6th European Conference on
Speech Communication and Technology
(EUROSPEECH'99)
Budapest, Hungary, September 5-9, 1999

A NOVEL HIGH QUALITY EFFICIENT ALGORITHM FOR TIME-SCALE MODIFICATION OF SPEECH

B. Lawlor[1] *and A. D. Fagan*
DSP Research Group
Dept. of Electronic & Electrical Engineering
University College Dublin, Ireland
rlawlor@faraday.ucd.ie
http://wwwdsp.ucd.ie/

ABSTRACT

We present a novel efficient algorithm for time-scale modification (TSM) of speech which gives output quality equal to that of a conventional TSM algorithm, but having computational load an order of magnitude less. The algorithm presented uses a fixed length rectangular stepping window and a simple peak alignment criterion to track the local natural scaling factor and adapt the window step size. The desired TSM factor is realised by the appropriate number of applications of the constantly varying local natural scaling factor. The local natural scaling factor estimate is updated at sub-pitch period intervals giving accurate pitch tracking and high quality in the output scaled signal.

Keywords**:** Speech, Time-Scale Modification (TSM), Synchronized Overlap-Add (SOLA), Adaptive Overlap-Add (AOLA)

1. INTRODUCTION

Much work has been done in the area of time-scale modification (TSM) of audio in general and of speech in particular. Speech related applications of TSM include speech synthesis - based on acoustical unit concatenation [1], voice transformation - e.g. making a female voice sound male [2], speech compression [3-5], foreign language learning, audio-typing and the training thereof, voice mail speed up / slow down, speech recognition and more.

In 1944 Dennis Gabor [4 part 3], working on speech compression, patented a kinematical frequency converter which scaled the frequency of an audio signal without affecting its duration. In a subsequent paper, [5 part 2], Gabor published details of a process which he called 'frequency compression with self-adjusting scale' which was basically a pitch-tracking overlap-add algorithm. Many TSM algorithms in practical use today have much in common with Gabor's approach.

Laroche [6] reviewed the many approaches to speech TSM and categorized them from crude but simple splicing methods to high quality but computationally intensive methods based on decomposing the signal into a sinusoidal part and a stochastic part. Among the various approaches, there is a clear tradeoff between computational load and output quality and improving one without compromising the other remains a challenge. The Synchronised Overlap-Add (SOLA) [7] algorithm appears to be the most popular method for speech related applications due to its good output quality combined with moderate computational load.

2. SYNCHRONIZED OVERLAP-ADD (SOLA)

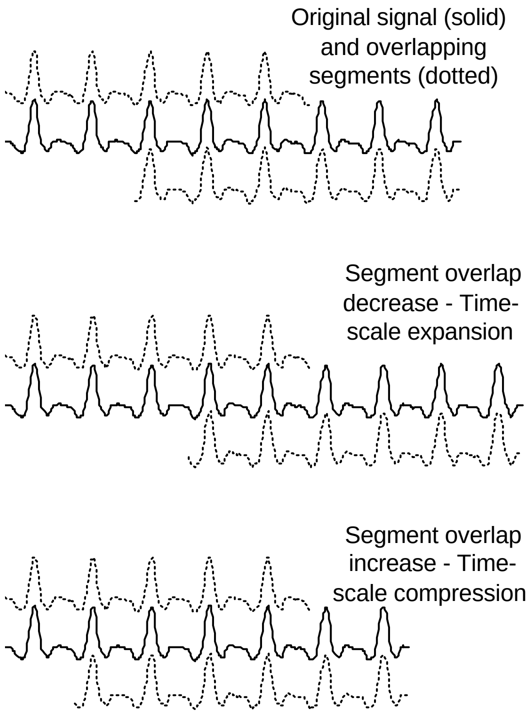
Original signal (solid) and overlapping segments (dotted)
Segment overlap decrease - Time-scale expansion
Segment overlap increase - Time-scale compression

Figure 1. Synchronised Overlap-Add

In 1985 Roucos [7] presented the SOLA algorithm for TSM of speech. Although this approach was based on an earlier joint time-frequency domain algorithm [8], it was a simplification of that algorithm and can be described independently. The SOLA algorithm uses overlapping frames (a frame being typically several pitch periods in duration) of the original speech signal.

[1] This Research was funded by the Dublin Institute of Technology.

By increasing the amount of overlap between successive frames, time-scale compression is achieved. Similarly, by decreasing the amount of overlap, time-scale expansion is achieved, as shown in *Figure 1*.

The SOLA algorithm uses a normalized cross-correlation search technique to find the best point at which to overlap the frames. Roucos used a weighting function (linear or raised cosine) to combine the overlapping frames. Although the SOLA algorithm involved multiple cross-correlation calculations, it was computationally less intensive than the joint time-frequency domain algorithm which it was based on. The local 'best match' search feature of the SOLA algorithm means that the TSM factor is allowed to vary from frame to frame giving high-quality modified speech.

2.1 SOLA Computational Load Estimate

For the *m*-th frame, the SOLA normalized cross-correlation measure, R_m, is computed for a range of possible frame alignment offsets, *k*, and can be expressed as :

$$R_m(k) = \frac{\sum_{j=0}^{L-1} y(mS_s + k + j)x(mS_a + j)}{\sqrt{\sum_{j=0}^{L-1} x^2(mS_a + j)\sum_{j=0}^{L-1} y^2(mS_s + k + j)}} \quad (1)$$

where *x(n)* represents the sampled signal to be time-scaled and *y(n)* the time-scaled signal. S_a represents the analysis interframe interval or step-size. If the time-scale factor is α, then $S_s = \alpha S_a$. In (1), *m* represents the frame number and *L* represents the length of the overlapping portions of *x(n)* and *y(n).* The best alignment offset, k_m, depends on the nature of the input signal. This means that the computational load associated with realizing some desired TSM factor, α, is also signal dependent making it difficult to estimate. Nonetheless, if certain reasonable assumptions are made, a rough estimate of the load can be calculated. In practice, a simplified normalized cross-correlation coefficient is used given by:

$$R'_m(k) = \frac{\sum_{j=0}^{L-1} y(mS_s + k + j)x(mS_a + j)}{\sum_{j=0}^{L-1} \left|x(mS_a + j)\right| \sum_{j=0}^{L-1} \left|y(mS_s + k + j)\right|} \quad (2)$$

giving a significant saving in the required number of multiply operations.

Referring to *Figure 1*, we assume that the analysis frame default step-size is equal to half the frame length, i.e. *N*/2, such that the dotted segments in the upper plot have a 50% overlap. For time-scale expansion, we assume that the alignment offset search range is from the 50% overlap to zero overlap. That is, the lower dotted segment moves to the right such that by the end of the search its left edge coincides with the right edge of the upper dotted segment. As the search range length equals *N*/2, the average overlap during each alignment offset search equals *N*/4. For time-scale compression, the lower segment is moved to the left and we assume that in this case the search range is from 50% overlap (*N*/2) to 100% overlap (*N*). Hence, in this case, the average overlap during each alignment offset search equals 3*N*/4. The main SOLA computational load is due mainly to the normalised cross-correlation feature. In (2), each denominator ∑ term requires one add operation for each step of the alignment offset search. One multiply operation per step is required to compute the product of the denominator ∑ terms and one divide operation to divide the denominator into the numerator. We count these as two multiply operations per step (a divide being similar to a multiply in computational load). Note that for compression, each denominator ∑ term requires an extra *N*/2 additions due to the larger overlap. The numerator of (2) can be computed efficiently using an FFT-based fast convolution algorithm [9]. Having computed the *N*/2 cross-correlation terms, the maximum one must be found in order to identify the optimum alignment offset. This requires *N*/2 comparison operations. Another element of the computational load is that associated with combining the overlapping segments. Typically, a linear or raised cosine function is used to weight the overlapping segments prior to adding them to ensure a smooth transition. If we set the cross-fade length equal to our average expansion overlap estimate, (*N*/4), then the number of multiply operations required to combine the overlapping segments becomes 2×*N*/4 = *N*/2 and the number of additions becomes *N*/4. The cross-fade transition length can be made smaller than the overlap length in which case these measures fall accordingly. For compression, although the average overlap is 3*N*/4, we assume for efficiency that the cross-fade is limited to the length-*N*/4 portion in the centre of the overlapping region. The TSM factor, α, also has a bearing on the computational load, for example, for time-scale expansion, $\alpha > 1$, the overall number of overlap-adds needed to realise the necessary expansion increases with α. Similarly, for time-scale compression, $\alpha < 1$, the number of overlap-adds decreases with alpha. Hence, the computational load is roughly proportional to α. Based on the above assumptions, *Table 1.* shows the resulting computational load estimates [10].

3. ADAPTIVE OVERLAP-ADD (AOLA)

Referring to *Figure 2*, the solid trace of plot (a) represents a rectangular windowed segment of the input signal. For voiced speech, the window length, *w*, is chosen such that it will accommodate at least two cycles of the lowest likely fundamental component.

For unvoiced speech, the choice is not critical and *w* can be left equal to the value chosen to satisfy the above voiced condition. We used *w* = 50 mS. Assuming we wish to scale the duration of this segment by some desired expansion factor, *de*, the steps involved in the algorithm are as follows:

1. The windowed input segment (a) is duplicated and the duplicate aligned with (a) as shown in plot (b). The alignment criterion is based on aligning the two largest peaks or troughs.
2. A synthetic segment, (c), is produced by fading gradually from (a) to (b) in the overlapping region. The natural expansion factor, *ne*, is given by the ratio of the lengths of (c) and (a) as indicated.
3. The rectangular window is stepped forward in time by *st* = |CD| = *w.(1-ne)/(1-de).*
4. The new step-size segment of the original is concatenated with (c) as indicated dotted such that the next segment to be expanded is the length-*w* portion of (c) above BD. Repeat from step 1 until the end of the input signal is reached.

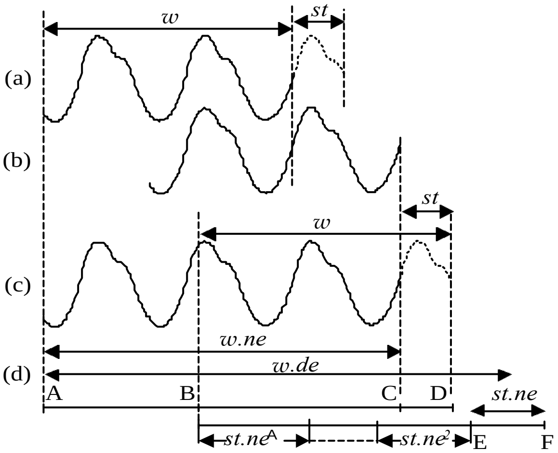

Figure 2. AOLA Time-Scale Expansion

Rationale: Line (d) represents the desired length to which we wish to expand (a), i.e. *w.de*. The segment of (c) above AB has been scaled by the desired factor, *de*, and is output. For each step of the window we repeat the peak search and update *ne*. Assuming *ne* to be approximately equal to its last value, segment BD of (c) expands in the same way as (a) to length |BF| ≈ *w.ne*. Step size portion CD of (c) expands to length |EF| ≈ |CD|*.ne,* but we require it to be expanded by factor *de*. To achieve this we must apply our natural expansion factor *A* times such that ne^A = *de*. If segment CD is to have *A* applications of natural expansion factor, *ne,* before leaving the expansion window, then our step size, *st,* must satisfy the following equation

$$st.ne + st.ne^{2} + \Lambda + st.ne^{A} \approx w.ne \qquad (3)$$

$$\Rightarrow \; st \approx w\,.\frac{1 - ne}{1 - ne^{A}} = w\,.\frac{1 - ne}{1 - de} \qquad (4)$$

As *ne* is continuously varying, (3) (and (4)) is an approximation. In fact, each of the *ne* and *st* terms in (3) are slightly different. By updating *ne* and hence *st* for every step of the window, the algorithm accurately adapts to the local signal characteristics.

For time-scale compression the approach is similar. In this case the peaks or troughs are aligned as before but the sections of (c) to the left and right of the central overlapping section are discarded leaving a naturally compressed segment. If the input segment has a natural compression factor, *nc*, and the desired compression factor is *dc*, then equation (4) becomes

$$st = w\,.\frac{1 - nc}{1 - dc} \qquad (5)$$

In order to relate the AOLA computational load to that of the SOLA algorithm, we assume *w* = *N* sample periods and that the cross-fade length is equal to *N*/4 as before. In practice, *w* < *N,* but provided that the percentage of the overlapping region used for cross-fading is the same for both algorithms, the cross-fade computational loads will also be approximately equal. For the AOLA algorithm, we need to find the two largest samples within an *N*-sample window. This requires 2*N* comparison operations. The estimates of the number of multiply (MUL), add and compare (Com) operations associated with the AOLA algorithm are shown in *Table 1*.

	a	Mul	Add	Com
S	>1	a$N(log_2N + 3/2)$	a$(N((3/2)log_2N + 3/4) \quad 2)$	a$N/2$
	<1	a$N(log_2N + 5/2)$	a$(N((3/2)log_2N + 13/4) \quad 1)$	
A		a$N/2$	a.$N/4$	a$2N$
A/S	>1	$1/2(log_2N + 3/2)$	$\approx 1/(6\ log_2N +3)$	4
	<1	$1/2(log_2N + 5/2)$	$\approx 1/(6\ log_2N +13)$	
A/S	>1	1/19 = 5.3 %	1/51 = 2.0 %	4
	<1	1/21 = 4.8 %	1/61 = 1.6 %	

Table 1. AOLA vs SOLA Computational Load Comparison: S = SOLA, A = AOLA

The lower A/S column shows the ratios of the AOLA to SOLA computational load estimates for a typical frame (or window) length *N* = 256 sample periods.

4. RESULTS

The AOLA algorithm was applied to a selection of speech signals for TSM factors in the range 0.5 to 2.0. These signals were also scaled using a commercially available SOLA algorithm and the results compared by informal listening tests.

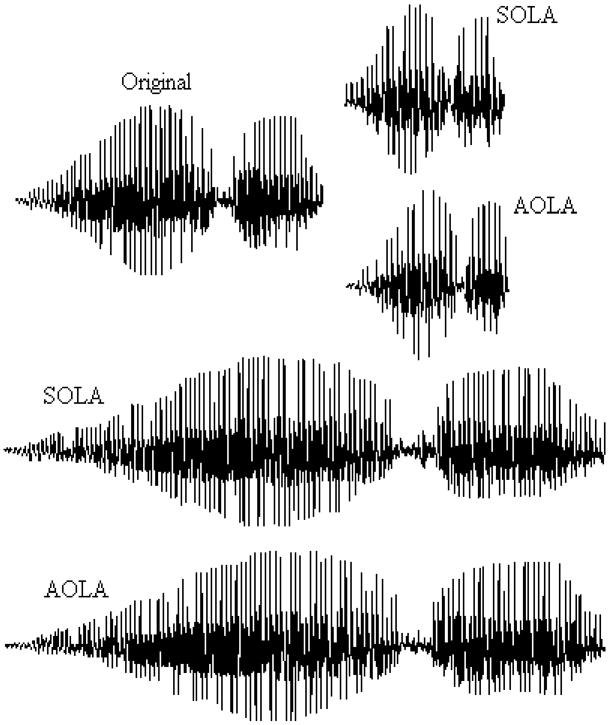

Figure 3. TSM results; a *= 0.5 and 2.0.*
Utterance: water from TIMIT Speech Corpus [11],
Signal name: TIMIT\TEST\DR1\FELC0\SA1.WAV

The AOLA output quality was deemed equal to the SOLA. *Figure 3* shows input and output plots of the utterance *water* extracted form one of the test signals [14]. On the CR-ROM version, the results for the complete SA1.WAV test signal can be listened to by clicking on the relevant signal in *Figure 3*.

5. DISCUSSION

The overall computational saving depends on the means of implementation. For example, a digital signal processor (DSP) can perform zero-overhead single cycle multiply, add and compare operations, i.e. a full multiply takes no longer than a simple add or compare. Hence, for a DSP implementation, the computational load ratio is equal to the ratio of the total AOLA operations to the total SOLA operations. For N = 256, this ratio is approximately equal to 12.1% for a > 1 and 10.9% for a < 1.

Using an ASIC implementation, the add and compare operations can be performed faster than the multiplies. For example, for a 16-bit word size, the computational load associated with a multiply operation is roughly equivalent to 16 add operations. In this case, the computational load ratio can be estimated by weighting the multiply estimates by 16 and then computing the ratio of total AOLA to SOLA operations. For N = 256 this is approximately 6.2% for a > 1 and 6.1% for a < 1. Hence, to realise the full computational advantage of the AOLA algorithm, an ASIC implementation is recommended.

6. CONCLUSION

By using a simple peak alignment criterion to track the local natural scaling factor and adapt the window step size, any desired TSM factor can be realised by repeated application of the local natural scaling factor. The approach removes the need for a computationally intensive search without sacrificing output quality.

7. REFERENCES

[1] Dutoit, T. (1994), High quality text-to-speech synthesis: a comparison of four candidate algorithms. *IEEE International conference on acoustics, speech and signal processing.*

[2] Atal, B. S. and Hanauer, S. L. (1971) Speech Analysis and Synthesis by Linear prediction of the Speech Wave. *Journal of the Acoustic Society of America*, Vol. 50, No.2 (Part 2) pp. 637 - 655.

[3] Lee, F. F. (1972) Time compression and expansion of speech by the sampling method. *Journal of the audio engineering society.*

[4] Gabor, D. (1946) Theory of communication, *Journal I.E.E.* Vol. 93, pp. 429 - 457. Part 1. The analysis of information, Part 2. The analysis of hearing, Part 3. Frequency compression and expansion

[5] Gabor, D. (1947) New possibilities in speech transmission. *Journal I.E.E.*, Vol. 94, No. 32, pp. 369 - 389. Part 1. An experimental frequency-convertor with fixed scale Part 2. Frequency compression with self-adjusting scale, Part 3. The phase variables of sound and their utilization.

[6] Moulines, E. and Laroche, J. (1995) Non-parametric techniques for pitch-scale and time-scale modification of speech. *Speech Communication* 16 (1995) 175-205.

[7] Roucos, S. and Wilgus, A. M. (1985) High-quality time-scale modification for speech. *IEEE proceedings on acoustics, speech and signal processing*.

[8] Griffin, D. W. and Lim, J. S. (1984) Signal Estimation from modified Short-Time Fourier Transform. *IEEE transactions on acoustics, speech and signal processing*, Vol. ASSP-32, No. 2.

[9] *Handbook for Digital Signal Processing*, Mitra, S. K. and Kaiser, J. F. (eds.), Wiley Interscience, 1993.

[10] Lawlor, B. and Fagan, A. D. (1999) A novel efficient algorithm for audio time-scale modification. Irish Signals and Systems Conference, June 24-25, NUI Galway.

[11] DARPA TIMIT Acoustic-Phonetic Continuous Speech Corpus, http://www.ntis.gov/fcpc/cpn4129.htm