Autocorrelation Method For High-Quality Time/Pitch-Scaling.

Jean Laroche

Télécom Paris
Département Signal
46 Rue Barrault
75634 Paris Cedex 13, FRANCE.
email: laroche@sig.enst.fr

ABSTRACT

In this contribution, a new method is described for high-quality time or pitch modifications of audio signals. The method is a simple but efficient improvement of the splice method. Thanks to its simplicity, the algorithm can be implemented to run in real-time on standard microprocessors. Informal listening tests have demonstrated the method's capability to modify high-quality audio signals *without introducing audible artifacts* for moderate modification factors (up to 15%).

1. Introduction

High-quality methods for time or pitch modification of audio signals have received increasing interest in recent years. Applications range from post-synchronization of video and audio to multi-track mixing. In video synchronization, one often needs to adjust slightly the duration of a recording to make it fit exactly the duration of the video clip without modifying its pitch. Acceptable duration discrepancies are usually lower than 20%. This operation requires a high-quality time-scaling algorithm. Pitch-scaling can be used to slightly adjust the pitch of a recording before integrating it into another mix. In this case, the duration of the signal must be preserved. The degree of pitch shift can be fixed, or be a function of time (e.g., to adjust the pitch of a singer). Pitch modification factors for this kind of use are usually lower than 6% (a semitone). Needless to say, for professional audio applications, time/pitch scaling algorithms must meet high-quality standards. Another desirable feature is the possibility to listen to the processed signal at the same time it is computed ('real-time' algorithms).
It is easy to see that time-scaling and pitch-scaling are dual methods. In order to change the pitch of a signal 1% up, one can extend its duration by 1%, and resample the resulting signal at a sampling rate 1% higher. Existing time-scaling methods range from very simple to rather complex algorithms. The most simple technique (on which our method is based) is the splice method, also called 'circular buffer method' due to the way it is implemented for pitch-shifting [1]. Because of its simplicity, the splice method has been used in many pitch-shifting devices even though it has a tendency to generate very audible artifacts. Linear prediction also provides a means to achieve pitch or time scaling [2], but the method works only on quasi-periodic signals, and generally does not meet high-quality standards. The short-time Fourier transform [3], and the related phase-vocoder technique [4] can also be used for time/pitch scaling. Their implementation is much more complex, but they usually give very good results, especially if small modification factors are used. Real-time implementations (at $48kHz$) require dedicated hardware. Other algorithms based on signal-models can also be used: The McAulay-Quatieri method [5] models the signal as a sum of sinusoids with time-varying parameters; the method proposed by Serra and Smith [6], is based on a decomposition of the signal into a sinusoidal part and a stochastic part. These techniques can perform extremely well even for large modification factors, but are expensive in terms of computation and lack robustness.
The algorithm presented here is based on the splice method. It is a simple algorithm which can be easily implemented on standard microprocessors and achieves very high-quality for modification factors up to 15%.

2. Presentation of the algorithm

The splice method and its limitations. In this section, we will only discuss time-scaling. As mentioned above, pitch-scaling is a dual operation and can easily be derived from time-scaling.
Modifying the duration of a signal without altering its pitch requires that some samples be 'created' (for time-expansion) or discarded (for time-compression). The splice method consists in regularly duplicating or discarding small pieces of the original signal, and using cross-fading to conceal the operation, as illustrated on Fig 1: for example, to reduce the duration of the signal by a factor α, one would discard L_s ms of signal every N input ms, thus outputting $N - L_s$ ms for N input ms, with $N/(N - L_s) = \alpha$. To avoid discontinuities in the output signal at the splice point, the first T_c ms of the discarded segment are cross-faded with the first T_c right after the discarded segment. Likewise, time-stretching by a factor α can be achieved

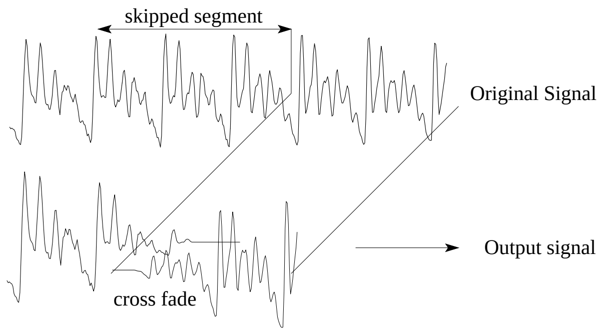

Figure 1: Time-compression of a signal by the splice method: The original signal is on top. The parts on both sides of the skipped segment are cross-faded to avoid discontinuities.

by duplicating L_s ms of the original signal every N input ms, thus outputting $N + L_s$ ms for N input ms, with $(N + L_s)/N = \alpha$. Cross-fading is needed here too in order to avoid discontinuities.
In the simplest splice method, the duration L_s of the discarded (or duplicated) segments is time-invariant, and usually chosen in the range $5ms$ to $20ms$.
The splice scheme is obviously extremely simple and easy to im-

plement on standard microprocessors. Unfortunately, it tends to generate conspicuous artifacts, mainly because the splice points and the duration of the discarded/duplicated segments are fixed parameters, and no optimization is performed with respect to the signal. Below is an example that illustrates this remark.
A speech signal sampled at $16kHz$ was time-stretched by the splice method with $L_s = 15ms$. The stretch-factor was 20% ($\alpha = 1.2$). A time domain representation of the output signal is given in Fig. 2. The output signal exhibits quite visible artifacts. The signal's quasi-periodicity has been broken in the splicing process. These artifacts, which are heard as recurring conspicuous glitches, make the standard splice method non suitable for high-quality time/pitch scaling. The problem here is that the duration of the duplicated segment is not chosen properly: In order to preserve the periodicity of the original signal, the duration of the duplicated segment should have been equal to a multiple of the pitch period.
A number of improvements have been suggested to avoid the kind of artifact mentioned above [1]. Most of them attempt to refine the selection of the splice points and of the segment duration, for example by examining the slope of the signal. We propose to use an estimate of the signal's autocorrelation to determine the best segment duration. The actual algorithm is presented below.

The autocorrelation-optimized splice method. We have seen that the crucial problem in the splice method is the choice of the segment duration. When the signal is periodic, the segment duration should be a multiple of the pitch period. When the signal is not exactly periodic, we define the optimal segment duration as *that for which the signals to cross-fade are the most similar*. In other words, when a segment of signal needs to be discarded (resp duplicated) at time t_i, we will choose the segment duration L_s for which the two signals $x(t+t_i)$ and $x(t+t_i+L_s)$ (resp $x(t+t_i)$ and $x(t+t_i-L_s)$) are the most similar (with t between 0 and T_a, T_a representing the time during which the two signals are compared). This choice is optimal in the sense that it makes the cross-fade between the two signals as transparent as possible.
In order to unify our approach, from now on positive segment duration L_s will correspond to time-compression ($\alpha < 1$) and negative L_s will correspond to time-stretching ($\alpha > 1$). This convention is coherent with the preceding equations.
To determine L_s, we calculate the unbiased signal's autocorrelation $R(\tau, t_i)$ defined by [7]

$$R(\tau, t_i) = \frac{1}{T_a - \tau} \sum_{k=1}^{T_a - \tau} x(t_i + k)x(t_i + \tau + k) \qquad (1)$$

Within a pre-defined interval $\tau_{\min}$ $\tau_{\max}$, we search for the value τ_0 which maximizes the autocorrelation. The segment duration is then set to τ_0. When the signal is time-stretched, the search is performed over negative values of τ ($\tau_{\min}$ and $\tau_{\max}$ negative). Likewise, when the signal is time-compressed, the search is performed on positive values of τ.
The autocorrelation-optimization works perfectly with periodic or nearly periodic signals, such as the one in Fig. 2. The bottom signal in Fig. 2 was obtained by the autocorrelation-optimized splice method. The signal is nearly periodic and the segment duration has been correctly adjusted to an integer number of pitch periods: The duplication is virtually transparent. In that sense, the autocorrelation method is optimal on periodic signal. On aperiodic signals, the duplication or the elimination of segments inevitably alters the original signal, but the 'optimal' choice of the segment length described above successfully conceals the alterations.

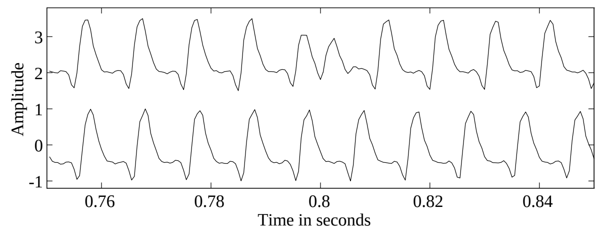

Figure 2: Time-stretching of a speech signal by the standard (top) and the autocorrelation-optimized (bottom) splice methods. Unoptimized splicing points generate visible artifacts, one of which can be seen at the center of the top signal.

Choice of the splice points. For a time-scaling factor α, t ms of input signal should correspond to αt ms of output signal for any value of t. In other word, the ideal relation between the duration t_i of the input signal and the duration t_o of the output signal should be $t_o = \alpha t_i$. Due to the way our method works (i.e. segments of signal are discarded or duplicated at discrete times,) the relation between input and output durations is not linear, but rather piecewise linear: For example, for a factor α time-stretching, from $t_i = 0$ to the first splice point, $t_o = t_i$ (since the signal is output at the same sampling rate it is input); during the duplication, L_s ms are output without using any additional input signal; from then until the next splice point we have $t_o = t_i + L_s$ and so on. Fig. 3 plots the relation between input and output times for a stretch factor of 1.5 and a fixed segment duration $L_s = 15ms$. The figure shows that a straight line with a

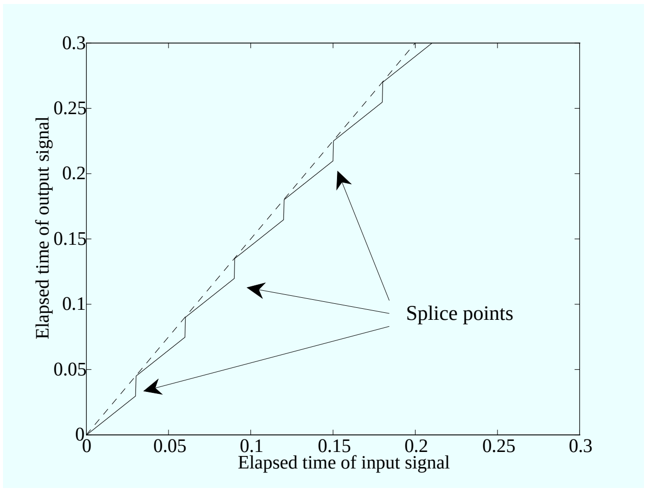

Figure 3: Relation between the input and output elapsed times (solid line). The dashed line shows ideal linear relation.

slope of 1.5 is approximated by segments of straight lines with slope 1 connected by vertical edges. The vertical edges correspond to the duplicating operations.
The discrepancy between the output elapsed time and the 'ideal' output elapsed time (the vertical distance on Fig. 3 between the solid line and the dashed line) needs to be limited to a maximum of about $50ms$. Discrepancies above that can become audible in certain kind of musical signals (e.g., music with a very precise rhythm for which the algorithm would alter the regularity of the rhythm).
In the optimized implementation of the splice method, the duration of the segment is not known in advance, and varies from splice to splice (the vertical edges of Fig. 3 have a variable length). Our algorithm keeps track of the input and output elapsed time, calculates the discrepancy and performs a splice operation whenever the dis-

crepancy reaches its maximum allowed value. The corresponding algorithm is summarized in Fig. 4

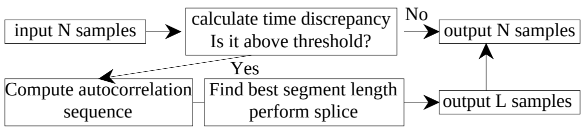
input N samples
calculate time discrepancy
Is it above threshold?
No
output N samples
Yes
Compute autocorrelation sequence
Find best segment length perform splice
output L samples

Figure 4: Diagram of the splice method with autocorrelation optimization in the case of a time-stretching operation.

Simultaneous time and pitch scaling. We now briefly describe how pitch-scaling can be done. In order to simultaneously perform a factor-α_t time-scaling and a factor-α_p pitch-scaling, the signal is first time-scaled by a factor $\alpha_p\alpha_t$ and the output signal resampled at the frequency Fs/α_p (where Fs is the original signal's sampling rate). This operation generates $\alpha_t n$ output samples for n input samples, and shifts the pitch by a factor α_p.
The simplest way to resample the output signal consists in linearly interpolating between available samples. In that case, the signal's high-frequency content is attenuated. Quadratic or cubic interpolation can be used instead to limit the high-frequency attenuation [8].

3. Adjusting the algorithm parameters

Parameter setting The optimized splice algorithm needs 6 parameters which are set as follows:

time-scale ratio	α		
maximum allowed discrepancy	$D_{\max}$	$=$	40 ms
minimum allowed segment duration	$L_s^{\min}$	$=$	10 ms
maximum allowed segment duration	$L_s^{\max}$	$=$	25 ms
autocorrelation window duration	T_a	$=$	40 ms
length of the cross-fade	T_c	$=$	30 ms

The time-scale ratio is given by the user. Its values are larger than 1 for time-stretching and lower than 1 for time-compression.
The maximum allowed discrepancy $D_{\max}$ is usually set to $40ms$. This ensures not audible time-warping will occur.
The maximum allowed segment duration $L_s^{\max}$ is set to about $25ms$. This value is a tradeoff between two considerations: When performing time-stretching, the duplication of segments of signal longer than about $25ms$ can generate artifacts, as the ear can separate acoustic events as close as 10 [9]. This is especially true when the signal includes extremely percussive sounds (e.g., high-hat, cymbals, etc) in which case the splicing operation can generate subtle but audible repeated strokes. On the other hand, we will see that longer segment durations should be favored as they minimize the number of splices per second, and therefore make the splicing process less audible. Note that a value of $25ms$ corresponds to the period of a sound with a pitch of about $40Hz$. This means than with this choice of $L_s^{\max}$, periodic sounds with a pitch lower than $40Hz$ will give birth to splicing artifacts. However, this value can be considered a reasonable lower limit for the frequency of periodic sounds in musical signals.
The minimum allowed values for the segment duration $L_s^{\min}$ is less critical: if a periodic signal has a period shorter than the minimum segment duration, the autocorrelation-optimization will select a segment duration multiple of the pitch period. However, in order to minimize the number of splices per second, we will tend to favor higher segment durations. A reasonable value is $L_s^{\min} = 10ms$.
The autocorrelation window duration T_a is set to $40ms$ (about two pitch periods for the periodic sound with the lowest allowed pitch). Its value along with $L_s^{\max}$, have a large impact on the computational cost.
The length of the cross-fade T_c is not critical and can be set to any value between 5 and $40ms$. It is reasonable to take $T_c = T_a$.

Number of splices per second. In order to determine the amount of computation needed to process 1 second of input sound, we first estimate the average number $< N_s >$ of splicing operations performed per second. Denoting $< L_s >$ the *average length of the discarded/duplicated segments*, and noting that each splicing operation generates (or eliminates) $< L_s >$ seconds of output signal in average, the duration t_o of the output signal is given by $t_o = t_i - < L_s >< N_s >$ in which $< L_s >$ is algebraic (positive for time-compression, negative for time-stretching). Since $t_o \approx \alpha t_i$, (t_i being the duration of the input signal,) we have

$$< N_s > \approx \frac{|\alpha - 1|}{< L_s >} \qquad (2)$$

We see that the longer the average segment duration, the lower the number of splicing operations per second. For a reasonable length of time and a sufficiently complex input signal, when the parameters are set to reasonable values (such as those suggested above), $< L_s >$ *is experimentally found to be very close to* $(L_s^{\min} + L_s^{\max})/2$. This remarkable fact has proved true for a large number of processed signals and seems to indicate that the average 'local pitch' of a complex musical signal is uniformly distributed on a fairly wide frequency range.
As a result, the number of splicing operation per second can be estimated as

$$< N_s > \approx \frac{2\,|\alpha - 1|}{L_s^{\min} + L_s^{\max}}$$

With the parameters selected above, $< N_s > \approx 57\ |\ \alpha - 1\ |\quad s^{-1}$. For a time-scale factor of 10%, about 6 splicing operations are performed per seconds. With this order of magnitude for $< N_s >$, the autocorrelation-optimized splice method usually generates inaudible artifacts.

4. 'Real-time' implementation

In this section, we address the problem of 'real-time' implementations of the autocorrelation-optimized splice method. The concept of real-time performance is meaningful for pitch-scaling, but is inconsistent with a time-scaling operation due to the fact that the output signal *has a different duration than the original signal*. However, we will say that an implementation achieves real-time performance when the processed sound can be heard at the same time it is processed, with a reasonable delay, say less than $10ms$.
The most computation-expensive step in the algorithm presented above, is the calculation of the autocorrelation sequence (Eq. 1). If Fs is the sampling rate in Hz and T_a is the autocorrelation window length in seconds, calculating the autocorrelation sequence $R(\tau, t_i)$ for $L_s^{\min} < \tau < L_s^{\max}$ can be shown to require a number N of multiply-add operations given by:

$$N \approx Fs^2 \left[(L_s^{\max} - L_s^{\min})(T_a - \frac{L_s^{\max} + L_s^{\min}}{2}) \right] \qquad (3)$$

For a sound sampled at $48kHz$, with the parameters selected above, the formula gives $N \approx 780,000$ multiply-adds. Since $< N_s >$ autocorrelation sequences must be calculated for each input second, this gives a total number of *4.6 million multiply-adds per second of input sound* (monophonic sound, $48kHz$ sampling rate, scale factor $\alpha = 1.1$). This order of magnitude is beyond the capabilities of most standard microprocessors. In order to achieve real-time performance, the algorithm needs to be simplified.

We propose two, possibly combined, solutions: 1) Calculate the autocorrelation on a down-sampled version of the input signal, 2) Use the Fourier Transform to calculate the autocorrelation sequence.

Down-sampled autocorrelation Eq. 3 shows that reducing the sampling rate by a factor D cuts the number of operations by a factor D^2. The idea is to down-sample the input signal by a large factor before calculating the autocorrelation. The maximum of the autocorrelation sequence is searched, and gives the optimal segment duration as described in the preceding sections. Of course, the splicing operation is performed on the signal at the original sampling rate Fs.

Note that the optimal segment length is now given as a number of samples at the sampling rate Fs/D, which means that the optimal segment length for the original signal at Fs is only known with a precision of $\pm D/2$ samples. This is not a problem in practice (at 48kHz, this uncertainty is below 1% the average length of discarded/duplicated segments for down-sampling factors up to 6):

We have found that fairly large down-sampling factors could be used without generating audible artifacts: down-sampling factors up to 6 give excellent results. The reason is that the maximum of the autocorrelation sequence depends mostly on the low-frequency content of the input, at least for the vast majority of musical signals.

Down-sampling requires low-pass filtering the signal. Because the down-sampled signal is only used to calculate the autocorrelation, frequency aliasing can be tolerated, as long as the aliased frequencies contribute less to the autocorrelation sequence than the low-frequency components. In practice, a simple D-lag comb-filter can be used to obtain the filtered signal $x_d(n)$: $x_d(n) = 1/D \sum_{i=0}^{D-1} x(n-i)$

The total number of multiply-adds per autocorrelation sequence is obtained by applying Eq. 3 to the new sampling rate Fs/D and adding the number of multiply-adds needed by the filter:

$$N_d = \left(\frac{Fs}{D}\right)^2 \left[(L_s^{\max} - L_s^{\min})(T_a - \frac{L_s^{\max} + L_s^{\min}}{2})\right] + FsT_a$$

With a down-sampling factor $D = 6$, the total number of multiply-adds per second of sound in the preceding case *drops from 5 millions to about 140,000*. This means that a microprocessor rated 2 MFlops (2 million floating point operations per second) can process a $48kHz$ stereo sound with a scale factor $\alpha = 1.1$ in real-time, using only 10% of its bandwidth (20% for a scale factor $\alpha = 1.2$). Microprocessors rated above 2 MFlops are now standard (e.g., Motorola 68040, Intel 486, Sparc TMS 390Z50 etc).

FFT implementation It is well known that the power spectrum of a signal is the Fourier Transform of its autocorrelation sequence. With certain precautions, the Fast Fourier Transform (FFT) can be used to compute the autocorrelation sequence[7]. T_a samples are selected and zero-padded to the power of two 2^m just above $(T_a + L_{\max})$. The 2^m point FFT of the zero-padded signal is computed, its square modulus is calculated, and the inverse FFT of the result is computed. The first $L_s^{\max}$ samples of the result correspond to the *biased* autocorrelation sequence $R_b(\tau, t_0)$ for $0 \leq \tau < L_s^{\max}$. The *unbiased* autocorrelation sequence is obtained by $R(\tau, t_0) = 1/(T_a - tau)R_b(\tau, t_0)$. The rest of the algorithm is unchanged.

The FFT implementation is interesting because FFTs are inexpensive in terms of computation. The operation count of the FFT implementation gives (taking $2^m \approx (T_a + L_{\max})$)

$$N_{FFT} = (L_s^{\max} + T_a)Fs\left[1 + 2\log_2(Fs(L_s^{\max} + T_a))\right]$$

which in the preceding case ($Fs = 48kHz$, mono sound) gives $N_{FFT} \approx 75,000$ multiply-adds per autocorrelation sequence. The total count is therefore about $450,000$ multiply-adds per second of sound (for $\alpha = 1.1$,) which is within reach of standard microprocessors.

5. conclusion

The autocorrelation-optimized splice method has been tested on a large number of musical and speech signals. It has shown *extremely good* performance for modification factors up to 15%: in most cases the algorithm generates virtually no audible artifact. Because the method is extremely simple, time/pitch-scale modifications with time-varying factors can be implemented in a straightforward manner, although this has not been discussed in the paper. We have found that modifications with factors above 20% tend to generate an undesirable ‘friction’ noise on certain kinds of musical signals, whereas other kinds (e.g., speech, rock music) tolerate factors up to 50 or 70%.

The method presented here is sub-optimal in the sense that only the segment durations are optimized, and not the splicing points: a better method has been tested which constantly monitors the signal’s autocorrelation for high values, and advances or delays the slicing operations to take advantage of any detected periodicity in the signal. Unfortunately, this scheme is much more expensive in terms of computation and cannot run in real-time on standard machines, although experiments have demonstrated better performance for significantly higher scaling factors (up to 50%). It remains certain that large-scale modifications require the use of more complex algorithms (phase vocoder, signal models, etc).

References

1. J. Benson. *Audio Engineering Handbook*. mcGraw-Hill, New York, 1988.
2. J. D. Markel and A. M. Gray. *Linear prediction of speech*. Springer-Verlag, Berlin, 1976.
3. M. R. Portnoff. Time–frequency representation of digital signals and systems based on short–time fourier analysis. *IEEE Trans. Acoust., Speech, Signal Processing.*, ASSP-28:55–69, Feb 1980.
4. J. A. Moorer. The use of the phase vocoder in computer music applications. *J. Audio Eng. Soc.*, 26(1), 1978.
5. R. J. McAulay and T. F. Quatieri. Speech analysis/synthesis based on a sinusoidal representation. *IEEE Trans. Acoust., Speech, Signal Processing.*, ASSP-34(4):744–754, Aug 1986.
6. X. Serra and J. Smith. Spectral modeling synthesis: A sound analysis/synthesis system based on a deterministic plus stochastic decomposition. *Computer Music Journal*, 14(4):12–24, Winter 1990.
7. A. V. Oppenheim and R. W. Schafer. *Digital Signal Processing*. Prentice Hall, Englewood Cliffs, New Jersey, 1975.
8. C. de. Boor. *A Practical Guide to Splines*. Springer-Verlag, Berlin, 1978.
9. B. C. J. Moore. *An introduction to the psychology of hearing*. Academic Press, second edition, 1982.