AUDIO TIME-SCALING FOR SLOW MOTION SPORTS VIDEOS

Alexis Moinet, Thierry Dutoit
TCTS Lab. – numediart Institute
University of Mons, Belgium

Philippe Latour
INTELSIG Laboratory – Dept. of Electrical Engineering and Computer Science
University of Liège, Belgium

ABSTRACT

Slow motion videos are frequently featured during broadcast of sports events. However, these videos do not feature any audio channel, apart from the live ambiance and comments from sports presenters. Standard audio time-scaling methods were not developed with such noisy signal in mind and they do not always permit to obtain an acceptable acoustic quality. In this work, we present a new approach that creates high-quality time-stretched version of sport audio recordings while preserving all their transient events.

1. INTRODUCTION

In today's sports television broadcast, slow motion videos are omnipresent. They highlight crucial actions and incidents, they communicate the ambiance of an event to the distant viewers, they help to understand referee's decisions, etc. However, these videos lack an important part of the atmosphere. There is no audio channel matching what is shown. Indeed, time-scaling an audio signal without any processing causes a scaling of the frequency content, up or down, whether it's accelerated or decelerated. In the domain of speech and music processing, many time-scaling methods [1] have been developed to address this issue. However, they are based on the hypothesis that the underlying signal is mostly a sum of sinusoids. As we show later, we obtained interesting results using a phase vocoder but transient detection is unreliable at best.

In [2], Picard introduces a method for time-scaling derived from sound texture synthesis. It decomposes a set of contact sound recordings into audio grains and computes *correlation patterns* between each grain and the rest of the recordings. The process then shifts the grains to new positions, corresponding to the desired speed factor α and fills the gaps with grains that maximize correlation patterns. This cannot be performed in realtime because it needs to compute the correlation patterns of every grain with every file. Besides, it supposes that a set of recordings is already available as opposed to our situation where we start "from scratch" for each new slow motion excerpt. Moreover, keeping all the recordings in memory is too strong a constraint for embedding it in actual slow motion systems. Nevertheless, the hypotheses of this method fit better with the signals recorded during sports events.

We propose a new method that decomposes the input file into non-overlapping grains that are re-spaced[1] according to a speed factor α, and then the empty spaces possibly created between two grains are filled with content generated "on-the-fly". This approach can be described as a three-step *"split-shift-fill"* algorithm. These three steps are detailed respectively in Sections 2, 3 and 4, and the process is summarized in Section 5 while some points are discussed in Section 6. The sports recordings used during the tests and the results of these tests are presented in Section 7.

[1]Note that each grain has a certain "freedom of movement" around its theoretical position to minimize artifacts. This is detailed in Section 3.

2. SPLIT

The first part of the algorithm divides the input signal $x(n)$ into frames of variable length, the so-called grains, whose boundaries are located at non-transient samples. Picard et al. [2] propose to place segmentation points at minimums of the spectral flux. However, we observed that using an energy measure, as explained in Section 2.1, gives similar or better results when applied to the data from Section 7.1 while being computationally less intensive.

2.1. Segmentation

The first step is to compute a *transientness* function. In this case though, contrary to standard transient detection algorithms, the goal is to detect non-transient parts of the signal. For this, we use, for instance, the energy measure $e(n)$ defined in Equation 1.

$$e(n) = \sum_{m=0}^{N-1} |x(nH - \frac{N}{2} + m)w_e(m)|^2 \quad (1)$$

with N the frame length, H the hopsize and w_e a N-sample Hann window function. In a second step, t_g and L_g, the two parameters of the g$^{\text{th}}$ grain g_g, are computed. t_g is the position of the first sample of g_g and L_g its number of samples. The length L_g of a grain is simply equal to the distance between its starting point and the starting point of the next grain

$$L_g = t_{g+1} - t_g \quad (2)$$

and t_g is computed by looking for a minimum of energy in a region located between L_{min} and L_{max} samples after t_{g-1}, as shown in Figure 1. L_{min} and L_{max} correspond respectively to the lower and upper limit of a grain length. t_g can be further refined by looking for the sample closest to zero in a subset x_s of the signal.

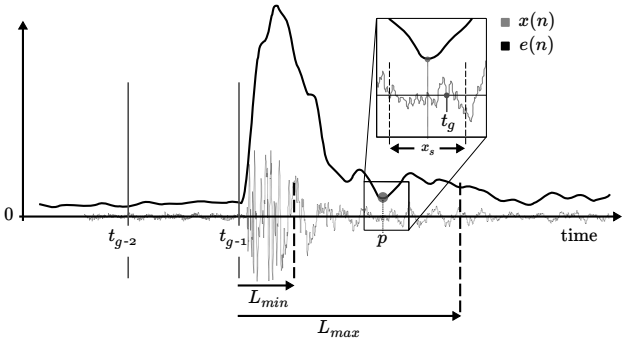

Figure 1: *The energy of an audio signal $x(n)$ is computed and its local minimums are used to place temporary segmentation points p which are then refined into actual segmentation points t_g.*

2.2. Parameters

We empirically set the computation parameters to $N = 256$ and $H = 4$, or 5.3 ms and 83.3 μs, respectively, for a sampling rate $F_s = 48$ kHz. As for the limits L_{min} and L_{max}, we arbitrarily set the minimum duration to 10 ms and the maximum one to 40 ms which corresponds to the duration of a video frame at 25 fps. This translates into $L_{min} = 480$ and $L_{max} = 1920$ samples. The value for N is picked as a compromise between two opposite trends. For longer frame sizes, $e(n)$ is smoother but each transient in $x(n)$ influences $e(n)$ over a larger span. This, in turn, blurs the location of minimal values that are essential to the segmentation. To put it differently, a small N makes it more likely that some of the frames used to compute $e(n)$ are devoid of any transient and, as such, are neat and valid segmentation points. However, too small a value for N produces a noisy $e(n)$ (closer to $x(n)^2$ as N decreases) which also reduces the segmentation quality.

3. SHIFT

In the second part of the method, the grains are shifted from their position t_g in $x(n)$ to a new position u_g in the output signal $y(n)$. Theoretically, for a speed factor α, the new position ought to be

$$u_g = u_{g-1} + \alpha L_{g-1} \tag{3}$$

with $u_0 = t_0 = 0$. However, if $\alpha > 1$, the method detailed in Section 4 has to generate content that fits perfectly into the gap created between two grains, which is often not possible. Besides, in the case $\alpha < 1$, it forces an overlap-add between g_{g-1} and g_g at a likely non-optimal position. Our experiments show that this is prone to artifacts and we apply two improvements to reduce them. On the one hand, as presented in Section 3.1, we use the principle from SOLA [3]: we allow the grains to be shifted around position u_g by δ_g samples, with $|\delta_g| \leq \Delta$, in order to minimize discontinuities when adding it to $y(n)$. For reasonable values of Δ, the desynchronization between audio and images in a slow motion video cannot be perceived while the acoustic quality is significantly increased. On the other hand, we show in Section 3.2 that, in some cases, the two grains can be directly concatenated.

3.1. Normal shift

To insert each grain in the output, we use a cross-correlation measure $\chi(n)$ between $\iota(n)$, the h first samples of grain g_g, and $o(n)$, the samples of $y(n)$ located $\pm\Delta$ samples around sample u_g, as defined in Equation 6.

$$\begin{aligned} \iota(n) &= \{x(t_g), \ldots, x(t_g + h - 1)\} \\ &= \{g_g(0), \ldots, g_g(h-1)\} \end{aligned} \tag{4}$$

$$o(n) = \{y(u_g - \Delta), \ldots, y(u_g + \Delta - 1)\} \tag{5}$$

$$\chi(m) = o(n) \star \iota(n) \tag{6}$$

Note that the computation of $\chi(n)$ supposes that the output samples, up to $y(u_g + \Delta - 1)$, have been previously filled, after insertion of g_{g-1}, with content generated by the method described in Section 4. The position p of the absolute maximum peak of $\chi_s(n)$, a subset of $\chi(n)$, determines δ_g, the position shift of g_g around u_g.

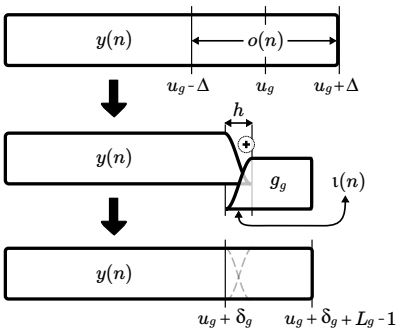

Figure 2: *g_g and $y(n)$ extremities are windowed by half of a $2h$-sample Hann window. Then g_g is overlap-added into $y(n)$, starting at position $u_g + \delta_g$, to obtain the next version of $y(n)$.*

3.1.1. Overlap-add

g_g is inserted into $y(n)$ at $u_g + \delta_g$ through overlap-add (OLA). The h first samples of g_g are windowed by w_h, the left half of a $2h$-sample Hann window and the h last valid samples of $y(n)$ are windowed by the complementary window $1 - w_h$. Figure 2 and Equations 7 to 9 present the overlap-add procedure. Note that in case $\chi_s(p)$ is negative (i.e. the samples are anti-correlated), the grain samples are inverted by a sign change before the OLA.

$$n = u_g + \delta_g + m \tag{7}$$

$$\varsigma_g = \text{sgn}(\chi_s(p)) \tag{8}$$

$$y(n) = y(n)(1 - w(m)) + \varsigma_g \, g_g(m) w(m) \tag{9}$$

for $m = [0, \ldots, L_g - 1]$, with ς_g equal to either 1 or -1 depending on the sign of $\chi_s(p)$. w is the left half of a Tukey window. The first and last h samples of a Tukey window are cosine tapers whereas the samples in-between are set to one. In other words, the first h samples of w are equal to w_h and the remainder equal to one.

3.2. Concatenation

If u_g is less than Δ samples away from the end of the previous grain in $y(n)$, inserting g_g becomes a simple concatenation of the two grains, ensuring perfect continuity in the output signal, as shown in Figure 3. The double condition in Equation 10 checks

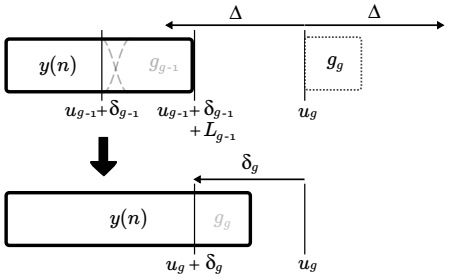

Figure 3: *In some cases, it is possible to append g_g directly to g_{g-1} in the output signal $y(n)$. This reduces the computational cost and increases the acoustic quality of the output.*

that $u_{g-1} + \delta_{g-1} + L_{g-1}$, the first sample after g_{g-1} in $y(n)$, is

within $u_g \pm \Delta$, the range of available positions for insertion of g_g.

$$u_g - \Delta \leq u_{g-1} + \delta_{g-1} + L_{g-1} \leq u_g + \Delta \tag{10}$$

If the condition is respected, the grain is added directly into the output as per Equations 11 to 13. Note, however, that it is multiplied by ς_{g-1}, as computed for g_{g-1} during the previous iteration of the algorithm (be it a normal shift or a concatenation). Obviously, if g_{g-1} has been inverted, g_g must be inverted as well.

$$n = u_{g-1} + \delta_{g-1} + L_{g-1} + m \tag{11}$$

$$\varsigma_g = \varsigma_{g-1} \tag{12}$$

$$y(n) = \varsigma_g \, g_g(m) \tag{13}$$

for $m = [0, \ldots, L_g - 1]$. In the concatenation case, as opposed to the computation made using a cross-correlation in Section 3.1, δ_g is computed using Equation 14, where it is the only unknown. The value of δ_g is used in the next iteration of the algorithm, when grain g_{g+1} is added to $y(n)$.

$$u_g + \delta_g = u_{g-1} + \delta_{g-1} + L_{g-1} \tag{14}$$

Section 6 discusses the advantages that this optimization brings.

3.3. Parameters

In the same way that we decided the maximum length L_{max} of each grain, we fixed $\Delta = 1024$ samples at $F_s = 48$ kHz. This corresponds more or less to 20 ms, so that the shift δ_g does not desynchronize perceptibly the audio and image streams [4], for a total span $[-\Delta, \ldots, \Delta]$ close to the 40 ms duration of a video frame at 25 fps. The second parameter is set to $h = 128$ samples.

4. FILL

The third part of the algorithm synthesizes content to fill the gap in $y(n)$ between g_{g-1} and g_g, when g_g cannot be concatenated. Since the sport recordings in the database of Section 7.1 are mainly made of noise, we first considered generating that content through spectral modification of Gaussian white noise. For this, we apply a Short-Time Fourier Transform (STFT) to Gaussian white noise and replace the spectral amplitude with that of $x(n)$ around t_g, followed by an inverse STFT (ISTFT). However, early experiments showed that in order to obtain synthetic signals that reproduces correctly the background noises (crowd, whistles, speech, etc.) we needed to use analysis frames at least 150 to 300 ms-long. Smaller lengths caused a distortion that seems to match Boll's description in [5] and referred to as *musical noise* in [6] and later publications, although their context is different (speech denoising).

Although this produces high-quality time-stretching in most part of the signal, it also causes artifacts in the synthetic content surrounding transients. Indeed, compared to common frame lengths, 150-300 ms is oversized and many analysis frames include transients. This corrupts the spectral amplitude used to modify the white noise and the synthetic part of the time-scaled signal sounds like transient smearing from a phase vocoder, except that it is interleaved with grains from the input signal, transients included.

4.1. Self cross-synthesis

In the noisy condition of the recordings, detecting and processing transients is not a reliable option. Therefore, in order to determine the spectral envelopes of the filtering while removing artifacts caused by nearby transients, we propose to combine short and long-term analyses through a form of cross-synthesis. Indeed, on the one hand we suppose that a short-term frame extracted around time t_g has a spectral envelope devoid of the influence of surrounding transients. However, it causes musical noise and lacks the pseudo-stationarity needed to create a synthetic signal of acceptable quality. On the other hand, a long-term frame has this pseudo-stationarity but its spectral amplitude is distorted by nearby transients that it contains. In the following, we apply cross-synthesis between these two frames. As they come from the same signal, centered on the same instant t_g, with only a different length, we use the name *self cross-synthesis*.

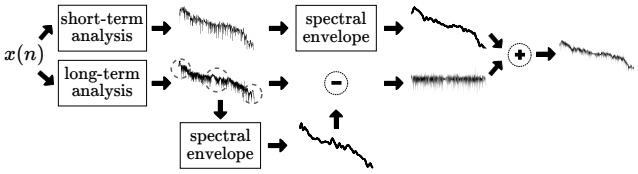

Figure 4: *Contribution of the long-term spectral envelope is removed from long-term amplitude spectrum and contribution of the short-term envelope is added back. The artifacts caused by a nearby transient in the long-term analysis (dashed circles) are absent from the cross-synthesized amplitude spectrum on the right.*

The principle of the self cross-synthesis method is illustrated in Figure 4. In order to fill the empty space between g_{g-1} and g_g, two analysis frames f_s and f_l are built from the input signal $x(n)$, both centered at t_g. The length of the frames is respectively N_s and N_l, with $N_s << N_l$. f_s is the *short-term analysis frame* and f_l the *long-term analysis frame*. For both frames, spectral envelopes are extracted and f_l has its spectral envelope compensated for. Therefore it is "whitened" into a signal whose spectral amplitude is globally flat while still containing all its local details. Then this "whitened" signal is "re-colored" using the spectral envelope of f_s, the short-term analysis frame. Finally the signal resulting from this self cross-synthesis is used to synthesize the content to fill the inter-grain empty space. For each aspect of this process, different tools could be used. Cross-synthesis is usually achieved either sample-by-sample with time-domain direct and inverse autoregressive filtering or, in the spectral domain, frame-by-frame through STFT and its inverse, by division and multiplication of the amplitude spectrum. Some preliminary tests showed that autoregressive filtering based on linear prediction analysis could cause an annoying metallic artifact when applied to the noisy background of sports events. Therefore a spectral framework seems more appropriate. In the next section we detail a way to carry out this spectral-only self cross-synthesis process in the cepstral domain. It is similar to the method decribed by Burr et al. in [7].

4.2. Cepstral cross-synthesis

For a given signal $x(n)$, each cepstral coefficient (or quefrency) of its real cepstrum $c_x(n)$ represents a different "level" of detail of $|X(k)|$, its amplitude spectrum. More exactly, the first coefficients represent a gross approximation of the amplitudes, the spectral envelope, whereas the higher quefrencies correspond to the finer details. Therefore, if one sets the values of $c_x(n)$ to zero for $n \geq C$ and then transforms this biased cepstrum back into the spectral domain, the resulting amplitude spectrum is an envelope of $|X(k)|$. The value of C determines the level of details, the *coarseness*, of

the envelope with higher values meaning more details. We use this property to compute $|F_\chi(k)|$, the spectral cross-synthesis of the two amplitude spectrums $|F_s(k)|$ and $|F_l(k)|$ of frames f_s and f_l.

First we compute $F_s(k)$ and $F_l(k)$, the discrete Fourier transforms (DFTs) of f_s and f_l, over N_l points[2]. Then, from $|F_s(k)|$ and $|F_l(k)|$, we compute $c_s(n)$ and $c_l(n)$, the cepstral transforms, as the discrete cosine transform (DCT)[3] of their logarithms. We then proceed to the creation of a new cepstrum $c_\chi(n)$ by concatenating the C first quefrencies of $c_s(n)$ and the $N_c - C$ last of $c_l(n)$, with N_c equal to the number of cepstral coefficients. Finally we compute the cross-synthesized amplitude spectrum $|F_\chi(k)|$ by inversion of the DCT and exponentiation. Note that since DFTs are symmetrical, we reduce the computational cost of the cepstral transformation by using the first half of each DFT in the DCT of the logarithms and, therefore, $N_c = {}^{N_l}/_2$ cepstral coefficients.

For some appropriate values of C, discussed in Section 4.5, the resulting set of amplitudes $|F_\chi(k)|$ has the spectral envelope of f_s, hopefully free from any artifact caused by the surrounding transients. It also has the fine details from f_l, which capture better information such as whistles, speech, and the long-term stationarity of the background noise of most sports such as football. Note that, of course, for a value $C = 0$, the transformation does nothing and is equivalent to using $|F_l(k)|$ as the spectral filter which is the approach described in the first paragraph of Section 4. Similarly for $C = N_c$, the process is equivalent to using only $|F_s(k)|$.

4.3. Low-pass filter

In practice, when applying this method, the spectrogram of the time-scaled signals has the expected aspect but there is also musical noise, albeit less than when using only f_s. Since it does not appear when using only f_l (i.e. $C = 0$), it is certainly due to the replacement of its first C cepstral coefficients with those of f_s. We add one more step to the process in order to reduce this defect. It smoothes the evolution over time of the short-term cepstral coefficients, $c_s(n)$, using a first order low-pass filter.

Let us consider $c_{s,g}(n)$ the n$^{\text{th}}$ cepstral coefficient computed from the short-term frame $f_{s,g}$ centered at each instant t_g. The low-pass filter is written

$$\hat{c}_{s,0}(n) = c_{s,0}(n) \tag{15}$$

$$\hat{c}_{s,g}(n) = (1-\lambda)\,\hat{c}_{s,g-1}(n) + \lambda\, c_{s,g}(n) \tag{16}$$

and for each gap between grains g_{g-1}, and g_g, $c_\chi(n)$ is built as

$$c_\chi(n) = \{\hat{c}_{s,g}(0), \ldots, \hat{c}_{s,g}(C-1), c_l(C), \ldots, c_l(N_c-1)\} \tag{17}$$

4.4. Insertion

Once the spectral amplitude $|F_\chi(k)|$ is computed, it is used in an ISTFT, with phases from the STFT of a Gaussian noise, to synthesize the content filling the inter-grain space. Z samples[4] of a colored noise, $z(n)$, are generated, with Z defined as

$$Z = (\alpha - 1)L_g + h + \Delta - \delta_{g-1} + \Theta$$

Then we compute the cross-correlation measure χ between the first Θ samples of $z(n)$ and the last h samples of g_{g-1}.

$$\iota(n) = \{z(0), \ldots, z(\Theta - 1)\} \tag{18}$$

$$o(n) = \varsigma_{g-1}\,\{g_{g-1}(L_{g-1} - h), \ldots, g_{g-1}(L_{g-1} - 1)\} \tag{19}$$

$$\chi(m) = o(n) \star \iota(n) \tag{20}$$

In the same way it is done in Section 3.1, the maximum peak of the absolute value of χ_s, a subset of χ, defines the position p of the part of $z(n)$ that overlaps best with g_{g-1} in the output signal. Finally the selected part of $z(n)$ is multiplied by ς, the sign of $\chi_s(p)$, and overlap-added to the output signal at the end of grain g_{g-1}, following Equations 21 to 23.

$$n = u_{g-1} + \delta_{g-1} + L_{g-1} - h + m \tag{21}$$

$$\varsigma = \text{sgn}(\chi_s(p)) \tag{22}$$

$$y(n) = y(n)(1 - w(m)) + \varsigma z(p+m)w(m) \tag{23}$$

for $m = [0, \ldots, Z - p - 1]$. w is a Tukey window equal to w_h, introduced in 3.1.1, for its first h samples and then set to one.

4.5. Parameters

There are several constant parameters to take into account in this method: N_s, N_l, λ and Θ. The parameters used during the tests are $N_s = 1024$ samples, $N_l = 8192$ samples, $\lambda = 0.25$ and $\Theta = 4096$. All but Θ, which is set arbitrarily, have been chosen experimentally through trial and error during informal tests. Note that, according to these preliminary tests, the sensitivity of the method to changes of these values is acceptable as they can vary in a relatively large span without affecting the acoustic quality of the time-streched audio. For instance, the most critical parameter is probably λ and it can vary with few noticeable effects, in a range $[0.15, \ldots, 0.35]$, with a complete range of possible values $[0, \ldots, 1]$. The other parameters are set to power-of-two values in order to speed up the computations but can take other values without altering much the quality of the results. Reducing Θ too much could however limit the possibilities to find a region of $z(n)$ that overlaps with g_{g-1} well enough that the junction is inaudible.

Another fundamental parameter is C, but contrary to the preceding parameters, it has been found to be dependent from the type of sound that is time-stretched. For quieter sports such as baseball, tennis and cricket a value $15 \leq C \leq 25$ generally gives the best results, whereas for noisy sports like football and rugby, a higher value of $45 \leq C \leq 55$ gives better results. It must be noted that these values are valid only for $N_l = 8192$. Indeed, the coarseness of the spectral envelopes depends on ${}^{C}/_{N_c}$, the percentage of cepstral coefficients that are retained and N_c, the total number of cepstral coefficients, is a function of N_l, with $N_c = {}^{N_l}/_2$.

5. SUMMARY

Figure 5 summarizes schematically the ordering of each step of the method during one iteration (i.e. the life cycle of a grain g_g). After a grain is extracted from the input, it is either concatenated to the previous grain or shifted. In case it is shifted, synthetic samples are first generated and inserted to fill the gap after the previous grain and then the new grain is positioned and inserted.

[2] f_s is zero-padded to N_l samples.

[3] Note that the DFT of an even symmetrical signal is equivalent to the DCT applied to its first half. Here, we use the DCT as a straightforward way to compute the cepstral coefficients from the even $|F_s(k)|$ and $|F_l(k)|$. This use of the DCT as a cepstral operator is also encountered, for instance, when computing Mel-Frequency Cepstral Coefficients (MFCC).

[4] Actually more samples are generated, but the first one are discarded as they are tampered by the analysis and synthesis windows of the STFT.

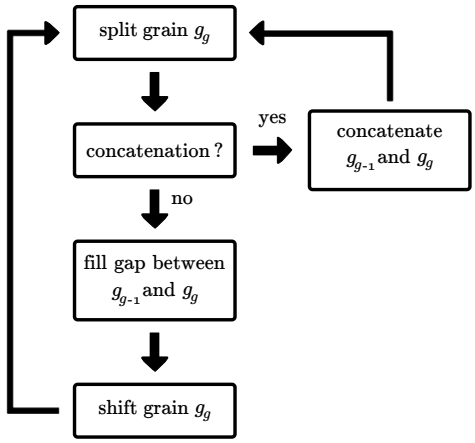

Figure 5: *Overview of the different steps of the method. As explained in Section 3, the shift step can be either a normal shift or a concatenation.*

6. DISCUSSION

The entirety of $x(n)$ is reproduced exactly into $y(n)$, grain by grain. For instance, a time scaling by a factor $\alpha = 1.25$ has 80% of its samples coming directly from the original recording. This is similar to standard time-domain algorithms. However, contrary to these, there is neither grain duplication nor change of hopsize hence no transient duplication. It means that all the transients, attacks and transitions are preserved in the time-scaled signal, while avoiding an unreliable transient detection in noisy environments.

The simplification obtained by concatenating some grains in Section 3.2 is especially useful for values of α close to one since it both reduces the computational cost and improves the acoustic quality of $y(n)$. On the one hand, for such ratios, this concatenation happens frequently and, each time, the algorithm avoid the computation of the cross-correlation $\chi(n)$ in Equation 6 and the generation of content $z(n)$ to fill a gap, as detailed in Section 4. These two operations are the most computationally intensive of the whole algorithm and the speed gain is significant, especially for values of $\alpha \leq 3$, as shown in Table 1. On the other hand, each replacement of an overlap-add by a concatenation increases the overall acoustic quality. Indeed, it reduces the number of potential discrepancies due to poor overlap-add conditions. It also suppresses cases where g_g could be shifted at a position located before $u_{g-1} + \delta_{g-1}$, when $\alpha L_{g-1} < \Delta$. In other words, where it could be inserted before g_{g-1}, thus inverting some grains in the output signal compared to the input. Besides, in the case $\alpha = 1$, the algorithm is transparent and $y(n) = x(n)$, which is not necessarily the case when only the cross-correlation-based OLA approach is used. As we can see in Table 1, for values of $\alpha \leq 1.5$, more than 50 % of the grains are directly concatenated, up to about 95 % for a speed ratio $^{25}/_{24}$. This translates into an equivalent speed up of the method as the computational cost of a concatenation is negligible.

As for the cepstrum concatenation-based cross-synthesis, note that cross-synthesis is usually [8] achieved by division and multiplication of $|F_l(k)|$ with the spectral envelopes of $|F_l(k)|$ and $|F_s(k)|$, respectively. These two approaches are mathematically equivalent, but the concatenation is obviously less computationally intensive. Indeed, not only does it replace multiplication and division operations with a simple concatenation but it also operates only one inverse DCT instead of two. Besides, in the future, it will allow more advanced combinations of the two spectral envelopes, such as linear combination of the cepstrum.

Table 1: *Average percentage of grains that are concatenated as a function of α for a 25.6 s file divided into 966 grains. Values of α are typical of slow motion sports videos and correspond respectively to $^{24}/_{25}$, 90%, 80%, $^{2}/_{3}$, 50%, 33%, 25% and 20%.*

α	1.042	1.11	1.25	1.5	2	3	4	5
%	94.7	86.2	71.8	52.7	28.7	8.3	2.3	0.5

We found a publication by Breithaupt et al. [9], in 2007, about using cepstral smoothing for speech denoising without musical noise. This cepstral smoothing is all but identical to the low-pass filtering we apply to the short-term cepstrum c_s in Equation 16. The only difference is that they apply the smoothing to the coefficients above a given quefrencies k_{low}, whereas we apply it to the ones below C. However, k_{low} is set to 4 where we use larger values of C, so all the coefficients that we smooth (and use in Equation 17) are low-passed as well in their implementation but for the first four. Note that their smoothing parameter $1 - \beta$, equivalent to our λ, is set to 0.2 (or $\beta = 0.8$), with some tests conducted at 0.3. This is coherent with the value $\lambda = 0.25$ that we selected. Note also that for some quefrencies, corresponding to the cepstrum of the pitch excitation, they use a different value of $\beta = 0.4$.

7. RESULTS

7.1. Data

We have at our disposal a database of recordings from different popular sports, namely: football (soccer), cricket, ice hockey, tennis, basketball and baseball. For each of these we have several excerpts coming from the same game. The total duration of videos in each sport varies between 200 and 1400 seconds. The audio channels are 16-bit PCM non-compressed, sampled at $F_s = 48$ kHz. The audio recordings have generally been created through professional audio mixing of one or several sources, performed by the directors during the events. These are the sounds intended for live broadcast to viewers (crowd cheers, referee whistles, on-field events, etc.), they are not the raw sounds coming directly from the on-field microphones. However, they are separate from the commenters audio recordings which are in other unused channels. For each recording, only one audio channel from the ambiance has been used during the development and tests. Time-scaling of stereo signals with preservation of spatialization and synchronization is a whole problem by itself and is not addressed in this work.

7.2. Performances

We measured the average execution time of a C++ implementation of our method for the same test file as in Table 1. Table 2 presents the average results obtained over 5 runs of the application for different common values of α. The processor is an Intel Core 2 Duo P9700 at 2.80GHz (only one core was used) and the operating system is linux (ubuntu 10.04 64 bits). The results clearly indicate that it can perform the operation in a realtime context.

Table 2: *Average runtimes of the process as a function of α for a file duration of 25.6 seconds. "% (out)" gives the relative runtimes of the process as percentages of the output duration ($\alpha \cdot 25.6$ s).*

α	1.042	1.11	1.25	1.5	2	3
$\bar{t}$ (s)	1.358	2.104	3.324	5.01	7.084	9.226
% (out)	5.1	7.4	10.4	13.0	13.8	12.0

7.3. Listening tests

We performed two series of tests. One of them measures the mean opinion scores (MOS) of the time-scaled sounds generated by either our method or a phase vocoder with automatic transient detection and processing. The other one is a comparative MOS (CMOS) between our method and that phase vocoder.

7.3.1. Phase vocoder with transient processing

The process used to handle transients is inspired from [10]. Briefly, it consists in detecting transients and processing them separately from the rest of the signal. To detect transients in a recording, the spectrum of the signal is divided into mel-spaced bands and a spectral flux is computed for each band. Then peaks in each spectral flux are identified and compared to an adaptive threshold. If a peak is above the threshold it is considered a transient. Multiple detection of a same transient across several bands[5] are merged. Each transient is then removed from the original recordings and the residual signal is time-stretched using a phase vocoder with an analysis frame length of 8192 samples (170.7 ms). Finally every transient is added into the time-stretched residual at its proper location.

7.3.2. Mean opinion scores

For each test, viewers are presented with two videos: an original sport recording and its slow motion version. They are asked to judge the quality of the audio on a scale between 0 (very bad) and 5 (excellent). The slow motion is randomly chosen among three possible speeds ($\alpha = 1.5$, 2 or 3) and the audio is processed using either our method or the phase vocoder from Section 7.3.1; which process is used is chosen randomly. Fifteen viewers participated and each took 20 tests whose results are shown in Table 3 and Figure 6. Our method obtains slightly but not significantly better scores. However, this preference decreases as α increases and for $\alpha = 3$ the two methods obtain comparable results. The

Table 3: *MOS average results (with .95 confidence intervals) for the phase vocoder (pvoc) and our method.*

MOS	overall	$\alpha = 1.5$	$\alpha = 2$	$\alpha = 3$
pvoc	3.3 ± 0.18	3.4 ± 0.25	3.6 ± 0.35	2.9 ± 0.3
ours	3.5 ± 0.16	3.9 ± 0.22	3.8 ± 0.23	2.9 ± 0.27

results considered sport-by-sport are consistent with the overall results from Table 3, with the exception of baseball which obtains an

[5]Taking into account possible minor time shifts across bands.

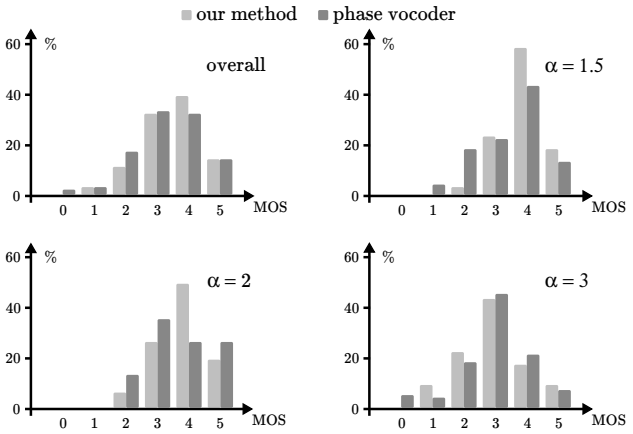

Figure 6: *Normalized histograms of the answers to the MOS tests globally and as a function of α. We can see that the results for both our method and the phase vocoder are similar, with a small overall preference for ours (more* "4"*, less* "2"*, no* "0"*). For the two methods, most of the results fall in the range 3-4 and are concentrated above the threshold of* poor quality *(2).*

over average score of 4.35 using our method and tennis which gets an under average score of 2.3 with the phase vocoder.

7.3.3. Comparative mean opinion scores

For each test, viewers are presented with three videos: an original sport recording and two slow motions A and B, with a speed factor randomly chosen among $\alpha = 1.5$, 2 or 3. One of A or B is processed using our method, the other with the phase vocoder, which one is A or B is random. Viewers are asked to choose which video has the best audio quality and how much better it is on a three-point scale (*slightly better*, *better* or *much better*). Fifteen viewers participated and each took 10 tests whose results are shown in Table 4 as a function of α.

Similarly to the MOS test, we observe on the histograms of Figure 7 that our method is globally considered as slightly better than the phase vocoder, although the mean and confidence intervals from Table 4 indicate that the advantage is only really significant for $\alpha = 1.5$. This does not seem to fit what we observe, especially in the histogram for $\alpha = 3$ which seems to exhibit a relatively strong bias in favor of our approach. The skewness parameter for that histogram, also given in Table 4 confirms the asymmetry in our favor and therefore a possible inadequation of the normal model.

Table 4: *CMOS averaged results (with .95 confidence intervals). The range of values is $-3 \leq$ CMOS ≤ 3. A positive value means that our method was preferred over the phase vocoder. Medians and skewnesses are also given as indicators of the bias of the histograms of Figure 7.*

	overall	$\alpha = 1.5$	$\alpha = 2$	$\alpha = 3$
CMOS	0.6 ± 0.25	0.9 ± 0.44	0.4 ± 0.42	0.5 ± 0.4
med.	1	1	1	1
skew.	-0.27	-0.18	-0.38	-0.53

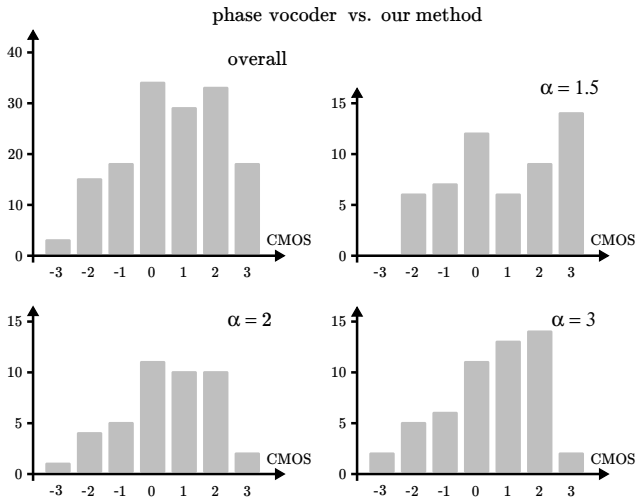

Figure 7: *Histograms of the answers to the CMOS tests globally and as a function of α. Positive values on the horizontal axis (1, 2 and 3) correspond to a preference for our approach, respectively* slightly better*,* better *and* much better*, as given by the viewers. Likewise, negative values indicate a preference for the phase vocoder. We can observe a small but clear bias in favor of our method (e.g. the small amount of -3 overall).*

Results considered on a sport-by-sport basis show a significant slight bias towards our method over the phase vocoder for football and tennis recordings. Our method is preferred for the other sports as well but not significantly.

7.3.4. Discussions

Informal discussions with viewers highlighted that, although it does not detect them, our approach handles transients much better, as expected, and is favored over the phase vocoder whenever a visually significant event (shoot, goalkeeper save, tennis service, etc.) is missed or mishandled by the transient processing integrated within the phase vocoder of Section 7.3.1. However, it can also present discrepancies, such as residual musical noise in crowd sounds or ripples in speech parts, when compared to the phase vocoder which is smoother. Note also that informal tests have shown that our method seems transparent or nearly transparent for $\alpha \leq 1.25$ whereas the phase vocoder can produce reverberation and smearing.

8. CONCLUSIONS

We presented a new method for time-scaling of audio recordings from sports events, in order to add an audio channel to slow motion videos. Tests have shown that the quality is acceptable for the viewers and similar or slightly superior to that obtained with state of the art approaches. Contrary to these, our approach is not based on sinusoidal and pseudo-periodical hypotheses and preserves transient events from the input signal without having to actually detect them. However, the results for background noises need to be improved, notably for large time-stretching factors. The algorithm is fast enough to consider a realtime interactive implementation for which various delays and the amount of samples needed in advance have to be carefully considered. However, slow motion videos are usually a playback of a past action for which all audio samples are already available. As such, the only delays to be actually considered would be the duration of each computation.

9. REFERENCES

[1] Jean Laroche, *Applications of Digital Signal Processing to Audio and Acoustics*, chapter Time and Pitch Scale Modification of Audio Signals, pp. 279–309, Kluwer Academic, 1998.

[2] C. Picard, N. Tsingos, and F. Faure, "Retargetting example sounds to interactive physics-driven animations", in *Proc. of the 35th International Conference on Audio for Games (AES)*, London, UK, February 11-13 2009.

[3] S. Roucos and A. M. Wilgus, "High quality time-scale modification for speech", in *Proc. of International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Tampa, Florida, USA, April 26–29, 1985, pp. 493–496.

[4] ITU-R Recommendation BT.1359, "Relative timing of sound and vision for broadcasting", Tech. Rep., International Telecommunication Union, Geneva, Switzerland, November 1998.

[5] S. Boll, "Suppression of acoustic noise in speech using spectral subtraction", *IEEE Trans. on Acoustics, Speech and Signal Processing*, vol. 27, no. 2, pp. 113–120, April 1979.

[6] Y. Ephraim and D. Malah, "Speech enhancement using a minimum-mean square error short-time spectral amplitude estimator", *IEEE Trans. on Acoustics, Speech and Signal Processing*, vol. 32, no. 6, pp. 1109–1121, December 1984.

[7] R. Burr and D. Lytle, "Comments on "a general method of minimum cross-entropy spectral estimation"", *IEEE Trans. on Acoustics, Speech and Signal Processing*, vol. 34, no. 5, pp. 1324–1326, October 1986.

[8] J. O. Smith III, *Spectral Audio Signal Processing*, `https://ccrma.stanford.edu/~jos/sasp/-Cross_Synthesis.html`, accessed 2012-12-03, online book.

[9] C. Breithaupt, T. Gerkmann, and R. Martin, "Cepstral smoothing of spectral filter gains for speech enhancement without musical noise", *IEEE Signal Processing Letters*, vol. 14, no. 12, pp. 1036–1039, December 2007.

[10] S. N. Levine and J. O. Smith III, "A sines+transients+noise audio representation for data compression and time/pitch scale modifications", in *Proc. of the 105th Audio Engineering Society Convention (AES)*, San Francisco, California, US, September 26–29, 1998.