REALTIME-STRETCHING OF SPEECH SIGNALS

Francesco Pesce

Dipartimento di Elettronica e Informatica
Università degli Studi di Padova
amadeus@dei.unipd.it

ABSTRACT

With this work we present a new time-stretching algorithm, capable of processing high-quality digital speech signals under realtime constraints. Considering the temporally altered signal's synthesis process, the proposed method is based on a variant of SOLA, optimized for voice, with several important improvements. With these improvements we can at the same time avoid SOLA's typical artifacts and gain a reduction in computational cost of about a tenfold in the average case, which permits an efficient implementation in common PC systems.

1. INTRODUCTION

A large number of time-stretching algorithms have been proposed in recent years, either in frequency or in time domain, but only a scarce number of these are capable of running in realtime on actual Personal Computers. In realtime applications, in fact, we need a method capable of producing a temporally altered version of the input signal after a perceptually negligible delay: for this to be actually feasible, the used algorithm must have very low latency times and low computational complexity so that a signal of duration T can be elaborated in $t_c \leq T$.

It is possible to enumerate a very large number of suitable applications for such a method, but this work was born to give a powerful tool in language-impaired children's training. LLI (Low Level Impaired) children, in fact, cannot understand rapid verbal stimula because of a low level auditive deficit: in these cases, interaction with temporally altered speech is useful to solve the impairment cause. In this specific context we must satisfy strict realtime constraints to permit an efficient doctor/patient interaction and at the same time guarantee an high quality synthesis for stretching factors in the range $1 \leq \alpha \leq 3$, largely sufficient for most applications.

This work is organized as follows: in section 2 classic SOLA algorithm is presented as the starting point of our method, then in 2.1 its fundamental parameters are optimized using a spectral consistency measure; the proposed method presentation follows (in section 3), capable of overcoming SOLA's limits, in particular the artifacts that SOLA introduces when processing percussive sounds; in the final section, then, results obtained with the proposed method are compared with some of the most known digital audio processing software tools.

2. SOLA ALGORITHM

For the most part, commercial time-stretching software products used by PC users adopt the SOLA (Synchronous OverLap and Add) algorithm, which bases the synthesis phase on the sum of windowed segments of the given signal. SOLA is part of a particular class of algorithms, called *time-domain methods*, that work only with the digital waveform temporal representation. This implies that they do not need to obtain (with DFTs or time-consuming cross-correlation methods) the spectral representation of the input signal, so they synthesize the output waveform only copying and mixing, in the appropriate way, packet of samples taken from the input waveform.

From an abstract point of view, we can express the time-stretching problem as the realization of a specific *time-warping* function $\tau(m)$, which permits, when given as an input to the algorithm, to obtain a synthesized waveform $y(n)$, locally maximally similar to $x(m)$, where locally is intended in the immediate neighboring of the correspondent temporal indexes $n = \tau(m)$. This concept can be matematically expressed as:

$$\forall m: \quad y(\tau(m)+k)w(k) \rightleftharpoons x(m+k)w(k) \tag{1}$$

where $w(k)$ is a window function and the symbol '$\rightleftharpoons$' has the abstract meaning of "the most possible similar to". If we assume the "maximum similarity relation" invariant to the Fourier transform, and if we remember the definition of the Short Time Fourier Transform $X_n(\omega)$ calculated over the windowed sequence $x(m)$:

$$X(\omega, m) = \sum_{k=-\infty}^{+\infty} x(m+k)w(k)e^{-j\omega k} \tag{2}$$

we can rewrite relation (1) in the frequency domain:

$$Y(\omega, \tau(m)) \rightleftharpoons X(\omega, m) \tag{3}$$

The concept so far expressed is important to understand the conceptual basis from which the SOLA method starts (in the time domain) and how to check if a given synthesized signal $y(n)$ is a "good" time-stretched replica of the input waveform transformed by the algorithm being tested (in the frequency domain).

The SOLA method, (fig. 1), considers the input signal $x(n)$ divided in fixed-length frames of N samples each, taken every S_a samples (a quantity called *analysis hop size*); the output signal $y(n)$, on the contrary, is logically divided in frames of the same length (N samples), but synthesized with a different step (S_s, *synthesis hop size*): in this way the stretching factor is given by the ratio between the synthesis process speed (S_s samples per step) and the analysis process speed (S_a samples per step), $\alpha = S_s/S_a$. We can notice that if S_a and S_s are kept constant during the elaboration of the whole $x(n)$, the *time-warping* function $\tau(m)$ can be expressed as:

$$n = \tau(m) = \frac{S_s}{S_a} m = \alpha m \tag{4}$$

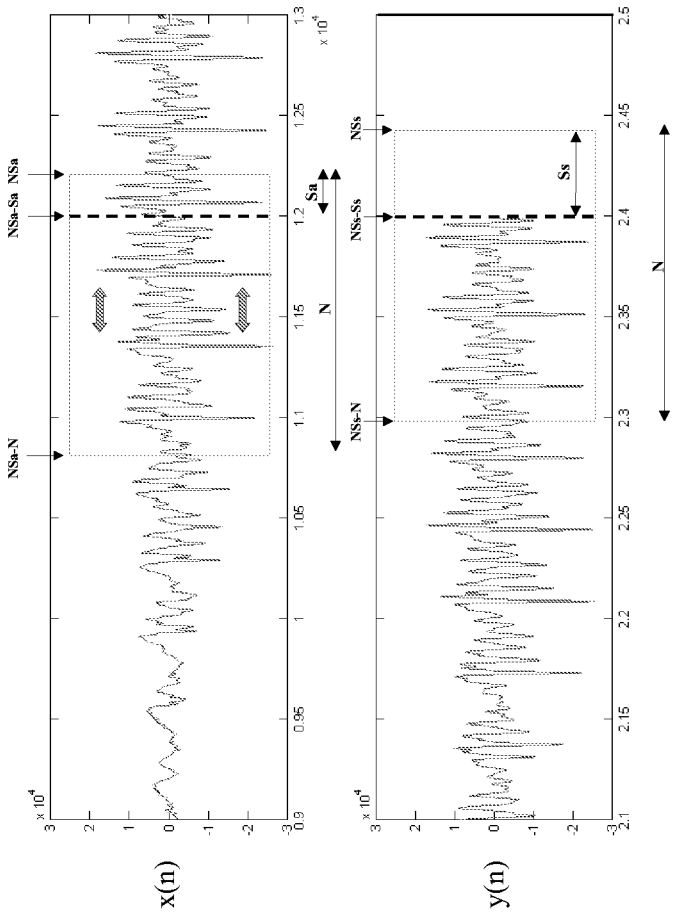

Figure 1: The generic SOLA algorithmic step. In this figure we can graphically visualiza the algorithm's work: the signal's logical division in N-sample frames is shown together with the values of all the parameters involved in the stretching process. From the two different time scales, expressed in sample numbers, we can also notice that $\alpha = 2$, $S_a \approx 220$ samples $\equiv 5$ ms.

In general, α can be time-varying (if the application requires this) and $\tau(m)$ can be described with more complex equations.

After an initialization phase, where SOLA copies the first N samples from $x(n)$ to $y(n)$, to obtain a minimum set of samples to work on,

$$y(0 \dots N-1) = x(0 \dots N-1) \qquad (5)$$
$$NS_a = N + S_a, \quad NS_s = N + S_s$$

during the generic step the algorithm examines an N-samples frame of $x(n)$, from $NS_a - 1$ towards the past of the signal, or $x(NS_a - N \dots NS_a - 1)$, and the temporally correspondent frame of the partially synthesized signal, $y(NS_s - N \dots NS_s - 1)$, where the last S_s samples are still to be reconstructed, because in the previous step NS_s was S_s samples before the actual S_s. While the synthesis frame is kept fixed, the analysis one is repositionable in a given range, and can be more precisely expressed as $x(NS_a - N + k \dots NS_a - 1 + k)$, with $-k_{max} \leq k \leq k_{max}$. The most important work for the algorithm is now to find the best overlap between the two frames to satisfy the (1). When this optimal overlap if found, it copies the last S_s samples from the so best-positioned input frame, $x(NS_a - N + k_m \dots NS_a - 1 + k_m)$, to the end of the current output frame and mixes the $(N - S_s)$ preceding from each frame with a *linear crossfade* to obtain a gradual transition to the new packet of samples:

$$y(NS_s - S_s + j) = x(NS_a - S_s + k_m + j) \qquad (6)$$
$$0 \leq j < S_s - 1$$
$$y(NS_s - N + j) = (1 - f(j))y(NS_s - N + j) + \qquad (7)$$
$$+ f(j)x(NS_a - N + k_m + j)$$
$$0 \leq j < N - S_s - 1$$

where $f(j)$ (*smoothing function*) is a linear slope.

The final effect of this method is a local replication of the waveform's periods (more or less evident depending from the value of parameter α), located by a *synchronization function*, which finds during each step, as stated, the optimal overlap between frames; this local replication of waveform's periods permits us to obtain a synthesized waveform with the same spectral properties, but with an altered temporal evolution.

Synchronization

To find the optimal value for k_m, assuring the best match between the two N-sample frames, we can use one of the three most diffused techniques:

1. Computation of the minimum vectorial inter-frame distance in L_1 sense (cross-AMDF)
2. Computation of the maximum cross-correlation $r_{x,y}(k)$
3. Computation of the maximum normalized cross-correlation $r_N(k)$, where every value taken from $r_{x,y}(k)$ is normalized dividing it by the product of the frame energies (one of which is varying with k).

Even if the last technique is conceptually preferable, in this work, for computational efficiency, we will use the second, which, under appropriate conditions explained in next paragraph, allows us to obtain results substantially similar to maximum $r_N(k)$ calculation:

$$r_{x,y}(k) = \sum_{i=0}^{L_m} x(NS_a - N + k + i)y(NS_s - N + i) \qquad (8)$$
$$-k_{max} \leq k \leq k_{max}, \quad L_m \doteq N - S_s$$

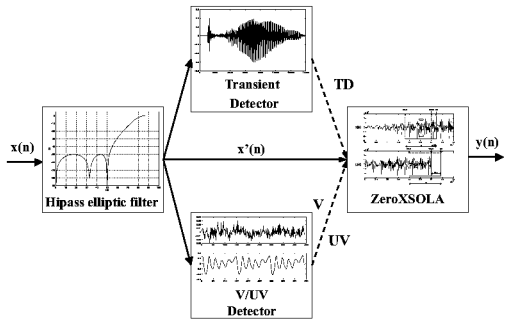

Figure 2: Block-scheme of the proposed algorithm

2.1. Parameter optimization

In a recent work, ([1]), J. Laroche and M. Dolson proposed an useful metric to measure time-stretching algorithms' synthesis quality:

$$D_M = \frac{\sum_{u=P}^{U-P-1}\sum_{k=0}^{N-1}[|Y(\alpha t_a^u,\omega_k)| - |X(t_a^u,\omega_k)|]^2}{\sum_{u=P}^{U-P-1}\sum_{k=0}^{N-1}|X(t_a^u,\omega_k)|^2} \quad (9)$$

This value is an average spectral distance measure between the Short Time Fourier Transform taken on input signal, $X(t_a^u,\omega_k)$, and the output signal's STFT, $Y(\alpha t_a^u,\omega_k)$, centered on *temporally corresponding* instants. As this value (expressed in dB) becomes more negative, the distances between correspondent spectra decrease, and the synthesis quality consequently increases.

In this work we will adopt D_M as a quality measure, following the operative indications proposed in [1], trying to determine optimal values for parameters S_a and N, under the following ipothesis: ($F_c = 44.1$ kHz, $b+1 = 16$ bit, $1 \leq \alpha \leq 3$).

As far as S_a is concerned, D_M computation on different sample speech phrases allows us to find an all-purpose value $S_a \equiv 5$ ms. When α is near 1, optimal values tend to migrate towards higher values (> 10 ms), but the 5 ms value is necessary for consistent stretching factors (near $\alpha = 3$), where synthesis quality becomes critical, and must be carefully regarded. To allow the use of constant-length data structures, which are particularly efficient, we have chosen to utilize the fixed value $S_a \equiv 5$ ms, independently from the stretching factor α.
The value of N, on the contrary, depends from α, following the law $N \equiv (20 + 5\alpha)$ ms. To explain how we obtained this dependency, is sufficient to observe how, during synchronization phase, the algorithm really uses only $(N - S_s)$ samples taken from $y(n)$ to find the best match: a necessary condition to successfully find the exact optimum is the presence of at least one pitch period in this interval. We can imagine to satisfy this condition, even in presence of very low male speech (fundamental frequencies near 70 Hz), imposing a 20 ms extension for such interval:

$$N - \alpha S_a \equiv 20\ ms \quad \Rightarrow \quad N \equiv (20 + 5\alpha)\ ms \quad (10)$$

This value, obtained in a teorical way, is largely confirmed by spectral-consistency measures.

3. THE PROPOSED ALGORITHM

Now we present the proposed algorithm (**TvdZeroXSOLA** - Transient and Voice Detecting, Zero Crossing Synchronous OverLap and Add), examining its block-scheme (fig. 2).

It is important to underline that this algorithm was obtained using several optimisations which assume a particular class of input signals: the monophonic speech waveforms. This method, in fact, was developed to obtain good results detecting and taking advantage of some properties of the vocal signals, such as "stable" harmonic spectra during vowels, high-frequency spectral contents of fricative consonants, sudden power increases on impulsive events (occlusives), and so on. It can also be used with other classes of input signals, but the results are worser if the signal is far from having time and frequency caracteristic similar to the human voice. These are the most important features the algorithm is expected to present:

- Realtime-compatible computational cost (with common Personal Computers)
- High fidelity to consonantic articulations, avoiding to introduce artifacts and transient smearing
- Preservation of occlusive transients, often echoed by SOLA
- High spectral stability of vowels

Keeping in mind these features, we present TvdZeroXSOLA's block scheme, discussed together with the architectural and computational motivations of each block's algorithmic solution.

Hipass filter
The input signal is first depurated of its spectral components falling under the vocal band (< 80 Hz, microphonic *rumbles* and common mode noise) to carefully preserve the signal's zero-crossings, a fundamental caracteristic to ensure proper functioning of following blocks. A first compromise is present in this module: it is important to preserve the spectral content of very low frequency components, such as dark vowels ("O", "U") pronounced by a very low-pitch male speaker, but there is an objective difficulty in designing high-quality filters where the ratio between stop-band and pass-band is sharply unbalanced (0.0038 in our case, using $f_{low} = 80Hz$). We cannot use linear-phase FIR filters, because even with 1000 coefficients the frequency response is far from the required shape, so the best choice is to use a fifth-order elliptic IIR filter, with stop-band $0Hz \leq f_{stop} < 85Hz$, attenuation $A_{dB} > 50dB$ and dB ripple $\leq 0.1dB$:

$$H(z) = G\ \frac{1 + b_{1,1}z^{-1} + b_{1,2}z^{-2}}{1 + a_{1,1}z^{-1} + a_{1,2}z^{-2}}\ \frac{1 + b_{2,1}z^{-1} + b_{2,2}z^{-2}}{1 + a_{2,1}z^{-1} + a_{2,2}z^{-2}}\ \frac{1 + b_3 z^{-1}}{1 + a_3 z^{-1}}$$

with gain G=0.982 and the following coefficients:

cell	$i=1$	$i=2$	$1=3$
$b_{i,1}$	-1.99994743995	-1.99998255906	-0.99999753613
$b_{i,2}$	1.00000000000	1.00000000000	
$a_{i,1}$	-1.99714732809	-1.98555631403	-0.98000158149
$a_{i,2}$	0.99728179642	0.98576624951	

The poles/zeros configuration of this filter is particularly critical concerning the noise arising from the use of finite-precision arithmetic; this noise is infact greatly magnified by the contribution of the poles, all near $|z| = 1$. To avoid this potential signal degradation, the filter is implemented using floating-point arithmetic: this design decision permits to leave the arithmetical

noise far from the 16-bit sampling boundary, but introduce an additional cost of 0.35 MFLOPS. This additional cost is not an issue with modern Floating Point Units, capable of performances at least two orders of magnitude above this value, and in general is not the bottleneck of these algorithms, which spend a consistent part of their time in the sinchronization process, requiring above 3 M.operation/s.

The filtered signal is then sent to *Transient Detector* and (potentially in parallel) to *Voice Detector* blocks, which, on the basis of parameters extracted from each frame, influence the synthesis process (ZeroXSOLA block).

Transient Detector

Individuation of occlusive transients plays a fundamental role during speech analysis, because it allows to recognize and isolate events of limited duration (approx. 10 ms for a 't', for example), which contain a particular information, strongly time-dependent in its perceptual meaning. This time-dependency must be preserved in order to ensure the correct interpretation of the occlusive consonant, so the segment of signal individuated by the "transient detector" cannot be time-stretched: it is simply copied into the output signar, with no further modification. We can notice that this allows us to overcome SOLA's most dangerous artifact: the presence of echoed transients in syntesized signal, caused by summation of transient's overlapped copies (this is the major drawback of each OverLapAdd method, caused by the basic method's "engine" itself). Potentially there is also a little drawback when detecting and copying transients: if the transient detector is not sufficiently selective and robust, it can circumscribe (as transients) a considerable amount of the input signal, forcing the synthesis process to skip large portions of signal, and so loosing much of the time-stretching factor. In fact, if we call l_T the ratio between the transient part and the total signal, we can see that, given an input stretching factor of α, the effective output stretching factor is

$$\hat{\alpha} = \frac{l_T + \alpha(1 - l_T)}{l_T + (1 - l_T)} = \alpha - l_T(\alpha - 1) \qquad (11)$$

if, for example, $\alpha = 3$ and $l_T = 25\%$, we obtain $\hat{\alpha} = 2.5$, with an error of 17% on the desired value.

Observing equation 11, one can think to pre-correct the input stretching factor of an appropriate quantity, in order to obtain the right output α:

$$\alpha = \hat{\alpha} = \beta - l_T(\beta - 1) \quad \Rightarrow \quad \beta = \frac{\alpha - l_T}{1 - l_T} \qquad (12)$$

so, if we want to expand with $\alpha = 2$ a signal covered by the 10% of transients, we must impose $\beta = 2.11$ on the input. Clearly, this correction process is applicable only if we know *in advance* the amount of "transients" (l_T), so only *offline*. Having to operate in a realtime context, it is impossible to know in advance how to correct the input stretching factor in order to obtain the right output expansion. With the proposed implementation of Transient Detector, which we now discuss, measurements conducted over a quite large database of speech samples show that, when simply neglecting this error, the difference between the desired stretching factor and the effective one is on the average case below the 5%. As far as realtime environment is concerned, we can neglect this problem.

The basic idea is to circumscribe regions where signal's power presents sudden increments, and let them untouched to preserve the complete perceptual meaning of these particular events. The

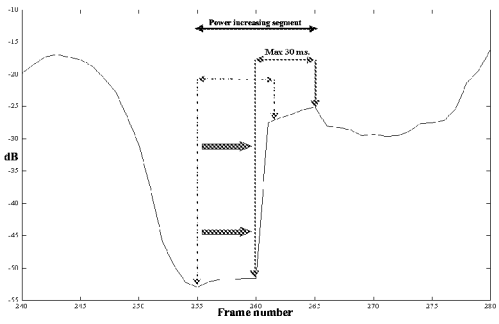

Figure 3: Individuation of an occlusive 'c'

proposed algorithm operates in three steps (fig. 3): first it calculates input signal's power profile, by means of integration over 25 ms frames, taken every 5 ms (S_a), then it individuates transients where signal's power grows at least 20 dB in at most 30 ms; finally, transient's estimation window is centered on the maximum-pendence interval over the power-increasing region, and all the underlying segment is simply copied into the output signal by ZeroXSOLA.

Voice Detector

The *Voice Detector* module is responsible of capturing fricative unvoiced consonants, sounds consisting for the most part of high-frequency noise, which can be synchronized with a simplified method, as we will see, speaking of ZeroXSOLA. These sounds take their origin for the most part from turbolencies of the air passing through the vocal chords, the teeth and the lips; none of them is put in vibration, so there is no pitch and no low-frequency resonances: the resulting sound is very similar to high-frequency shaped random noise.

A very straighforward method to detect sounds naturally configured as high-frequency noise can be computing the ratio beetween the signal's energy over 3 kHz (which is a typical boundary for human voice) and under 2 kHz on the given frame. If this ratio is above an appropriate value (obtained in a sperimental way, for example) the corresponding signal frame is classified as fricative unvoiced "noise". This method is clearly correct from a theoretical point of view, but it is also quite inefficient, because it requires two parallel filters to separate the two frequency bands and two integrators to compute the low and high-frequency energies.

An alternative and very efficient method is simply counting the signal's zero crossings in the given frame. If this value is above a well-experimented value, the frame is classified as fricative unvoiced, or as voiced in the opposite case. To understand why this method is equally correct from a theoretical point of view, we can simply remember that an high number of zero crossings is caracteristic of a signal with a considerable high-frequency energy content: drastically simplifying, it is easy to verify that a simple sinusoid presenting $n \geq 5$ crossings/ms (of both pendences) has a periodicity $T_s \leq 0.5$ ms, and so frequency $f_s \geq 2$ kHz.

The proposed algorithm operates dividing the input signal in 5 ms (S_a) frames and counting all signal's zero crossings, of both pendences, into each frame: if the resulting value is above the threshold of 5 crossings every millisecond, the frame currently under estimation is considered part of a fricative sound, or vocalized in the opposite case. The proposed boundary value of 5

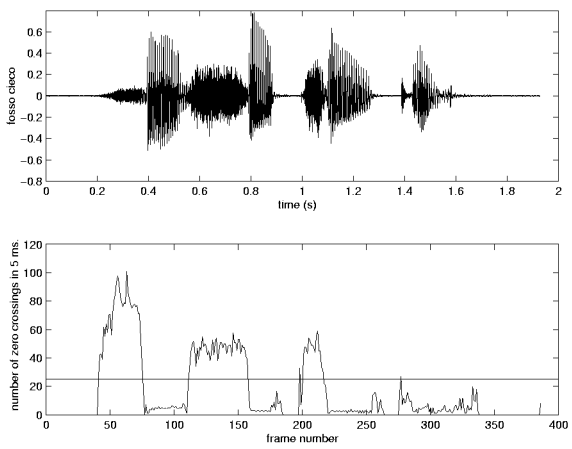

Figure 4: The italian words 'fosso cieco' and the corresponding curve describing the zero crossing density on 5 ms frames. It is immediately evident the very well defined gap between fricative consonants, where the crossing density gets very high values, and wovels, where the curve goes below the traced boundary.

crossings/ms is, as already noted, the correspondent of the 2 kHz frequency "break" in the parallel filters method, and it is largely confirmed from experimental results (see fig. 4).

Synthesis block - ZeroXSOLA

The synthesis block, ZeroXSOLA, heart of the whole algorithm, receives the data stream to be processed from the hipass filter and the informations about transients and fricative frames from the Transient Detector and the Voice Detector modules, in order to adapt its behavior on the basis of the signal's analyzed characteristics. This block is a strongly modified version of classical SOLA, with a fast and accurate synchronization phase, which can save a considerable amount of computational power (above a factor of 100 on vocalized sounds!) without loosing accuracy in finding the best overlap between input and output frames.

The basic idea (fig. 5) is to perform synchronization only in correspondence of the signal's zero crossings: we avoid the computation of $r_{x,y}(k)$ for each $-k_{max} \leq k \leq k_{max}$ firstly aligning the synthesis frame towards the nearest crossing, and then, during the maximum $r_{x,y}(k)$ search phase, aligning the analysis frame on subsequent zero crossings with the same sign, limiting the search to the interval $-k_{max} \leq k \leq k_{max}$, as usual; we then compute $r_{x,y}(k)$ only for these values of k, adding at most the nearest ones to improve the algorithm's strength in presence of noise. In this way, we compute *only the most significative cross-correlations*, that is to say the cross-correlation values that have higher probabilities to be optimal: as demonstrate the spectral consistency measures that are reported at the end of the next section, this method has the great advantage of consistently lowering computational costs (above a factor of 100 in highly vocalized frames) without affecting the synthesized signal's quality. In figure 5 is shown ZeroXSOLA's behavior when synchronizing a male-speech vowel: in this lucky situation we compute only two cross-correlations every 350 samples, one of which is the searched maximum.

The synthesis algorithm behavior depends on three conditions:

(1) If the current frame is part of a transient, the corresponding signal's segment is simply copied, as we have already seen. Note that Transient Detector detects a transient if there is a power increase of at least 20 dB in at most 30 ms: given that we use 5 ms frames, ZeroXSOLA copies at most 6 frames leaving them untouched, plus a margin of $\lfloor \alpha - 1 \rfloor$, useful to avoid the risk of having echoed copies of the transient near the last copied frames, due to the subsequent mixes performed in the following synthesis steps.

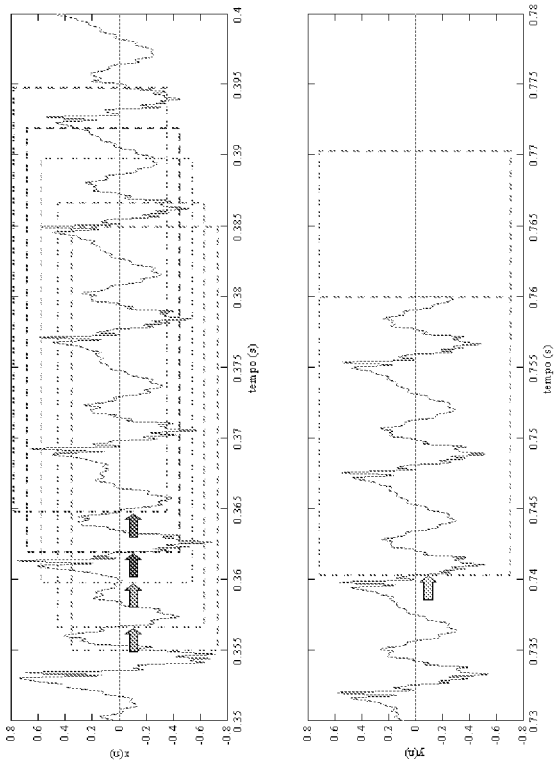

Figure 5: Synchronization of a vowel with ZeroXSOLA.

(2) If the current frame is part of a vocalized sound, we perform synchronization with ZeroXSOLA over a sufficiently extended interval, imposing $k_{max} \equiv 6$ ms, value needed in order to have at least one pitch period in the search region. The proposed synchronization phase implementation is divided in two steps: first we compute a first approximation of the cross-correlation values in the immediate neighboring of the zero crossing points:

$$r_{x,y}(k) = \sum_{i=0}^{L_m/p} x(NS_a - N + k + i * p)y(NS_s - N + i * p)$$
$$-k_{max} \leq k \leq k_{max}, \quad k \in \{crossings\} \pm \{1..\gamma\}$$

In this way we do not calculate the complete cross-correlation values, but only a $1/p$ fraction: if p/N sufficiently small, the results obtained have sufficient significance for this first phase (with values of $p/N < 0.01$, D_M index changes are negligible). Then the research is refined computing the complete cross-correlation values in a restricted interval around the position of the maximum

found during the first step:

$$r_{x,y}(k) = \sum_{i=0}^{L_m} x(NS_a - N + k + i)y(NS_s - N + i)$$
$$k_{M,1} - \delta \leq k \leq k_{M,1} + \delta$$

where γ and δ values depend on the signal pendency in the crossing point, on the input signal to noise ratio and on other contributions, but with experimental measurements on D_M, we found a good all-purpose couple of values: $\gamma \equiv 70\mu s$, $\delta \equiv 20\mu s$.

(3) Finally, if we are in presence of a fricative and unvoiced frame, we can consistently reduce the search interval to $k_{max} \equiv 0.2$ ms, thanks to the spectral caracteristics presented by fricative sounds, or better we can also drastically simplify the synthesis phase choosing casually as optimal one of the nearest zero crossings, because high-frequency noise is completely aperiodic and consequently scarcely cross-correlable. Measures of D_M conducted over many sample speech phrases demonstrate that these two different options produce fairly identical results.

4. RESULTS

We have measured both the synthesis quality and the computational time required to expand by different stretching factors in the range $2 \leq \alpha \leq 3$ several vocal samples, some directly recorded with a microphone, some extracted from commercial CDs and some found on the Internet. Measurements on D_M were conducted using the same settings described in [1]:

DFT frame length:	1024 samples	(23.2 ms)
STFT step in $y(n)$:	256 samples	(5.8 ms)
STFT step in $x(n)$:	$\frac{256}{\alpha}$ samples	($\frac{5.8}{\alpha}$ ms)
FFT Window function:	Hamming=$(0.54 - 0.46\cos(2\pi\frac{i}{1024}))$	

We present now the results obtained expanding with $\alpha = 2$ two speech samples found on the Internet (fonts: www.enounce.com, www.prosoniq.com) using four different softwares: SMS 1.01 by Xavier Serra, implementing the Sinusoidal Model of Sound, Wavelab 2.01 (© Steimberg 1998) and Cool Edit Pro 1.1 (© Syntrillium 1998), two SOLA's implementations, and PPL 1.0, implementing the proposed algorithm in two different versions, with or without Transient Detector module, to underline the importance of a differentiated processing of occlusive speech segments.
In fig. 6, we can appreciate the proposed method's superiority, both in synthesis quality and in computational timings terms.

5. CONCLUSIONS

With this work we have proposed a new time-stretching algorithm, optimized for speech signals. Compared with other existent solutions, ([6], [7]), our method uses a more refined approach in recognizing transients, without requiring high computational resources as other implementations do ([7]). Thanks to the zero-crossing synchronization process, then, a significant reduction of computational complexity was possible, obtaining an implementation capable of running in a real-time environment on economic home-PCs.

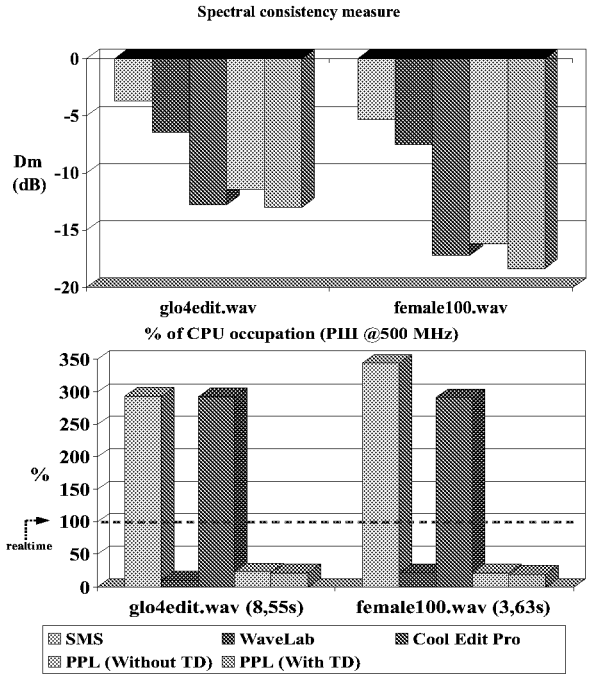

Figure 6: Results on sample phrases ($\alpha = 2$).

6. REFERENCES

[1] J. Laroche, M. Dolson, "Improved Phase Vocoder Time-Scale Modification of Audio", IEEE Transactions on ASSP, vol. 7, pp. 323-332, 1999

[2] S. Roucos, A. M. Wilgus, "High Quality Time-Scale Modification for Speech", IEEE ICASSP 1985 pp. 493-496, 1985

[3] J. L. Wayman, D. L. Wilson, "Some Improvements on the Synchronized-Overlap-Add Method of Time Scale Modification for Use in Real-Time Speech Compression and Noise Filtering", IEEE Transactions on ASSP, vol. 1, pp.139-140, 1988

[4] E. Hardam, "High Quality Time-Scale Modification of Speech Signals using Fast Synchronized-OverladAdd Algorithms", IEEE ICASSP 1990, vol. 1, pp. 409-412, 1990

[5] W. Werhelst, M. Roelands, "An Overlapp-Add Technique based on Waveform Similarity (WSOLA) for High Quality Time-Scale Modification of Speech", IEEE ICASSP 1993, vol 2, pp. 554-557, 1993

[6] R. Ren, "An Edge Detection Method for Time-Scale Modification of Acoustic Signals", www.cs.ust.hk/∼rren/sound_tech/ TSM_Paper_Long.htm, 1998 (Unpublished)

[7] S. Lee, H. D. Kim, H. S. Kim, "Variable Time-Scale Modification of Speech using Transient Information", IEEE ICASSP 1997, vol. 2, pp. 1319-1322, 1997