The SOLAFS Time-Scale Modification Algorithm[1]

Don Hejna
Bruce R. Musicus

M.S. 6/4B
Bolt Beranek & Newman
10 Moulton St.
Cambridge, MA 02138
Phone: (617) 873-3309

July 31, 1991

Abstract

We present a new algorithm for time-scale modification of acoustic signals, which is similar to the Synchronized Overlap-Add Algorithm (SOLA) of [RW85, MEJ86], but with substantially reduced computational requirements. The method allows playing back pre-recorded signals at a wide variety of rates without an accompanying change in pitch. The input is segmented at a dynamic *analysis* rate into frames we call windows, which are shifted in time to maintain the desired average time-compression or expansion, and then overlap-added together at a fixed *synthesis* rate to form the output. The algorithm differs from SOLA in that it is the starting position of the windows during analysis which are shifted about their target positions in order to maximize the similarity of the windows in the overlap regions. The overlap regions in the output occur at a perfectly periodic rate. Shift prediction is used to further reduce computation.

Index Terms: Time-Scale Modification

[1]This research has been partly funded by ROLM, and in part by the Advanced Research Project Agency monitored by ONR under Contract No. N00014-85-K-0272.

1 Introduction

There are a large number of applications in which it is desirable to modify the time-scale of speech, music, or other acoustic material, without modifying the pitch. Radio stations can use the technique to speed up dance music, the blind can speed up recorded lectures, instructional material for foreign languages can be slowed down, dubbed sound tracks can be synchronized to a video signal and compressed into convenient time slots, dictation tapes can be slowed down for transcription, voice mail systems can allow listening to messages at a faster or slower rate, and so forth. The key issues in designing time-scale modification (TSM) systems are that the local pitch period remains unchanged (no Donald Duck or Minnie Mouse effects), and that no audible splicing, reverberation, or other artifacts are introduced.

One of the original attempts at TSM was by Fairbanks [?], who used a fixed rate mechanical splicing scheme to compress or expand tape recordings. Malah suggested a Time Domain Harmonic Scaling algorithm (TDHS) [Mal79], which improves on this by synchronizing the splice points to the local pitch period, and using overlap-add techniques to fade smoothly between the splices. Griffin and Lim [GL84] introduced a frequency-domain approach which iteratively synthesizes a signal whose spectrogram is a compressed or expanded version of the original signal's spectrogram. Though this technique works well on almost any acoustic material, an enormous amount of computation is required. Roucos and Wilgus [RW85] originally introduced the Synchronized Overlap-Add (SOLA) method as a quick initial guess for the Griffin and Lim algorithm. In practice, however, this method worked so well that further processing was unnecessary.

SOLA starts by cutting the input signal $x[n]$ into possibly overlapping windows $x_m[n]$ with a fixed length W, and separated by a fixed *analysis* distance S_A.

$$x_m[n] = \begin{cases} x[mS_A + n] & \text{for } n = 0, \ldots, W-1 \\ 0 & \text{otherwise} \end{cases}$$

The windows are then overlapped and recombined, with the separation between them compressed or expanded, so that on average the windows are separated by a new *synthesis* distance S_S. The ratio $\alpha = S_S/S_A$ gives the desired compression or expansion rate with $\alpha > 1$ corresponding to expansion and $\alpha < 1$ corresponding to compression. The synthesis shift used for each window $x_m[n]$ is allowed to vary by an amount k_m in order to maximize the similarity of the data in the overlapping regions before the overlap-add step. Thus the output is formed recursively by:

$$y[mS_S + k_m + n] \leftarrow \begin{cases} \beta_m[n]y[mS_S + k_m + n] + (1 - \beta_m[n])x_m[n] & \text{for } n = 0, \ldots, W_{OV}^m - 1 \\ x_m[n] & \text{for } n = W_{OV}^m, \ldots, W - 1 \end{cases}$$
$$\text{where: } W_{OV}^m = k_{m-1} - k_m + W - S_S$$

and where the shift k_m is selected to maximize the similarity (e.g. cross-correlation or average magnitude difference) in the overlap region between the current output $y[n]$ and the window $x_m[n]$. $\beta_m[n]$ is a fading factor (e.g. averaging or linear fade) which should be chosen to minimize any audible splicing artifacts.

The difficulty with the SOLA algorithm is that the amount of overlap W_{OV}^m between the output and the newest window varies with k_m, and this complicates the work required to compute the similarity and to fade across the overlap region. Also, depending on the shifts k_m, more than two windows may overlap in certain regions, which further complicates the averaging computation.

2 SOLAFS

In this paper we introduce a new variation of SOLA, which we call *Synchronized Overlap-Add, Fixed Synthesis* (SOLAFS). The key idea is that the windows are overlap-added into the output at a fixed *synthesis* rate, while the starting positions of the windows $x_m[n]$ are adjusted during *analysis* in order to maximize the similarity between the output and the new window in the overlap region. Thus during *analysis*, the windows are chosen as:

$$x_m[n] = \begin{cases} x[mS_A + k_m + n] & \text{for } n = 0, \ldots, W-1 \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

and the output is recursively formed by:

$$y[mS_s + n] \leftarrow \begin{cases} \beta[n]y[mS_S + n] + (1 - \beta[n])x_m[n] & \text{for } n = 0, \ldots, W_{OV} - 1 \\ x_m[n] & \text{for } n = W_{OV}, \ldots, W - 1 \end{cases} \tag{2}$$

add [illegible]

where $W_{OV} = W - S_S$ is the number of points in the overlap region. Note that the shift k_m now affects where the window starts in the input stream. The optimal shift is determined by maximizing a similarity measure between the overlapping points in x and y. A similarity measure which works well in practice is the normalized cross-correlation between x and y in the overlap region:

$$k_m \leftarrow \max_{0 \le k \le K_{max}} R^m_{xy}[k] \tag{3}$$

where K_{max} is the maximum allowable shift from the window's initial starting position, and

$$R^m_{xy}[k] = \frac{r^m_{xy}[k]}{\left(r^m_{xx}[k]\right)^{1/2}\left(r^m_{yy}\right)^{1/2}}$$

where

$$\begin{aligned} r^m_{xy}[k] &= \sum_{n=0}^{W_{OV}-1} x[mS_A + k + n]y[mS_S + n] \\ r^m_{xx}[k] &= \sum_{n=0}^{W_{OV}-1} x^2[mS_A + k + n] \\ r^m_{yy} &= \sum_{n=0}^{W_{OV}-1} y^2[mS_S + n] \end{aligned}$$

[illegible]

$$= x[mS_A + \tau_m + n]$$

$$\text{where: } \tau_m = k_{m-1} + S_S - S_A$$

Suppose $0 \le \tau_m \le K_{max}$. Then when we cross-correlate the last W_{OV} points of the output $y[mS_S+n]$ with the first W_{OV} points of possible analysis windows $x[mS_A+k+n]$, we must find the maximum to be at $k_m = \tau_m$. With this offset, the output and input points in the overlap region are identical, and the normalized cross-correlation is 1. Thus the m^{th} shift, k_m, should be determined by:

$$k_m \leftarrow \begin{cases} \tau_m = k_{m-1} + (S_S - S_A) & \text{if } 0 \le \tau_m \le K_{max} \\ \max\limits_{0 \le k \le K_{max}} R^m_{xy}[k] & \text{otherwise} \end{cases} \tag{4}$$

Furthermore, if the m^{th} shift is predictable, then the averaging in (2) is unnecessary since the points overlap-added together are identical. The input can simply be copied into the output stream. In effect, shift prediction behaves like a *modify on demand* system, since splicing and overlap-adding will only be necessary if the predicted shift τ_m falls outside the allowable range $[0, K_{max}]$. For mild compression or expansion, with $S_S \approx S_A$, most of the shifts will be predictable, and only occasional splicing will be necessary to modify the time-scale.

4 Simplifying the Shift Calculation

The bulk of the computation in SOLAFS revolves around computing the normalized cross-correlation $R^m_{xy}[k]$ and picking the maximum. This can be simplified by a number of tricks. To avoid the square root, we could instead choose k_m by:

$$k_m \leftarrow \max_{0 \le k \le K_{max}} \frac{r^m_{xy}[k] \left| r^m_{xy}[k] \right|}{r^m_{xx}[k] r^m_{yy}} \tag{5}$$

or even more simply:

$$k_m \leftarrow \max_{0 \le k \le K_{max}} \frac{r_{xy}^m[k]\,|r_{xy}^m[k]|}{r_{xx}^m[k]} \qquad (6)$$

Further simplifications result by computing $r_{xx}^m[k]$ recursively:

$$r_{xx}^m[k+1] = r_{xx}^m[k] + x^2[mS_A + k + W] - x^2[mS_A + k]$$

Both methods (5) and (6) give precisely the same answer as (3). Method (6) uses the least computation, since the constant r_{yy}^m is not used and thus need not be computed. Methods (3) and (5), on the other hand, are always scaled so that their magnitudes are less than or equal to 1. This may be convenient in a fixed-point implementation. For all three approaches, with fixed point arithmetic care must be used to avoid overflow when computing the cross-correlations r_{xy}, r_{xx}, and r_{yy}.

Careful attention should also be paid to the buffering of data. SOLAFS requires a W_{OV} length output buffer to hold the last points of the output, $y[mS_S], \ldots, y[mS_S + W_{OV} - 1]$, and a $W + K_{max}$ length input buffer to hold the input data points that might be used in the next analysis window, $x[mS_A], \ldots, x[mS_A + W + K_{max} - 1]$. Beware that in a real-time application, time-scale compression will require reading in input data at a much faster rate than usual. This may cause difficulties if the data is stored in compressed form and must be decoded, or if the storage unit is slow.

5 Parameter Choices

Large shifts S_S, S_A, and windows W cause problems in time-scale modification because the signal data may change character radically between windows. Note from () that $|(S_S - S_A)|$ determines the minimum number of data points inserted or deleted when the shift predicted lies outside the range $[0, K_{max}]$. This is why small analysis shifts are beneficial in both SOLAFS and SOLA. In SOLA, the benefit in signal quality due to small analysis shifts was offset by an accompanying increase in computation. In SOLAFS, prediction allows using small analysis shifts with no significant increase in computations. Note that while the number of windows increases with decreasing analysis shift, S_A, the number of predictable shifts increases since the quantity $(S_S - S_A)$ in (4) decreases. Fortunately, shift prediction eliminates most of the computational drawbacks of using small S_S, S_A, and W values.

The window size, synthesis shift, and overlap region length are all inter-related. The computations required to determine unpredictable shift values is $O[K_{max} W_O \bigcirc\!\!\!\!V^{\overset{?}{2}}]$, and thus efficient parameter combinations will use as small $W_O \bigcirc\!\!\!\!V$ as possible. The number of overlap points W_{OV} must not be too small, or else the variance of the similarity computation will be too large and transitions between segments will be detectable. For voice-mail applications with 8KHz sampling, $W_{OV} = \underset{\approx 4ms}{30}$ samples appears to be sufficient and results in smooth transitions.

To determine an appropriate window size, start with $W = S_S + W_{OV}$. To ensure that only two windows overlap at any point in the output, we will require that $S_S \geq W_{OV}$. Thus the smallest useful synthesis shift is $S_S = W_{OV}$, and the smallest useful window length is $W = 2W_{OV}$. No deleterious effects were noted in the output produced when more than two windows overlap (multiply overlapping regions of overlap), however, no beneficial effects were noted either. The analysis shift S_A is chosen to achieve the desired compression or expansion rate. Note that non-integer values of S_A are acceptable, since S_A is only used to compute the starting point of the range of starting positions of the windows at each iteration.

The maximum shift K_{max} is an important parameter. To avoid pitch fracturing, this must be chosen to be larger than the largest expected pitch period in the input. In a voice-mail application with male speakers and 8KHz sampling, a good choice is $K_{max} = 100$ samples. This choice allows synchronization of periods down to 80 Hz when TSMing music as well. Recall that K_{max} together with S_S and S_A determines the maximum duration deleted or replicated in the output from ().

It is not necessary to choose S_A to be larger than K_{max}. However, if $S_A < K_{max}$, some care should be used to ensure that during analysis each window starts at a time no earlier than the previous window, $k_m + S_A \geq k_{m-1}$. (Modify equation (4) so that the maximum over

$R_{xy}^{m}[k]$ is computed only over the range $\max(0, k_{m-1} - S_A) \leq k \leq K_{max}$.)

6 Experimental Results

It works great! No audible artifacts! Lower computation than SOLA or TDHS! Clearly the winning technique of all!

The SOLAFS algorithm provides time-scale modified speech over a wide range of compression and expansion. With the proper choice of parameters, the TSMed speech contains no audible artifacts and preserves speaker identity. SOLAFS requires significantly less computation than SOLA or TDHS and is unaffected by the presence of white noise in the signal. The straightforward nature of the algorithm and high-quality speech produced by it indicate the SOLAFS is clearly superior to TDHS and SOLA in many applications.

7 Additional Variations

Numerous similarity measures are possible for determining the shift values. It is straightforward to extend any number of pitch-detection techniques to align the segments in the overlap region. Several methods for aligning windows are detailed in [Hej90]. It was also noted that alignment is most critical during voiced portions of speech signals. The nature of the signal in these regions, large amplitude fundamental periods make decimation and reduced shift resolution (evaluating the similarity for every other shift) possible to further reduce computations. It is also possible to overlap-add/linearly fade over more data points than are used in the similarity calculation. This allows smoother transitions without an increase in computation, but restricts the similarity measure to a fraction of the total segments to be overlap-added.

8 Acknowledgements

Much of the insight into the SOLAFS technique stems from a direct extension of knowledge gained in analyzing the behavior of the SOLA algorithm in [Hej90]. The authors wish to thank those at ROLM and the MIT VI-A Internship Program that made such research possible.

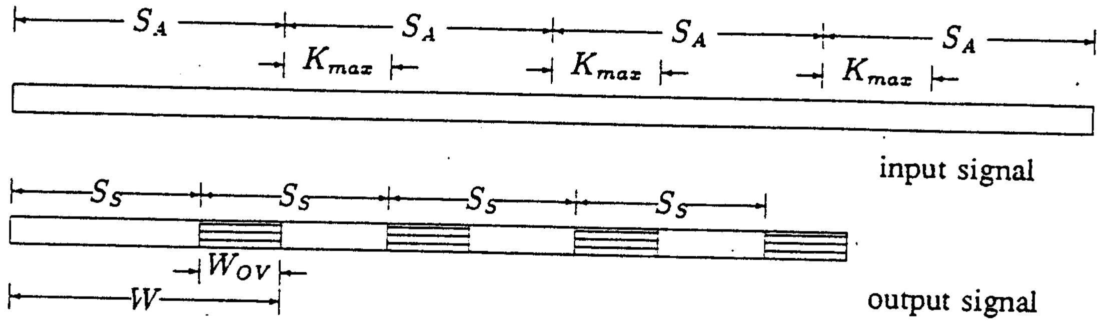

Figure 1: Operation of SOLAFS for Time-Scale Compression

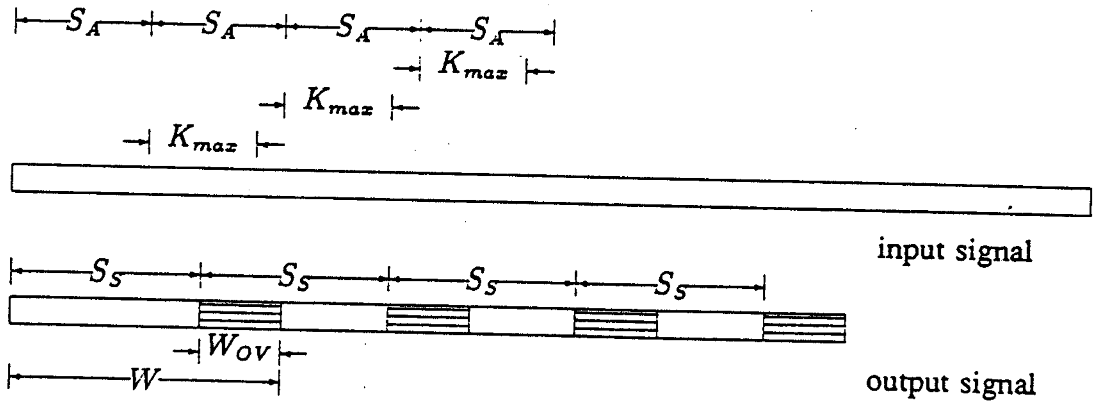

Figure 2: Operation of SOLAFS for Time-Scale Expansion

References

[GL84] D. W. Griffin and J. S. Lim. Signal estimation from modified short-time fourier transform. *IEEE Transactions on ASSP*, ASSP-32, No. 2:236–243, April 1984.

[Hej90] D. J. Hejna. Real-time time-scale modification of speech via the synchronized overlap-add algorithm. Master's thesis, M.I.T., 1990.

[Mal79] D. Malah. Time-domain algorithms for harmonic bandwidth reduction and time scaling of speech signals. *IEEE Transactions on ASSP*, ASSP-27:121–133, April 1979.

[MEJ86] J. Makhoul and A. El-Jaroudi. Time-scale modification in medium to low rate speech coding. *Proceedings ICASSP '86*, pages 1705–1708, 1986.

[RW85] S. Roucos and A. M. Wilgus. High quality time-scale modification for speech. *Proceedings ICASSP 86, Tokyo*, pages 493–496, March 1985.