PVSOLA: A PHASE VOCODER WITH SYNCHRONIZED OVERLAP-ADD

Alexis Moinet
TCTS Lab.
Faculté polytechnique
University of Mons, Belgium
alexis.moinet@umons.ac.be

Thierry Dutoit
TCTS Lab.
Faculté polytechnique
University of Mons, Belgium
thierry.dutoit@umons.ac.be

ABSTRACT

In this paper we present an original method mixing temporal and spectral processing to reduce the phasiness in the phase vocoder. Phasiness is an inherent artifact of the phase vocoder that appears when a sound is slowed down. The audio is perceived as muffled, reverberant and/or moving away from the microphone. This is due to the loss of coherence between the phases across the bins of the Short-Term Fourier Transform over time. Here the phase vocoder is used almost as usual, except that its phases are regularly reset in order to keep them coherent. Phase reset consists in using a frame from the input signal for synthesis without modifying it. The position of that frame in the output audio is adjusted using cross-correlation, as is done in many temporal time-stretching methods. The method is compared with three state-of-the-art algorithms. The results show a significant improvement over existing processes although some test samples present artifacts.

1. INTRODUCTION

Time-stretching of an audio signal is a process that increases or reduces the length of the signal while preserving its acoustic quality. In other words it reduces or increases the playback speed of the sound without changing its perceived content, as opposed to a change of the sampling frequency that causes a downward or upward frequency shift.

Many algorithms have been developed to achieve such a transformation. They generally belong to one of three categories [1]: time-domain, frequency-domain and model-based algorithms, although some methods combine several approaches (time and frequency, frequency and model).

Time-domain methods such as SOLA (*synchronized overlap-add*), WSOLA (*waveform similarity-based synchronized overlap-add*), SOLAFS (*synchronized overlap-add, fixed synthesis*), TD-PSOLA (*time-domain pitch-synchronous overlap-add*) [2, 3, 4] and their variants are usually applied to monophonic signals, for instance speech and singing recordings. The basic principle of these methods is to segment the signal into overlapping *frames* (i.e. blocks of consecutive audio samples) and either duplicate (drop) some frames or increase (reduce) the shift between each frame, in order to extend (compress) the duration of the signal.

Frequency or spectral-domain algorithms are most often based on the phase vocoder [5]. Compared to time-domain approaches, the phase vocoder has the advantage to work with both mono and polyphonic signals. Besides it theoretically overlaps frames perfectly in phase with each other. However in practice it produces a sound that can be perceived as muffled, reverberant and/or moving away from the microphone [6, 7]. This distortion is called *phasiness* [8] and the accepted explanation for its presence is a loss of coherence between the phases across the bins of the Short-Term Fourier Transform over time, also called loss of vertical phase coherence. Different methods have been proposed in order to attenuate this artifact in [6, 7, 9].

Model-based approaches transform the audio signal into a set of frame-adaptive parameters that are decimated or interpolated to synthesize a time-scaled version of the sound. Linear Prediction-based analysis/synthesis, Harmonic plus Noise Model [10], Spectral Modeling Synthesis [11] and Sine + Transient + Noise Model [12] are good examples.

Some methods combine several approaches, as an enhanced version of SOLA [13] where a phase vocoder is used to modify the phases of each frame so that they overlap properly instead of adapting their position in the output audio signal. Another example is [14] which concatenates groups of time-domain frames with groups of frames generated by the phase vocoder. Besides STRAIGHT [15] could be considered as a mixed method to a certain extent.

In this paper we propose a new approach where a SOLA-like algorithm is used to periodically adapt the position of some frames in a phase vocoder (as opposed to using a phase vocoder to adapt the frames of SOLA in [13]). These frames are analysis frames used without phase modification which in turn causes a phase reset of the vocoder. This reduces the phasiness observed in audio signals without requiring any phase locking. We named this method PVSOLA (*Phase Vocoder with Synchronized Overlap Add*).

Phase reset or time-domain frame insertion has already been introduced by Karrer [16], Röbel [17] and Doran et al. [14]. Karrer resets the phases of the vocoder during silent parts, so that the distortion that it might cause is inaudible. Röbel preserves the transient components of a signal by resetting the phase-vocoder whenever a transient event is detected. Doran et al. do not abruptly reset the vocoder, instead they progressively alter the phases of the synthesis frames in order to regain coherency with the input signal. When the output and input signal become eventually in phase, a group of frames from the input is directly inserted in the output which is equivalent to a reset of the phase vocoder.

We review the principle of an STFT-based phase vocoder in Section 2 with the description of two possible approaches and different phase locking methods. Then we introduce an implementation of our method in Section 3 and we discuss its results and future developments in Sections 4 and 5.

This work is supported by a public-private partnership between University of Mons and EVS Broadcast Equipment SA, Belgium.

2. PHASE VOCODER

The underlying hypothesis of the phase vocoder is that a signal $x(n)$, sampled at frequency F_s, is a sum of P sinusoids, called *partials* [18]:

$$x(n) = \sum_{i=1}^{P} A_i \cos(\frac{n}{F_s}\omega_i + \phi_i) \quad (1)$$

each with its own angular frequency ω_i, amplitude A_i and phase ϕ_i. These 3 parameters are presumed to vary relatively slowly over time so that the signal is quasi-stationary and pseudo-periodic (e.g. speech and music). By segmenting the signal into overlapping frames to compute a Short-Term Fourier Transform (STFT), it is possible to use and modify the spectral amplitude and phase of each frame to either time-shift them (Section 2.1) or to interpolate new frames from them (Section 2.2).

2.1. Frame shifting

The most common implementation of the phase vocoder found in the literature [5, 7, 18] uses different sizes for the shift between frames (*hopsize*) during the analysis and the synthesis steps. The ratio between these two hopsizes equals the desired slow-down/speed-up factor. This means that to change the speed by a factor α with a synthesis hopsize R_s the analysis hopsize R_a must be:

$$R_a = \alpha R_s \quad (2)$$

Since the relative position of each frame in the output signal is different from that of the frames in the input signal, a simple overlap-add of the frames to generate that output will cause phase discontinuities. The main idea behind the phase vocoder is to adapt the phase of each partial according to the new hopsize R_s so that all the frames overlap seamlessly. Roughly speaking the adaptation needs to keep constant the variation of phase over time.

For each bin k of the STFT the phase variation between input frames i and $i-1$ is compared to the expected phase variation for that bin (a function of k and R_a). The difference between these two values (the *heterodyned phase increment*) is converted to the range $\pm\pi$ (Equation 6), divided by α and added to the theoretical phase variation for bin k in the output signal (a function of k and R_s). Finally this value is added to the phase of output frame $i-1$ to obtain the phase of output frame i (Equation 7). Note that the input frame 0 is reused as output frame 0 (Equation 3) and that the spectral amplitudes are not modified (Equation 4).

$$Y(0) = X(0) \quad (3)$$
$$|Y(i)| = |X(i)| \quad (4)$$
$$\Omega = \{0, \ldots, k\frac{2\pi}{L}, \ldots, (L-1)\frac{2\pi}{L}\} \quad (5)$$
$$\Delta\phi(i) = [\angle X(i) - \angle X(i-1) - R_a\Omega]_{2\pi} \quad (6)$$
$$\angle Y(i) = \angle Y(i-1) + R_s(\Omega + \frac{\Delta\phi(i)}{R_a}) \quad (7)$$

where $X(i)$ and $Y(i)$ are the Discrete Fourier Transforms (DFT) of the i^{th} input and output frames. $X(i)$, $Y(i)$, Ω and $\Delta\Phi(i)$ are L-sample vectors with L the length of a frame. $[]_{2\pi}$ denotes the conversion of the phase to the range $\pm\pi$ [18].

Once the DFT of a frame has been calculated the synthesis frame samples are computed by Inverse Discrete Fourier Transform (IDFT) and the frame is added by overlap-add to the output signal.

2.2. Frame generation

Another implementation of the phase vocoder was proposed by Dan Ellis in [19]. Contrary to the previous method it uses the same hopsize between the frames at analysis and synthesis time. Obviously when doing time-stretching the number of frames used to synthesize the output is different from the number of frames extracted from the input. Frames have to be dropped or created one way or another. In the algorithm developed by Ellis all frames are generated by interpolating the spectral amplitudes and accumulating the phase variations between the analysis frames.

The first step sets the initial synthesis frame spectrum $Y(0)$ equal to the initial analysis frame spectrum $X(0)$:

$$|Y(0)| = |X(0)| \quad (8)$$
$$\angle Y(0) = \angle X(0) \quad (9)$$

For the following synthesis frames the synthesis frame indices j are linearly mapped to the analysis indices i using Equation 10:

$$i = \alpha j \quad (10)$$

where i is generally not an integer value. For instance if the speed factor α is 0.5 (2× slower), $Y(7)$ corresponds to a frame position in the original audio equal to $\alpha \times 7 = 3.5$ (i.e. located between $X(3)$ and $X(4)$).

The spectrum $Y(j)$ of the j^{th} synthesis frame is a function of the amplitude and phase variations of its "surrounding" analysis frames as well as $\angle Y(j-1)$:

$$\lambda = i - \lfloor i \rfloor \quad (11)$$
$$|Y(j)| = (1-\lambda)|X(\lfloor i \rfloor)| + \lambda|X(\lfloor i \rfloor + 1)| \quad (12)$$
$$\Delta\phi(i) = [\angle X(\lfloor i \rfloor + 1) - \angle X(\lfloor i \rfloor)]_{2\pi} \quad (13)$$
$$\angle Y(j) = \angle Y(j-1) + \Delta\phi(i) \quad (14)$$

where $\lfloor i \rfloor$ is the integer value of i (the largest integer not greater than i). Finally the IFFT of each $Y(j)$ is computed and the samples are overlap-added into the output signal.

2.3. Phase locking

The methods presented in Section 2.1 and 2.2 are applied independently to each bin k of the spectrum in order to keep intact the phase constraints along the time (or horizontal) axis of the spectrogram. As a consequence there is no constraints with regard to the vertical axis: if there is a dependency between bins $k-1$, k, and $k+1$ in the input signal it is lost in the process. This causes the apparition of the phasiness artifact [8].

In order to correct this problem several algorithms have been proposed. In [6] Puckette uses the phase of the sum of the spectral values from bins $k-1$, k, and $k+1$ as the final phase value $\angle Y^*(i)$ for bin k:

$$\angle Y_k^*(i) = \angle(Y_{k-1}(i) + Y_k(i) + Y_{k+1}(i)) \quad (15)$$

Laroche et al. [7] proposed a somewhat more complex approach: the peaks in the spectrum are detected and the phases of their corresponding bins are updated as usual by the phase vocoder. The other bins located in the region of influence of each peak have their phases modified so as to keep constant their phase deviation from the peak's phase. As a result there is a horizontal phase locking for the peaks and a vertical phase locking for all the other parts

of the spectrum. A refinement of this method is to track the trajectories of the peaks over time and use the previous phase of each peak to compute the new one. This is important if a peak changes from one bin to another to avoid its phase being based on the phase of a previous non-peak bin. However tracking peaks over time is not always straightforward (peaks can appear, disappear, split or merge which increases the complexity of the task).

For small lengthening ratio Dorran et al. [14] recover phase coherence by slightly adjusting the phase of each synthesis frames so that after a few frames it converges to an almost perfect overlap with the analysis frame. From that point on a group of frames from the original signal can be added directly to the output signal without any phase transformation and therefore resulting in a (locally) perfect-quality audio signal. The phase gradual adjustment is calculated in order to be perceptually undetectable by a human ear.

3. PVSOLA

The new method presented in this section comes from an experimental observation we made on the phase vocoder (using [19]) and on the phase-locked vocoder (using *Identity Phase Locking* [7] as implemented in [18]):

> *Phasiness in the vocoder does not appear (or is not perceived) immediately. It takes a few frames before becoming noticeable.*

A simple experiment to observe this phenomenon is to alter a phase-locked vocoder so that the phase-locking happens only once every C frame. The other frames are processed with a normal phase vocoder. For small values of C (typically 3 to 5 frames), the difference in phasiness with a fully locked signal is barely noticeable at all (some artifact/ripples may appear in the spectrogram though). For larger values of C phasiness becomes audible in the vocoder output. We propose the following explanation for this behavior: the loss of vertical coherence is a slow phenomenon, it is not instantaneous, and the spectral content also vary relatively slowly (hypothesis of quasi-stationarity in Section 2). Therefore every time a peak is detected and locked its neighboring bins undergo some kind of phase reset: their final phase is only a function of the change of the peak's phase and their phase difference relatively to the peak's original phase. As for the peak, since the signal varies slowly it can be assumed that its position remains more or less coherent from one frame to another (or even across 3 to 5 frames) even if it changes of bin (the bin change is never an important jump in frequency).

3.1. Method overview

Based on these observations we propose to combine a time-domain and a frequency-domain approach. The method consists in a periodic reset of a phase vocoder by copying a frame directly from the input into the output and using it as a new starting point for the vocoder. The insertion point for the frame in the output is chosen by means of a cross-correlation measure.

3.2. Implementation details

We propose the following framework: first we generate C synthesis frames $(f_0, \ldots, f_{c-1})$ using a phase vocoder. Each frame f_i is L-sample long and is inserted in the output signal by overlap-add at sample t_i with:

$$t_i = iR_s \tag{16}$$

where t_i is the position at which the first sample of the synthesis frame is inserted and R_s is the hopsize at synthesis (note that we choose $R_s = L/4$ as is usually done in the literature). The last frame generated (f_{c-1}) is inserted at position t_{c-1}, the next one (f_c) should be inserted at t_c. Now instead of another vocoded frame we want to insert a frame f^* extracted directly from the input audio in order to naturally reset the phase of the vocoder but we know that this would cause phase discontinuities.

In order to minimize such discontinuities we allow to shift the position of f^* around t_c in the range $t_c \pm T$ (T is called the *tolerance*). The shift is obtained by computing the cross-correlation between the samples already in the output and the samples of f^*. However some samples of the output are "incomplete", they still need to be overlap-added with samples that would have been generated in the next steps of the phase vocoder (i.e. samples obtained by overlap-adding frames $f_c, f_{c+1}, \ldots$). As a result a frame overlapped in another position than t_c would cause a discontinuity in the otherwise constant time-envelope of the time-scaled signal. Besides the cross-correlation would be biased toward negative shifts around t_c. To overcome these problems additional frames $(f_c, f_{c+1}, \ldots, f_{\text{F}})$ are generated by the phase vocoder and temporarily inserted so that t_{F} respects the constraint in Equation 17:

$$t_{\text{F}} > t_c + L + T \tag{17}$$

which means that the first sample of the coming frame f_{F} would be inserted T samples after the end of f_c and that the output signal is "complete" up to sample t_{F} (no samples would be overlap-added anymore before that sample in a normal phase vocoder).

Position t_c corresponds to a position u_c in the input signal:

$$u_c = \alpha t_c \tag{18}$$

The next step consists in selecting a frame f^* of length L starting at sample u_c[1] in the input signal and adding it in the output signal at position $t_c + \delta$ with $-T \leq \delta \leq T$ (we fixed the tolerance $T = 2R_s$). Equation 21 defines χ, a cross-correlation measure between the frame f^* (Equation 20) and the output samples o (Equation 19) already generated:

$$o = \{y(t_c - 2R_s), \ldots, y(t_c + L - 1 + 2R_s)\} \tag{19}$$

$$f^* = \{x(u_c)h^2(0), \ldots, x(u_c + L - 1)h^2(L-1)\} \tag{20}$$

$$\chi = \text{xcorr}(o, f) \tag{21}$$

where $\{\}$ stands for a vector of values (a frame), $h^2(n)$ is the square of a Hann window (as defined in Equation 26) and xcorr is the cross-correlation function. $x(n)$ and $y(n)$ are the original and time-stretched signal respectively. The optimal value of δ corresponds to the position of the maximum of $|\chi_s|$, the subset of χ (as defined in Equation 23) that corresponds to an insertion of f^* in the position range $t_c \pm 2R_s$. Figure 1 shows an example of finding the offset δ using Equations 22 to 25:

$$\varepsilon = L + 4R_s = 2L \tag{22}$$

$$\chi_s = \{\chi(\varepsilon), \ldots, \chi(\varepsilon + 4R_s)\} \tag{23}$$

$$p = \text{argmax}(|\chi_s|) \tag{24}$$

$$\delta = p - 2R_s \tag{25}$$

[1] rounded to the nearest integer

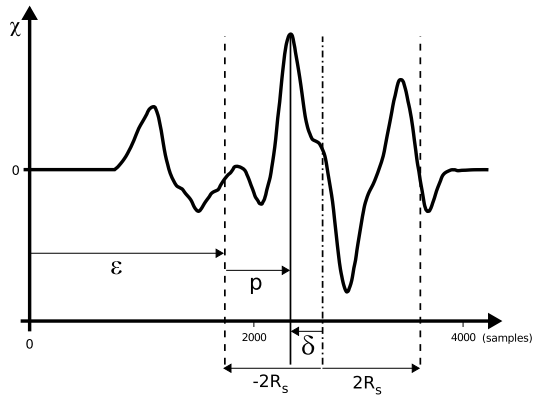

Figure 1: *δ is computed from the position p of the maximum value of a subset of χ. The dashed lines delimit the subset χ_s and the dash-dotted line represents a positioning of f^* exactly at $t = t_c$. In this example δ is < 0 and $\chi_s(p) > 0$. The frame length L is 1024.*

Notice that each frame processed through the phase vocoder undergoes two hann-windowing: one before the DFT and one after the IDFT before being overlap-added in the time-stretched signal. Therefore f^* has to be windowed by the square of a Hann window (Equation 20) in order to overlap-add properly with the output signal and the future frames. The Hann window $h(n)$ is defined as:

$$h(n) = \begin{cases} 0.5 - 0.5\cos(\frac{2\pi n}{L}) & \text{if } n = 0, \ldots, L-1 \\ 0 & \text{otherwise} \end{cases} \quad (26)$$

This definition is slightly different from the definition usually encountered (the denominator in the fraction is L instead of $L-1$) for the cumulated windowing would present a small ripple otherwise as explained in [20].

Then f^* is multiplied by the sign of $\chi_s(p)$ (in case of a negative peak) and overlap-added to the output audio (Figure 2).

Before inserting f^* the output samples between $t_c + \delta$ and $t_c + \delta + L - 1$ are windowed by a function $w(n)$ so that the overall accumulated windowing of the output remains constant (taking into account the frames yet to come). This also means that the samples of the output signal beyond $t_c + \delta + L - R_s$ that have been generated to compute the cross-correlation are set to zero. The computation of the envelope $w(n)$ applied to the time-stretched signal is presented in Figure 3 and Equation 27:

$$w(n) = h^2(n + 3R_s) + h^2(n + 2R_s) + h^2(n + R_s) \quad (27)$$

Finally since the frame f^* has been inserted "as is" the phase vocoder can be reinitialized to start a new step of the time-scaling process as if f^* were its initial frame f_0 and $t_c + \delta$ were its initial time position t_0. Note that each analysis frame used during this new step must be inverted if $\chi_s(p) < 0$.

3.3. Discussion

It is important to notice that due to the accumulation of shifts δ (one for each iteration) a drift from the original speed factor α could occur if no measure is taken to correct it. In our implementation we sum the values of δ for each phase reset and obtain a drift Δ. When Δ exceeds $\pm R_s$ the number of frames synthesized in the next iteration will be $C \mp 1$ and the value of Δ will change to $\Delta \mp R_s$. Theoretically Δ could even exceeds $\pm 2R_s$, in which case the number of frames synthesized will be $C \mp 2$ and Δ will become $\Delta \mp 2R_s$.

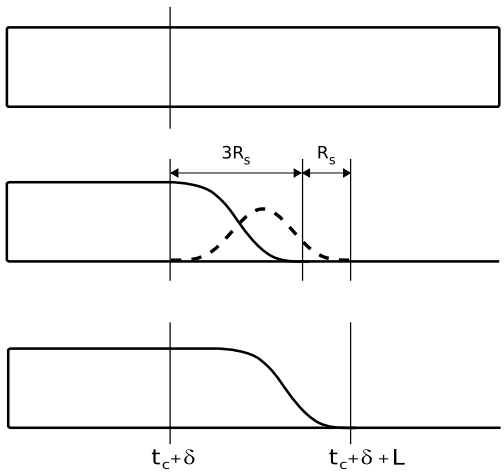

Figure 2: *Schematic view of the insertion of a frame f^* at position $t_c + \delta$. Top: output signal after insertion of additional frames for cross-correlation computation. Middle: windowed output signal (solid line) and frame f^* windowed by the square of a Hann window (dashed line). Bottom: resulting signal before the next iteration. The upcoming windowed frames will add to a constant time-envelope with this signal.*

Another interesting fact is that if we set $C = 0$, the resulting algorithm is very close to a SOLA-like method except that the additional frames used for the cross-correlation are still generated by a phase vocoder. On the contrary $C = \infty$ changes the method back into a non-locked phase vocoder.

Finally in Section 3.2 we take the first sample of a frame as the reference for positioning. One might use the middle sample of each frame instead. This will not create any significant difference with the method proposed above.

4. RESULTS

This method can be applied to any phase-vocoder algorithm. For the following tests we implemented a modified version of the algorithm from [19]. We performed both formal and informal assesments presented respectively in Section 4.1 and 4.2.

4.1. Formal listening tests

We use sentences selected from the CMU ARCTIC databases [21] among the 4 US speakers, namely *clb*, *slt*, *bdl* and *rms* (two female and two male speakers). 50 sentences are randomly picked for each speaker and each sentence is processed by 4 different algorithms: a phase-vocoder, a phase-locked vocoder, a time-domain method (SOLAFS) and our method PVSOLA. Each process is applied with two speed factors: $\alpha = 1/1.5$ and $\alpha = 1/3$ (i.e. 1.5 and 3 times slower).

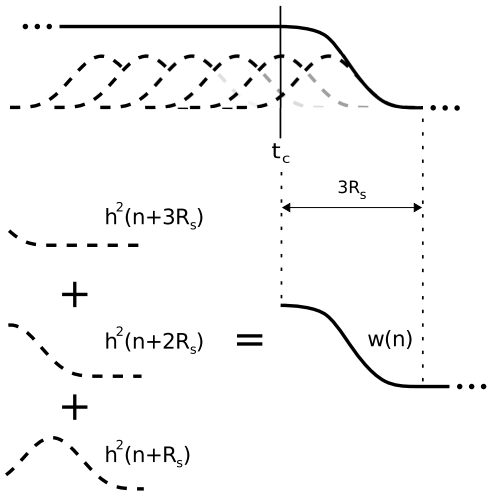

Figure 3: *Schematic view of the computation process for the weighting function $w(n)$ that will be applied to the output signal after $t_c + \delta$. Top: in a standard phase vocoder, the squared Hann windows would sum to a constant value except for the last samples because there are frames not yet overlap-added after t_c. We want to reproduce that behavior at $t_c + \delta$ so that f^* overlap-adds seamlessly. Bottom: The time envelope is the sum of three squared Hann windows with a shift R_s between each one.*

For the two phase vocoders we use the implementation available in [18] and for SOLAFS we use the implementation from [22]. We empirically set $L = 512$ samples and $R_s = L/4$ for the vocoders and PVSOLA. In our informal tests SOLAFS generally provided better quality with $L = 256$ so we kept that value. The parameters specific to PVSOLA are $C = 3$ and $T = 2R_s$.

PVSOLA is compared to the other three methods via a *Comparative Mean Opinion Score* (CMOS) test [23]. Participants are given the unprocessed audio signal as a reference (R) and they are asked to score the comparative quality of two time-stretched versions of the signal (both of them with the same speed modification). One is PVSOLA, the other is randomly chosen among the three state-of-the-art algorithms. The two signals are randomly presented as A and B. Each listener takes 30 tests, 10 for each concurrent method. The question asked is: "*When compared to reference R, A is:* much better, better, slightly better, about the same, slightly worse, worse, much worse *than B ?*"

Each choice made by a listener corresponds to a score between ± 3. In case A is PVSOLA, "much better" is worth 3 points, "better" 2 points and so on until "much worse" which means -3 points. On the contrary when B is PVSOLA, the scale is reversed with "much worse" worth 3 points and "much better" -3 points. In short when PVSOLA is preferred it gets a positive grade and when it is not it gets a negative one. 16 people took the test (among which 9 are working in speech processing) and the results are shown in Table 1 and Figure 4 and 5.

From these results one can see that for a speed slowdown factor of 1.5 our method is globally preferred except for SOLAFS with female voices where both methods are deemed equivalent. Besides SOLAFS performs relatively better than the phase-locked vocoder which in turn performs better than the phase vocoder. This is an expected result as time-domain methods usually give better results when applied to speech and the phase-locked vocoder is

Table 1: *CMOS test results with 0.95 confidence intervals for female (*clb *and* slt*) and male (*bdl *and* rms*) speakers. PVSOLA is compared to the phase vocoder (*pvoc*), the phase-locked vocoder (*plock*) and* SOLAFS.

	female	
$1/\alpha$	1.5	3
pvoc	2.03 ± 0.3	0.66 ± 0.43
plock	0.97 ± 0.41	1.86 ± 0.3
solafs	0.14 ± 0.32	1.21 ± 0.27
	male	
$1/\alpha$	1.5	3
pvoc	2.49 ± 0.32	1.05 ± 0.47
plock	1.78 ± 0.29	1.71 ± 0.3
solafs	1.13 ± 0.36	1.77 ± 0.27

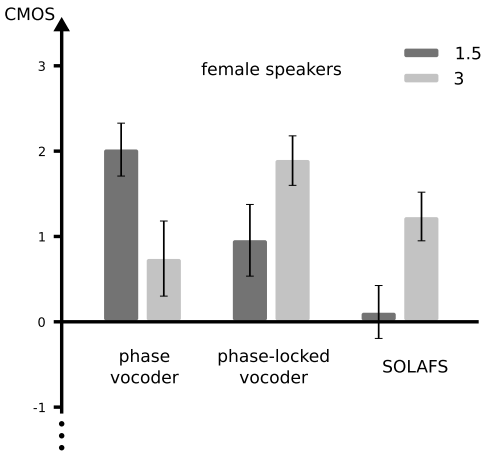

Figure 4: *Results for the CMOS test for female speakers* clb *and* slt*. The dark and light gray bars represent the mean CMOS score for a speed ratio of respectively 1.5 and 3. 0.95 confidence intervals are indicated for information.*

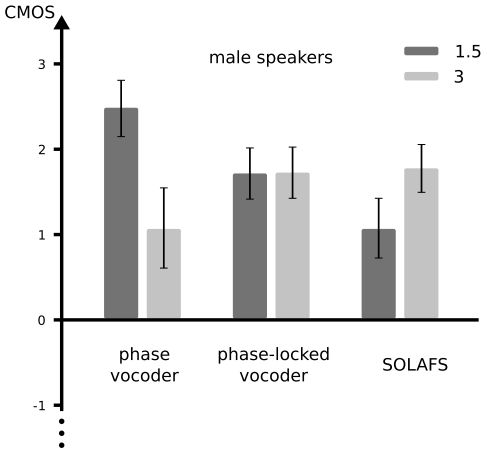

Figure 5: *Results for the CMOS test for male speakers* bdl *and* rms*. The dark and light gray bars represent the mean CMOS score for a speed ratio of respectively 1.5 and 3. 0.95 confidence intervals are indicated for information.*

supposed to be better than the phase vocoder.

For the higher slowdown factor 3, our method is again observed to outperform other approaches, notably better than SOLAFS in both tables and better than the phase-locked vocoder for female voices, but it has lost ground to the normal phase vocoder which has a better score than the two other approaches. After the test we discussed this with the listeners and we could establish that it was not a mistake. Indeed with this time-stretching ratio every method produces more artifacts (frame repetition for SOLAFS, metallic sound for the phase-locked vocoder, phasiness for the phase vocoder and some sort of amplitude modulations for PVSOLA). The listeners said that in some cases they "preferred" the defect of the phase vocoder to that of PVSOLA for a certain number of sentences of the dateset. It is still a minority of files for which this happens since the overall result is still in favor of PVSOLA but this has to be analyzed further.

4.2. Informal tests and discussions

We applied the algorithm to various signals: speech, singing voice, mono and polyphonic music and obtained improved results over all other methods for monophonic signals (speech, singing and music) while the algorithm suffers from audible phase mismatches for polyphonic signals.

Several values for C and L have been tried and the best tradeoff seems to be $C = 3$ and $L = 512$ samples for a sampling frequency $F_s = 16$ kHz (i.e. $L = 32$ ms). As for other sampling frequencies (in singing and music data) we set L so that it also corresponds to about 30 ms. Nevertheless we noticed that in general the algorithm is not very sensitive to the value of L (between 20 and 40 ms). For $C = 3$ and a reasonable speed factor (between 1 and 3 times slower) we generally notice an important reduction of the phasiness. We generated some test samples for even slower speed factor ($\times$ 5) with mixed results (some good, others presenting many artifacts).

For larger values of C perceptible phase incoherencies appear in the time-stretched signals probably because the phases of the different partials are already out-of-phase with each other. It seems that the cross-correlation measure can help to match some of these partials with the ones from the input frame f^* but not all of them thus creating artifacts that resemble an amplitude modulation (the audio sounds "hashed", sometimes a beat appears at a frequency corresponding to CR_s). Note that even for values of $C \leq 3$ these mismatches may still appear but to a lesser extent, they are often almost inaudible. However discussions with listeners have shown that in some worst-case scenarios they can become a real inconvenience as explained in section 4.1.

As a side-effect of the algorithm, transients tend to be well-preserved contrary to what happens with time-domain (transient duplication) or phase vocoder-based algorithms (transient smearing). Apparently f^* can be advantageously positioned so that the transient is preserved due to the relatively large value of T. Although this may prove interesting it is not systematic and has yet to be investigated.

The main drawback of our method lies in its computational complexity when compared with time-domain or phase vocoder approaches. Indeed not only do we compute a cross-correlation every C frame but we also generate extra frames for its computation that will be eventually dropped and replaced by new ones. Roughly speaking we measured that our MATLAB implementation was three to four times slower than a phase vocoder. A profiling of the process shows that the most time-consuming task is by far the cross-correlation computation (about 40%). However results of benchmarking within MATLAB must always be taken with care since some operations (such as selecting a frame in a signal) are not well-optimized. We estimate that a C implementation of PVSOLA could be less than two times slower than that of a phase vocoder.

5. FUTURE WORK

We plan to work on different aspects of PVSOLA that can be improved:

- in [13] Röbel proposes to modify a cross-correlation to take into account only the partials and ignore the noisy components. We could use this method to refine the positioning of the frames f^* and reduce the artifacts of PVSOLA.
- For the moment we developed and implemented our algorithm as a SOLA-modified phase vocoder. A major change would be to use a WSOLA-like approach to the selection of f^*. Indeed we could select a frame from the input signal that would be optimal for an insertion at t_c instead of trying to find the best position $t_c + \delta$ for a given frame. This would suppress at the same time the need for additional frames (used for the cross-correlation computation) and for occasional additions or removals of frames when $|\Delta| > R_s$ (see Section 3.3). We are currently working on this topic.
- The results on polyphonic sounds are not as good as those on monophonic sounds. We plan to investigate this problem as well.
- PVSOLA has only been tested on a standard phase vocoder. Using a phase-locked vocoder could make it possible to increase the optimal value for C thus reducing the computational load.

6. CONCLUSIONS

This paper presented a new approach to modify the length of an audio signal without changing its perceived content. The method proposes a combination of a time-domain and a frequency-domain process. It consists in a periodic reset of a phase vocoder by copying a frame directly from the input into the output and using it as a new starting point for the vocoder. The insertion point for the frame in the output is chosen by means of a cross-correlation measure. Informal listening tests have highlighted a reduction of the phase vocoder's phasiness and formal listening tests have shown that our method was generally preferred to existing state-of-the-art algorithms. Both formal and informal tests have pointed out that under certain circumstances the quality of the time-stretched audio could be perceived poorly because of discontinuities in the signal. Various suggestions have been made to improve this situation as part of future work or ongoing research.

7. EXTERNAL LINKS

Examples of audio time-stretching with PVSOLA are available at: http://tcts.fpms.ac.be/~moinet/pvsola/

8. REFERENCES

[1] J. Bonada, "Audio time-scale modification in the context of professional audio post-production", research work for PhD program, Universitat Pompeu Fabra, Barcelona, Fall 2002.

[2] W. Verhelst, "Overlap-add methods for time-scaling of speech", *Speech Communication*, vol. 30, no. 4, pp. 207–221, April 2000.

[3] D. Hejna and B.R. Musicus, "The SOLAFS time-scale modification algorithm", Tech. Rep., BBN Technical Report, July 1991.

[4] E. Moulines, F. Charpentier, and C. Hamon, "A diphone synthesis system based on time-domain prosodic modifications of speech", in *Proc. of International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Glasgow, Scotland, May 23-26 1989, pp. 238–241.

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

[6] M. Puckette, "Phase-locked vocoder", in *Proc. of IEEE ASSP Workshop on Applications of Signal Processing to Audio and Acoustics*, Mohonk, NY, USA, Oct. 15-18 1995, pp. 222–225.

[7] J. Laroche and M. Dolson, "Improved phase vocoder time-scale modification of audio", *IEEE Transactions on Speech and Audio Processing*, vol. 7, no. 3, pp. 323–332, May 1999.

[8] J. Laroche and M. Dolson, "Phase-vocoder: about this phasiness business", in *Proc. of 1997 IEEE ASSP Workshop on Applications of Signal Processing to Audio and Acoustics*, New Paltz, NY, USA, Oct. 19-22 1997, pp. 55–58.

[9] J. Bonada, "Automatic technique in frequency domain for near-lossless time-scale modification of audio", in *Proc. of the International Computer Music Conference (ICMC)*, Berlin, Germany, 27 August – 1 September 2000, pp. 396–399.

[10] Y. Stylianou, *Harmonic plus noise models for speech combined with statistical methods, for speech and speaker modifications*, Ph.D. thesis, École Nationale Supérieure des Télécommunications, 1996.

[11] X. Serra and J. Bonada, "Sound transformations based on the sms high level attributes", in *Proc. of the 1st International Conference on Digital Audio Effects (DAFx-98)*, Barcelona, Spain, Nov. 19-21 1998.

[12] T.S. Verma and T.H.Y. Meng, "Time scale modification using a sines+transients+noise signal model", in *Proc. of the 1st International Conference on Digital Audio Effects (DAFx-98)*, Barcelona, Spain, Nov. 19-21 1998, pp. 49–52.

[13] A. Röbel, "A shape-invariant phase vocoder for speech transformation", in *Proc. of the 13th International Conference on Digital Audio Effects (DAFx-10)*, Graz, Austria, Sept. 6-10 2010.

[14] D. Doran, E. Coyle, and R. Lawlor, "An efficient phasiness reduction technique for moderate audio time-scale modification", in *Proc. of the 7th International Conference on Digital Audio Effects (DAFx-04)*, London, UK, Oct. 5-8 2004, pp. 83–88.

[15] H. Kawahara, I. Masuda-Katsuse, and A. de Cheveigne, "Restructuring speech representations using a pitch-adaptive time-frequency smoothing and an instantaneous-frequency-based f0 extraction: Possible role of a repetitive structure in sounds", *Speech Communication*, vol. 27, no. 3-4, pp. 187–207, April 1999.

[16] T. Karrer, E. Lee, and J. Borchers, "Phavorit: A phase vocoder for real-time interactive time-stretching", in *Proc. of the International Computer Music Conference (ICMC)*, New Orleans, USA, Nov. 6-11 2006, pp. 708–715.

[17] A. Röbel, "A new approach to transient processing in the phase vocoder", in *Proc. of the 6th International Conference on Digital Audio Effects (DAFx-03)*, London, UK, Sept. 8-11 2003.

[18] T. Dutoit and J. Laroche, *Applied Signal Processing – A Matlab-Based Proof of Concept*, chapter How does audio effects processor perform pitch shifting ?, pp. 149–185, Springer Science+Business Media, 2009.

[19] D. P. W. Ellis, "A phase vocoder in Matlab", 2002, Web resource, last consulted in March 2011.

[20] A. De Götzen, N. Bernardini, and D. Arfib, "Traditional (?) implementations of a phase-vocoder: the tricks of the trade", in *Proc. of the 3rd International Conference on Digital Audio Effects (DAFx-00)*, Verona, Italy, Dec. 7-9 2000, pp. 37–44.

[21] John Kominek and Alan W Black, "CMU arctic databases for speech synthesis", Tech. Rep., Language Technologies Institute, School of Computer Science, Carnegie Mellon University, 2003.

[22] D. P. W. Ellis, "SOLAFS in Matlab", 2006, Web resource, last consulted in March 2011.

[23] V. Grancharov and W. Kleijn, *Handbook of Speech Processing*, chapter Speech Quality Assessment, pp. 83–99, Springer, 2007.