HIGH QUALITY TIME-SCALE MODIFICATION OF SPEECH USING A PEAK ALIGNMENT OVERLAP-ADD ALGORITHM (PAOLA)

David Dorran, Robert Lawlor** and Eugene Coyle**

Dublin Institute of Technology*. National University of Ireland, Maynooth**.

ABSTRACT

The duration of a speech passage can be altered using audio time-scale modification techniques. Time-scale modification can be achieved in the time domain by segmenting the input signal into overlapping frames and recombining the frames with an overlap differing from the analysis overlap. We present a time-scale modification algorithm that uses a simple peak alignment technique to synchronize overlapping synthesis frames. The peak alignment overlap-add (PAOLA) algorithm also takes advantage of waveform properties to ensure a high quality output for the minimum number of iterations. The new algorithm produces a time-scaled output of approximately equal quality to that of an adaptive implementation of the commercially popular synchronised overlap-add (SOLA) algorithm, but offers a computational saving ranging from a factor of 15 (for a time-scale factor of 0.5) to 170 (for a time-scale factor of 1.1).

1. INTRODUCTION

Time-scale modification of speech allows the rate of articulation of a speech passage be increased or decreased, ideally without affecting the quality, pitch or naturalness of the original signal. This facility is useful for such applications as enhancement of degraded speech, foreign language learning and fast playback for telephone answering machines. Altering the time-scale of an audio signal can be achieved in the time domain or frequency domain, with advantages and disadvantages associated with each.

Frequency domain techniques are capable of applying high quality time-scale modifications to a variety of complex audio signals within a wide range of time-scale factors, but their versatility comes at the expense of their computational burden. Time domain techniques, although unsuited to complex audio signals, are well suited to single speaker signals. They are capable of applying high quality time-scale modifications to speech equal to that of frequency domain techniques for moderate time-scale factors ranging from 0.5 – 2.5. Time domain techniques have the advantage of being much less computationally intensive than their frequency domain counterparts.

The synchronised overlap-add (SOLA) algorithm [1] is a commercially popular time domain technique, which we summarise in section 2. Sub-section 2.1 outlines the synchronised and adaptive overlap-add (SAOLA) algorithm [2] that improves the output quality of SOLA for high time-scale factors and reduces the computational load for low time-scale factors. In section 3 we introduce the peak alignment overlap-add (PAOLA) algorithm, which offers a significant reduction in computational load on SAOLA but produces an output of approximately the same quality. Furthermore, we derive a set of equations that ensure optimum parameter choice for a given time-scale factor. Sections 4 and 5 present a comparison of SAOLA and PAOLA in terms of computational load and output quality, respectively. Section 6 concludes the paper.

2. SYNCHRONISED OVERLAP-ADD (SOLA)

SOLA [1] segments the input signal x into m overlapping frames, of length N samples, each segment being S_a samples apart. S_a is the analysis step size. The time-scaled output y is synthesized by overlapping successive frames with each frame a distance of $S_s + k_m$ samples apart. S_s is the synthesis step size, and is related to S_a by $S_s = \alpha S_a$, where α is the time scaling factor. k_m is a deviation allowance that ensures that successive synthesis frames overlap in a synchronous manner. k_m is chosen such that

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

is a maximum for $k = k_m$, where m represents the m^{th} input frame and L_m is the length of the overlapping region. k is in the range $k_{min} \leq k \leq k_{max}$.

$R_m(k)$ is a correlation function which ensures that successive synthesis frames overlap at the 'best' location i.e. that location where the overlapping frames are most similar. Having located the 'best' position at which to overlap, the overlapping regions of the frames are weighted prior to combination, generally using a linear or raised-cosine function. The output is then given by

$$y(mS_s + k_m + j) := (1-f(j))y(mS_s + k_m + j) + f(j)x(mS_a + j), 0 \le j \le L_m - 1 \quad (2a)$$

$$y(mS_s + k_m + j) = x(mS_a + j), L_m \le j \le N - 1 \quad (2b)$$

where $:=$ in equation (2a) means 'becomes equal to' and $f(j)$ is a weighting function such that $0 \le f(j) \le 1$.

A linear weighting function can be expressed as

$$f(j) = 0, j < 0 \quad (3a)$$

$$f(j) = j / (L_m - 1), 0 \le j \le L_m - 1 \quad (3b)$$

$$f(j) = 1, j > L_m - 1 \quad (3c)$$

Typically, N is in the range of 20ms to 30ms (corresponding to 320 samples and 480 samples at a sampling rate of 16kHz, respectively), S_a is in the range of $N/3$ to $N/2$ samples, k_{min} is $-N/2$ and k_{max} is $N/2$. [3] and [4] report that k_{min} can be set to 0.

2.1. Synchronised and adaptive overlap-add (SAOLA)

In general the parameters N, S_a , k_{min} and k_{max} are fixed for SOLA at algorithm development, which can be problematic. Consider the case where S_a is fixed at $N/3$, k is in the range 0 to $N/2$ and k_m for the previous iteration was 0. If $\alpha = 2$ then $S_s = 2N/3$. For this case the number of possible overlaps is limited to $N/3$ i.e. from an overlap of $N/3$ to an overlap of 1. By limiting the number of possible overlaps the output quality is degraded. It can easily be shown that the number of possible overlaps is less than $N/2$ for $\alpha > 1.5$. This problem could be alleviated by allowing k be in the range $-N/2$ *to* $N/2$. For this case, the number of possible overlaps is less than $N/2$ for $\alpha > 3$. However, the number of possible overlaps is greater than $N/2$ for $\alpha < 3$ and equal to N for $\alpha \le 1.5$. In [2] it is shown that $N/2$ possible overlaps are adequate and any number greater than this increases the computational load unnecessarily. From above, S_s should ideally be $N/2$ for all α, allowing $N/2$ possible overlaps for all α, when k is in the range of $N/2$ to 0. SAOLA [2] achieves this by allowing S_a be adaptive i.e.

$$S_a = N/(2\alpha) \quad (4)$$

This result also has the effect of reducing the number of computations required for low time-scale factors.

3. PEAK ALIGNMENT OVERLAP-ADD (PAOLA)

The PAOLA algorithm operates in a similar manner to SOLA except that it uses a simple peak alignment technique to ensure synthesis frames overlap in a synchronous manner. PAOLA also takes waveform properties into consideration to provide a high quality output and to perform the minimum number of iterations for the desired time-scale factor. The adaptive overlap-add (AOLA) algorithm [5] also uses a peak alignment technique, but differs from PAOLA in implementation, with PAOLA offering a reduction in computational load.

For the m^{th} iteration, the PAOLA algorithm first searches the current output for the maximum peak $y_m(p_y)$ in the region $y_m(M_m - j),\ 0 \le j < SR$, where M_m is the length of the current output y_m after m iterations and SR is the length of the search region. Next, the maximum peak $x_m(p_x)$ is found in the region $x_m(j)$, $0 \le j < SR$, where x_m is m^{th} input frame and is given by

$$x_m = x(mS_a + j),\ 0 \le j < N. \quad (5)$$

The m^{th} input frame is then overlap-added with y_m such that the located peaks $x_m(p_x)$ and $y_m(p_y)$ are aligned producing y_{m+1}. Peak alignment is ensured by overlapping by an amount

$$L_m = p_x + M_m - p_y + 1 \quad (6)$$

The average overlap length is SR and determines the synthesis step size S_s, since $S_s + SR = N$ (see fig. 1 (b)). $S_s = \alpha S_a$ as in SOLA.

The overlapping regions of y_m and the m^{th} input frame are weighted prior to combination resulting in

$$y_{m+1}(j) = y_m(j),\ 0 \le j \le M_m - L_m - 1 \quad (7a)$$

$$y_{m+1}(M_m - L_m + j) = y_m(M_m - L_m + j)(1-f(j)) + x_m(j)f(j),\ 0 \le j \le L_m - 1 \quad (7b)$$

$$y_{m+1}(M_m - L_m + j) = x_m(j)\ ,\ L_m \le j \le N \quad (7c)$$

where $f(j)$ is a linear weighting function.

The m^{th} iteration of the algorithm can basically be thought of as overlap-adding frame m with frame $m-1$, with an overlap equal to L_m, since frame $m-1$ was overlap-added to y_{m-1} to produce y_m. This is illustrated in fig. 1 (a) and fig. 1 (b). The analysis overlap is $N - S_a$, where N is the length of the analysis frame.

Consider the case where $p_x = 0$ and $p_y = M_m$, then $L_m = 1$, illustrated in fig. 1 (c). In this case the analysis-overlapping region is almost repeated, except for one sample. For high quality time-scale modification the repeated segment should be short enough to ensure quaisi-stationarity during voiced regions, so

$$N - S_a \le L_{stat} \quad (8)$$

where L_{stat} is that length that ensures that the segment is quaisi-stationary during voiced regions. Since $N = SR + S_s$ and $S_s = \alpha S_a$

$$(\alpha - 1)S_a \le L_{stat} - SR \quad (9)$$

So,

$$S_a \le \frac{L_{stat} - SR}{\alpha - 1} \quad \text{for } \alpha > 1 \quad (10a)$$

and

$$S_a \geq \frac{L_{stat} - SR}{\alpha - 1} \quad \text{for } \alpha < 1 \tag{10b}$$

Now consider the case where $p_x = SR - 1$ and $p_y = M_m - (SR - 1)$, then $L_m = 2SR - 1$ i.e. the maximum overlap. This case is illustrated in fig. 1 (d). In this case a segment of length $S_a - (S_s - SR)$ is discarded during synthesis. For high quality time-scale modification the discarded segment should be short enough to ensure quaisi-stationarity during voiced regions so

$$S_a - (S_s - SR) \leq L_{stat} \tag{11}$$

Since $S_s = \alpha S_a$

$$(1 - \alpha)S_a \leq L_{stat} - SR \tag{12}$$

So,

$$S_a \geq \frac{L_{stat} - SR}{1 - \alpha} \quad \text{for } \alpha > 1 \tag{13a}$$

and

$$S_a \leq \frac{L_{stat} - SR}{1 - \alpha} \quad \text{for } \alpha < 1 \tag{13b}$$

Combining (10a) and (13a) gives

$$\frac{L_{stat} - SR}{\alpha - 1} \geq S_a \geq \frac{L_{stat} - SR}{1 - \alpha} \quad \text{for } \alpha > 1 \tag{14a}$$

Combining (10b) and (13b) gives

$$\frac{L_{stat} - SR}{1 - \alpha} \geq S_a \geq \frac{L_{stat} - SR}{\alpha - 1} \quad \text{for } \alpha < 1 \tag{14b}$$

The number of iterations that are executed is inversely proportional to S_a, therefore S_a should be maximised giving

$$S_a = \frac{L_{stat} - SR}{|1 - \alpha|} \quad \text{for all } \alpha \tag{15}$$

And since $N = SR + \alpha S_a$

$$N = SR + \alpha\left(\frac{L_{stat} - SR}{|1 - \alpha|}\right) \quad \text{for all } \alpha \tag{16}$$

From (16), as α approaches 1 the window length N approaches infinity. N must be limited to the length of the input x for the algorithm to operate as expected, and so for time-scale factors of 1 (and very close to 1) the output is a duplicate of the input.

The above analysis requires that the window length be at least $2SR$. Consider the case where the window length is less than $2SR$, then from (16)

$$2SR > SR + \alpha\left(\frac{L_{stat} - SR}{|1 - \alpha|}\right) \tag{17}$$

Equation (17) holds true for

$$\alpha < \frac{SR}{L_{stat}} \tag{18}$$

and for

$$(L_{stat} - 2SR)\alpha < -SR \tag{19}$$

The search region SR must contain at least one cycle of the lowest likely fundamental component to ensure that a peak exists within SR. For speech, this corresponds to about 8ms duration (128 samples for a sampling rate of 16kHz). L_{stat} is typically in the region of 19ms (304 samples for a sampling rate of 16kHz). The condition described by equation (19) has a positive solution for α only when $L_{stat} < 2SR$ and in practice this situation does not arise. The condition of equation (18) occurs for low time-scale factors i.e. $\alpha \leq 0.4$ approximately. When this condition occurs SR should be decreased or L_{stat} should be increased. Either of these operations reduces the quality of the output but ensures that the algorithm operates as expected. Intuitively, as the time-scale factor approaches zero very large segments must be discarded to achieve the desired time-scale modification.

Equations (15) and (16) provide us with the optimum analysis step size and window length to produce a high quality output for the minimum number of iterations.

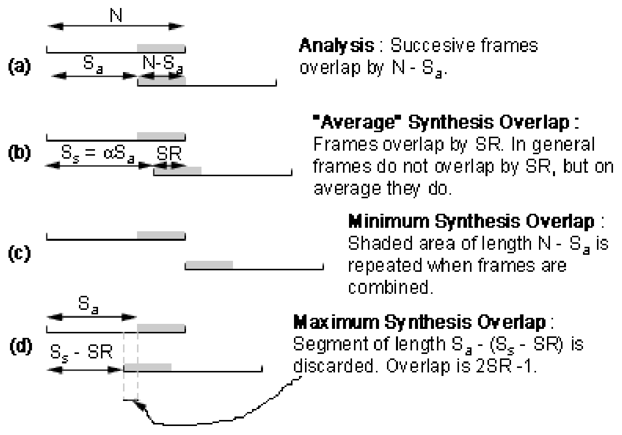
N
(a)
S_a
$N-S_a$
Analysis : Succesive frames overlap by $N - S_a$.
(b)
$S_s = \alpha S_a$
SR
"Average" Synthesis Overlap : Frames overlap by SR. In general frames do not overlap by SR, but on average they do.
(c)
Minimum Synthesis Overlap : Shaded area of length $N - S_a$ is repeated when frames are combined.
(d)
S_a
S_s - SR
Maximum Synthesis Overlap : Segment of length $S_a - (S_s - SR)$ is discarded. Overlap is 2SR -1.

Fig. 1. PAOLA Analysis and Synthesis

4. COMPUTATIONAL LOAD COMPARISON

For both SAOLA and PAOLA the number of iterations required to time-scale a signal x of length L_x is equal to the number of analysis frames m, which is given by

$$m = L_x / S_a \tag{20}$$

where S_a is the analysis step size.

PAOLA requires $2SR$ comparisons to locate the peaks in the input frame and current output per iteration. Linearly cross fading the overlapping regions requires $2SR$ multiplies and SR additions, on average, per iteration.

Using an approach similar to the one used to calculate the computational load of SOLA in [5], it can be shown that the number of computations required per iteration of the SAOLA algorithm is $(3/2)N + N\text{Log}_2(3N)$ multiplications, $(3/2)N\text{Log}_2(3N) - 4/3 + N/4$ additions and $N/2$ comparisons. Full details of the calculation of the computational load estimate for SAOLA can be found in [2].

Table 1 displays the number of computations required to implement SAOLA and PAOLA with L_x normalized to 1.

	SAOLA	PAOLA
Multiplies	$2\alpha\text{Log}_2(3\text{N}) + 3\alpha$	$\left(\frac{\lvert 1-\alpha \rvert}{L_{stat} - SR}\right)2SR$
Additions	$3\alpha\text{Log}_2(3\text{N}) + \alpha/2 + 8\alpha/(3\text{N})$	$\left(\frac{\lvert 1-\alpha \rvert}{L_{stat} - SR}\right)SR$
Comparisons	α	$\left(\frac{\lvert 1-\alpha \rvert}{L_{stat} - SR}\right)2SR$

Table 1. SAOLA and PAOLA Computational Load Estimate

As mentioned in [5], a digital signal processor (DSP) can perform single cycle multiply, add and compare operations. However, an application specific integrated circuit (ASIC) multiply operation is approximately equivalent to 16 addition operations. Therefore, to calculate the total number of ASIC operations we weight the number of multiply operations by 16.

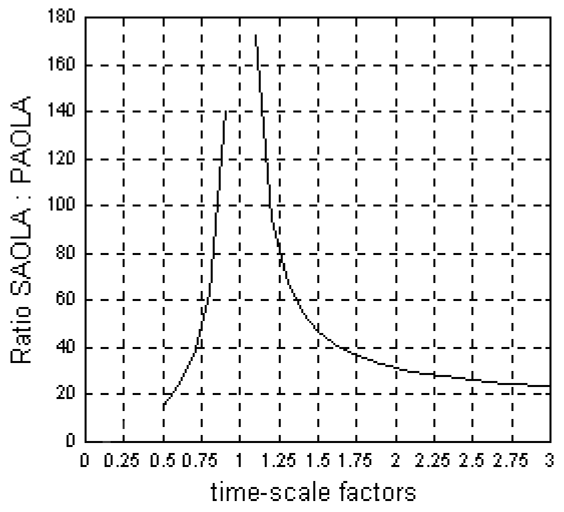

Fig. 2. Ratio of SAOLA to PAOLA Operations

Fig. 2 shows the ratio of SAOLA operations to PAOLA operations within time-scale factors ranging from 0.5 to 3 for DSP implementations. The sampling rate = 16kHz, N = 30ms, L_{stat} = 19ms, SR = 8ms. Similar plots were found for sampling rates of 8kHz and 44.1kHz, and also for ASIC implementations.

5. OUTPUT QUALITY COMPARISON

16 evaluation subjects of various age and gender carried out informal listening tests. The test comprised of 20 comparisons between a track time-scaled by PAOLA and the same track time-scaled by SAOLA, using the same time-scale factor. The subjects were not informed which track was a SAOLA time-scaled track or which was a PAOLA time-scaled track. The tests covered a selection of time-scale factors ranging from 0.5 to 3 and contained an equal number of male and female speakers. For all tests the sampling rate = 16kHz, N = 30ms, $k_{min} = 0$, $k_{max} = N/2$, L_{stat} = 19ms, SR = 8ms.

The listening tests showed that the output quality of signals time-scaled by SAOLA and PAOLA are approximately equal.

6. CONCLUSION

The PAOLA algorithm produces an output of quality approximately equal to that of the SAOLA algorithm with a computational saving ranging from a factor of 15 (for a time-scale factor of 0.5) to 170 (for a time-scale factor of 1.1), as shown in fig. 2. We also found that the PAOLA algorithm is capable of producing comprehensible speech for time-scale factors as high as 8.

7. REFERENCES

[1] Roucos S. and Wilgus A.M., "High Quality Time-Scale Modification for Speech", IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 493-496, March 1985.

[2] Dorran D., Lawlor, R. and Coyle E., "Time-Scale Modification of Speech using a Synchronised and Adaptive Overlap-Add (SAOLA) Algorithm", Submitted to the Audio Engineering Society 114th Convention 2003, Amsterdam, The Netherlands.

[3] Hardam, E., "High quality time scale modification of speech signals using fast synchronised-overlap-add algorithms", Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing, Volume 1, pp. 409-412, 1990.

[4] Wong, J.W.C., Au, O.C. and Wong, P.H.W, **"**Fast time scale modification using envelope-matching technique (EM-TSM)**"**. Proc. of the IEEE International Symposium on Circuits and Systems, Volume 5, pp. 550–553, May 1998.

[5] Lawlor, R and Fagan A.D., "A Novel High Quality Efficient Algorithm for Time-Scale Modification of Speech", Eurospeech '99, Budapest, Hungary, September 1999.