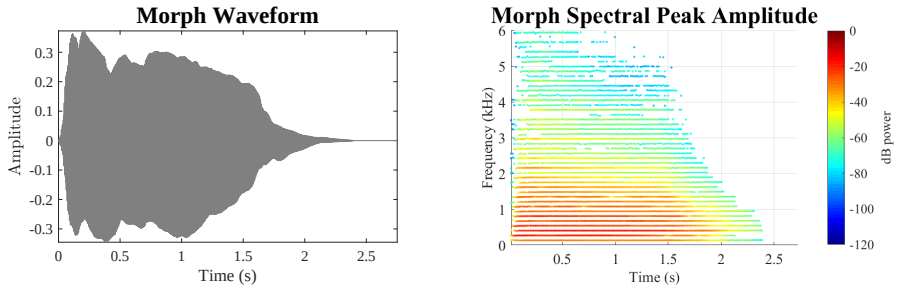
Fig. 13: Comparison of different resynthesis methods in the SMT.

with the original phase and to *source_sin_thres_synthPRFI.wav* for the resynthesis with the reconstructed phase in Fig. 13 (a). Similarly, for Fig. 13 (a), listen to *target_sin_thres_synthPI.wav* and to *target_sin_thres_synthPRFI.wav*.

Additionally, Fig. 13 shows the impact of *harmonic selection* in the SMT. Most pitched musical instruments are designed to present clear modes of vibration that are nearly harmonic [13]. The piano is a notorious exception where the stiffness of the strings results in slightly inharmonic notes [2]. Nevertheless, the majority of pitched acoustic musical instruments produces sounds with most of the oscillatory energy concentrated around harmonics of the fundamental frequency. Therefore, the harmonic selection step is not expected to result in perceptually different sounds than keeping all spectral peaks. Listen to *source_sin_harm_synthPI.wav* and *source_sin_harm_synthPRFI.wav* to compare the waveforms shown in Fig. 13 (c). For the waveforms shown in Fig. 13 (d), listen to *target_sin_harm_synthPI.wav* and to *target_sin_harm_synthPRFI.wav*.

7 Morphing

Finally, the morph is achieved by resynthesizing the set of interpolated parameters M with PRFI, as shown in Fig. 14. Figure 14 (a) shows the waveform and



(a) Waveform of morphed sound. (b) Spectrogram of morphed sound.

Fig. 14: Morphed musical instrument sound. The left-hand side shows the waveform and the right-hand side shows the spectral peaks with their corresponding amplitudes.

Fig. 14 (b) shows the spectral peaks of the morph. Visually, it is more intuitive to use the spectral peaks than the waveforms to confirm that M is indeed intermediate between S and T. For example, it is not visually intuitive that the waveform of Fig. 14 (a) is perceptually halfway between those of Fig. 2 (a) and Fig. 2 (b). However, visual comparison between Fig. 14 and Fig. 6 reveals that the spectral peaks in Fig. 14 (b) correspond to intermediate peaks between those of Fig. 6 (c) and Fig. 6 (d). Naturally, for sound morphing, the perceptual comparison is more important than the visual intuition. Listen to the original sounds *source_orig.wav* and *target_orig.wav* and then to *accordion_tuba_morph_alpha05.wav* to assess if M is indeed perceptually intermediate between S and T.

8 Conclusions and Perspectives

This work has described how to use the Sound Morphing Toolbox (SMT) to morph musical instrument sounds with the sinusoidal model. The audio processing steps were illustrated with figures and citations to the reference implementations. The SMT is open-source and freely available under a GNU3 license. Time-varying morphs [6] will be incorporated into a future version. Future development of the SMT will also add a GUI and an implementation of the hybrid source-filter model and the sophisticated sound morphing algorithm that uses it [5]. Finally, the SMT is currently an *alpha release* with possible bugs in the code due to limited testing. Adoption and use of the SMT by the community is encouraged to provide usability testing and bug corrections that might lead to a *beta release*.

References

1. Abe, M., III, J.O.S.: CQIFFT: Correcting bias in a sinusoidal parameter estimator based on quadratic interpolation of fft magnitude peaks. Tech. Rep. STAN-M-117, Center for Computer Research in Music and Acoustics – Department of Music, Stanford University (2004)
2. Bader, R., Hansen, U.: Modeling of musical instruments. In: Havelock, D., Kuwano, S., Vorländer, M. (eds.) Handbook of Signal Processing in Acoustics, pp. 419–446. Springer New York (2009)
3. Boashash, B.: Estimating and interpreting the instantaneous frequency of a signal. I. Fundamentals. Proceedings of the IEEE **80**(4), 520–538 (1992)
4. Boashash, B.: Estimating and interpreting the instantaneous frequency of a signal. II. Algorithms and applications. Proceedings of the IEEE **80**(4), 540–568 (1992)
5. Caetano, M., Kafentzis, G.P., Mouchtaris, A., Stylianou, Y.: Full-band quasi-harmonic analysis and synthesis of musical instrument sounds with adaptive sinusoids. Appl Sci **6**(5), 127 (2016)
6. Caetano, M., Rodet, X.: Musical instrument sound morphing guided by perceptually motivated features. IEEE Trans. Audio Speech Lang. Process. **21**(8), 1666–1675 (2013)
7. Caetano, M.: Morphing musical instrument sounds with the Sound Morphing Toolbox. In: Proceedings of the 14th International Symposium on Computer Music Interdisciplinary Research, CMMR 2019. pp. 171–182. Marseille, France (October 2019)
8. Caetano, M., Kafentzis, G.P., Degottex, G., Mouchtaris, A., Stylianou, Y.: Evaluating how well filtered white noise models the residual from sinusoidal modeling of musical instrument sounds. In: Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA). pp. 1–4. New Paltz, NY (October 2013)
9. Camacho, A., Harris, J.G.: A sawtooth waveform inspired pitch estimator for speech and music. J Acoust Soc Am **124**(3), 1638–1652 (2008)
10. Carral, S.: Determining the just noticeable difference in timbre through spectral morphing: A trombone example. Acta Acust united Ac **97**, 466–476 (05 2011)
11. Driedger, J., Müller, M.: A review of time-scale modification of music signals. Appl Sci **6**(2) (2016)
12. Fitz, K., Haken, L., Lefvert, S., Champion, C., O'Donnell, M.: Cell-utes and flutter-tongued cats: Sound morphing using loris and the reassigned bandwidth-enhanced model. Comput. Music J **27**(3), 44–65 (Sep 2003)
13. Fletcher, N.H., Rossing, T.D.: The Physics of Musical Instruments. Springer, New York, second edn. (1998)
14. George, E.B., Smith, M.J.T.: Speech analysis/synthesis and modification using an analysis-by-synthesis/overlap-add sinusoidal model. IEEE Trans. Speech Audio Process. **5**(5), 389–406 (Sep 1997)
15. George, E.B., Smith, M.J.T.: Analysis-by-synthesis/overlap-add sinusoidal modeling applied to the analysis and synthesis of musical tones. J. Audio Eng. Soc **40**(6), 497–516 (1992)
16. Grey, J.M., Gordon, J.W.: Perceptual effects of spectral modifications on musical timbres. J Acoust Soc Am **63**(5), 1493–1500 (1978)
17. Harris, F.J.: On the use of windows for harmonic analysis with the discrete Fourier transform. Proceedings of the IEEE **66**(1), 51–83 (Jan 1978)

18. Harvey, J.: "Mortuos Plango, Vivos Voco": A realization at IRCAM. Comput. Music J **5**(4), 22–24 (1981)
19. Hejna, D., Musicus, B.R.: The SOLA-FS time-scale modification algorithm. Tech. rep., BBN (1991)
20. McAulay, R., Quatieri, T.: Magnitude-only reconstruction using a sinusoidal speech model. In: Proc. ICASSP. vol. 9, pp. 441–444 (March 1984)
21. McAulay, R., Quatieri, T.: Speech analysis/synthesis based on a sinusoidal representation. IEEE Trans. Acoust. Speech Signal Process. **34**(4), 744–754 (Aug 1986)
22. McNabb, M.: "Dreamsong": The composition. Comput. Music J **5**(4), 36–53 (1981)
23. Picinbono, B.: On instantaneous amplitude and phase of signals. IEEE Transactions on Signal Processing **45**(3), 552–560 (1997)
24. Portnoff, M.: Implementation of the digital phase vocoder using the fast Fourier transform. IEEE Transactions on Acoustics, Speech, and Signal Processing **24**(3), 243–248 (1976)
25. Serra, X.: Musical Signal Processing, chap. Musical Sound Modeling with Sinusoids plus Noise, pp. 91–122. G. D. Poli, A. Picialli, S. T. Pope, and C. Roads Eds. Swets & Zeitlinger, Lisse, Switzerland (1996)
26. Serra, X., Smith, J.O.: Spectral Modeling Synthesis: A Sound Analysis/Synthesis Based on a Deterministic plus Stochastic Decomposition. Comput. Music J **14**, 12–24 (1990)
27. Siedenburg, K., Jones-Mollerup, K., McAdams, S.: Acoustic and categorical dissimilarity of musical timbre: Evidence from asymmetries between acoustic and chimeric sounds. Front Psychol **6**, 1977 (2016)
28. Slaney, M., Covell, M., Lassiter, B.: Automatic audio morphing. In: Proc. ICASSP. vol. 2, pp. 1001–1004 (May 1996)
29. Smith, J.O., Serra, X.: PARSHL: An analysis/synthesis program for non- harmonic sounds based on a sinusoidal representation. In: Proc. ICMC. pp. 290–297 (1987)
30. Tellman, E., Haken, L., Holloway, B.: Timbre morphing of sounds with unequal numbers of features. J. Audio Eng. Soc **43**(9), 678–689 (1995)
31. Werner, K.J., Germain, F.G.: Sinusoidal parameter estimation using quadratic interpolation around power-scaled magnitude spectrum peaks. Appl Sci **6**(10), 306 (2016)
32. Wishart, T.: On Sonic Art. Routledge, New York, 1 edn. (1996)