Real–Time Time–Scale Modification of Speech via the Synchronized Overlap–Add Algorithm

by

Donald Joseph Hejna, Jr.

Submitted to the Department of Electrical Engineering and Computer Science

in Partial Fulfillment of the Requirements for the degrees of

Master of Science

and

Bachelor of Science

at the Massachusetts Institute of Technology

February 1990

Author _____
Department of Electrical Engineering and Computer Science
April 28, 1990

Certified by _____
Bruce Musicus
Thesis Supervisor

Certified by _____
Charles Williams
Thesis Supervisor

Accepted by _____
Arthur C. Smith
Chairman, Departmental Committee on Graduate Students

Real–Time Time–Scale Modification of Speech via the Synchronized Overlap–Add Algorithm

by

Donald Joseph Hejna, Jr.

Submitted to the
Department of Electrical Engineering and Computer Science

April 28, 1990

In Partial Fulfillment of the Requirements for the degrees of
Master of Science
and
Bachelor of Science

Abstract

This thesis explores the Synchronized Overlap-Add (SOLA) technique for real-time time-scale modification of speech in telephony and voice-mail environments. Numerous methods of reducing computational requirements and the associated effects on quality are detailed. The performance of various windowing and update functions are compared. It quantifies many of the undesirable effects such as reverberation, clicks, and warble associated with certain choices of parameters, and reveals parameter interactions that account for high quality output associated with low analysis shifts. A slightly modified version of the SOLA algorithm, SOLA-b, is presented which significantly simplifies implementation and can reduce computations several-fold without an accompanying loss of quality. An outline for implementation on real-time signal processing hardware is provided.

Thesis Supervisor: Bruce Musicus
Title: Associate Professor, Department of Electrical Engineering and Computer Science

Thesis Supervisor: Charles Williams
Title: Company Supervisor (ROLM Systems)

Acknowledgments

This thesis was performed at ROLM Systems in Santa Clara, California under the VI-A Internship Program at the Massachusetts Institute of Technology. The program offers students the opportunity to complete a Masters and Bachelors thesis while at the participating company under supervision of an M.I.T. professor and members of the company.

The Author wishes to thank everyone at ROLM for their enthusiasm and unending support: Micheal Agah, Tho Autrong, Andy Crowe, Valerie Dorfman, Mark Douglas, John Gladys, Dan Lai, Jerry Lin, Mike Locke, Pedro Rump, Ben Wilbanks, and Chuck Williams. I would also like to thank those at M.I.T.: Professor Bruce Musicus for many stimulating and encouraging conversations throughout the project; Kirk Johnson for his speech signal display program; Kelly Savage and Cas Wierzynski for their input to the thesis document, and the VI-A Program administrators who made this project possible.

Most importantly I would like to thank my parents whose guidance, support, and confidence in my abilities inspired me to work harder than I ever thought I could have.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Time-Scale Modification of Speech

Time-scale modification of speech refers to processing performed on speech signals that changes the perceived rate of articulation without affecting the pitch or intelligibility of the speech. Such modification can be categorized into two classes: time-scale compression (or speed-up) which increases the rate of articulation; and time-scale expansion (or slow-down) which decreases the rate of articulation. Speed-up is generally desired when a segment of speech contains little pertinent information and the goal is to extract the informational content in as little time as possible (i.e. a verbose speech) or when searching for a specific utterance quickly. Alternatively, the goal of slow-down in most cases is to decrease the rate of articulation to aid in comprehension or dictation of rapidly spoken speech segments with important information, such as an address or phone number.

A good time-scale modification (TSM) algorithm is one that produces "natural-sounding" speech over the range of playback rates that is of interest to the end user (possibly from three times slow down to two or three times speed up). Intelligibility, tonal quality, and speaker recognition should be preserved, and processing artifacts (pops, clicks, burbles, reverberation, etc.) should be kept to a minimum. A convenient method for modifying the time-scale of speech has many useful applications.

1.1.1 Time-Scale Compression and Applications

During compression the informational content (data) of the modified signal is reduced relative to the original signal, resulting in a segment of shorter duration. The goal of such modification is to increase the perceived rate of articulation without introducing undesired artifacts. Ideally the modification should remove an integer multiple of the local pitch period. These deletions should be distributed evenly throughout the segment, and to preserve intelligibility, no phoneme should be completely removed.

Compression, or speed-up, applied to normal speech would allow more information to be transferred in a given time interval than is possible by speaking quickly. This technique would allow shorter commercials, "speed-reading" for the blind, and the ability to compress movies into convenient time slots without deletions.

1.1.2 Time-Scale Expansion and Applications

During slow-down the informational content (data) of the modified signal must be increased relative to the original signal, resulting in a segment of longer duration. Such modification would slow the perceived rate of articulation. Thus stringent constraints are placed on the quality of the resulting signal. Ideally the expansion employed should insert additional pitch periods distributed evenly throughout the entire segment. This proves to be difficult, however, as the local pitch period varies across phonemes and may be difficult to gauge during nonperiodic portions of the speech signal such as fricatives.

Slow-down would increase the intelligibility of speech which is difficult to understand, or could be used to aid in the transcription of information in rapidly spoken segments. Additionally, fluent speech could be slowed to aid those learning foreign languages.

1.1.3 Data Compression Applications

The ability to perform high quality compression and expansion provides means for a time-based voice compression system. If time-scale compression could be followed by expansion without error, combining the two techniques would reduce the data required for coding and storing speech signals. This method of compression could be combined with other compression techniques to further reduce the bit rate [3][14]. Time-scale compressed speech could be coded

using alternate techniques such as vector quantization, quadrature mirror filtering, and pulse code modulation. After decoding, the time-scale compressed signal could be expanded by an appropriate factor to obtain speech with the original time-scale.

1.2 Existing Time-Scale Modification Techniques

Several algorithms have been developed to achieve time-scale modifications based on the inherent structure of the speech signal. Time-domain techniques rely on the periodic nature of speech, while analysis/synthesis techniques exploit redundancies in the signal to reduce the speech waveform into a limited set of time varying parameters.

1.2.1 Time-Domain Algorithms—TDHS and SOLA

Time-domain techniques operate by inserting or deleting segments of the speech signal. This method, first proposed by Fairbanks in the 1940's, results in discontinuities in the transitions between inserted or deleted segments. Such discontinuities lead to bothersome clicks and pops in the resulting time-scaled signal. Several attempts [4] [11] have been made to minimize the effects of inter-segment transitions in the final signal by improving the splicing technique or windowing adjoining segments. These techniques improve quality at the expense of increasing complexity.

The Time-Domain Harmonic Scaling (TDHS) algorithm [4] employs multiple correlations of signal segments to determine local pitch periods along intervals of the input signal. A triangular windowing function is aligned with the pitch periods and the resulting segments are added such that pitch periods are inserted or deleted to create a time-scale modified signal. The algorithm requires accurate pitch determination to operate successfully, thus pitch variations in the input signal must be tracked accurately. In general, the pitch period is stationary only over very short intervals and may vary drastically between phonemes. Consequently, the length of the triangular window must vary as well as the length of inserted or deleted segments, which increases complexity. Numerous methods for determining pitch in the time-domain have been forwarded [8][7][12][10].

TDHS provides good quality in the class of low complexity time-domain algorithms. It

uses pitch-synchronous windowing intervals and variable length windowing functions to reduce inter-segment discontinuities in the output signal. An interesting alternative is the Synchronized Overlap-Add (SOLA) algorithm originally proposed by Roucos and Wilgus [11]. The SOLA algorithm has low complexity and operates in the time-domain, but does not rely on pitch tracking. Because SOLA uses fixed window lengths and fixed windowing intervals, it has certain advantages for real-time implementation.

This thesis explores the SOLA technique for real-time time-scale modification of speech. The algorithm operates by segmenting the input signal, then adding these segments with different interframe shifts to construct a time-scale modified signal. The input signal segments are aligned to maximize the correlation of overlapping segments before they are added. It is this alignment that distinguishes the SOLA technique from a simple overlap-add (OLA) technique and drastically improves the quality of the output signal.

1.2.2 Frequency-Domain Algorithms

Another class of algorithms for time-scale modification of speech operate on frequency-domain representations of short segments of the speech signal known as Short-Time Fourier Transforms (STFTs). These algorithms offer distinct advantages in that they are usually very robust and perform equally well on music and non-speech signals. Many can also be shown to converge to minima of mathematical distance functions, as opposed to the more heuristic time-domain techniques. Several frequency-domain algorithms provide superior quality but are computationally intensive. For many applications however the computational burdens of these algorithms do not outweigh the benefits.

The method proposed by Portnoff [5] reconstructs a modified speech signal from modified versions of its original STFTs magnitude and phase. The technique is performed by computing STFTs of fixed duration at fixed intervals along the input signal. Using the STFT magnitude and an unwrapped STFT phase, intermediate STFTs are interpolated for expansion or deleted for compression. The inverse of each STFT generates a fixed amount of data points which are then windowed and added to form the final output. Varying the number of STFTs varies the duration of the output signal obtained. This algorithm proves prohibitive in real time implementations however, because the Fourier transform and inverse Fourier transform

operations require many computations. Also, slight errors in the phase unwrapping operation reduce signal quality substantially.

An algorithm proposed by Griffin and Lim [1] computes fixed duration STFTs along the input signal at a fixed analysis interval, and along an initial guess signal at a fixed synthesis interval. The ratio of the synthesis and analysis intervals determines the time-scale modification performed. A compressed signal will be obtained when the synthesis shift is less than the analysis shift. The algorithm then minimizes the error between the STFT magnitudes of the guess signal and the STFT magnitudes of the input signal. This is accomplished by iteratively updating the initial guess signal. At each iteration the STFT magnitude of the current guess signal is modified and the guess signal is replaced with the inverse STFT of the modified STFT magnitude and original STFT phase of the current guess signal. After numerous iterations, this method will converge to a global minimum of the mean-square error between the input signal STFT magnitudes and guess signal STFT magnitudes. It is a non-causal process in that each iteration modifies the entire signal. Again this technique provides good results but proves to be so costly that an efficient real-time implementation is not practical.

1.2.3 Analysis/Synthesis Algorithms

A third class of TSM algorithms operate by reducing the speech signal into a set of time varying parameters (analysis) which can be time-scaled and used to reconstruct a time-scale modified signal (synthesis) [6][9]. For example, a method suggested by Quatrieri and McAulay [6] assumes a limited number of sinusoids. During analysis a limited set of sinusoids is used to model the speech signal. By varying the rate at which the sequence of sinusoids is played back, the time-scale of the signal can be modified. These algorithms require less computation than frequency domain techniques, but are restricted to signals which can be represented by a limited number of time-varying parameters. Analysis/Synthesis algorithms generally perform poorly on more complex signals, such as speech corrupted by noise or containing music.

1.3 Summary

Numerous time-scale modification techniques providing superior quality are available. Frequency-domain techniques prove difficult to implement in real-time however. The class of time-domain algorithms currently available provide the most feasible real-time implementations. The SOLA algorithm appears particularly promising in its implementation.

This thesis focuses on the Synchronized Overlap-Add (SOLA) algorithm for time-scale modification of speech. It explores the SOLA algorithm as an alternative to the more computationally intensive frequency-domain and analysis/synthesis algorithms and outlines methods for improving the quality/computation ratio of the SOLA algorithm. Additionally, the effects of parameter interactions on output signal quality are examined, and a robust implementation for telephony quality speech is provided.

All tests are performed on speech which has been bandlimited to 3.8 kHz and sampled at 8 kHz. Numbers appearing in parentheses after durations in milliseconds refer to the corresponding number of samples of a signal sampled at 8 kHz unless otherwise specified.

The following outline may be used to direct the reader to specific topics of interest:

Chapter 1 presents several applications and previous methods for time-scale modification of speech.

Chapter 2 introduces the SOLA algorithm originally proposed by Roucos and Wilgus. This chapter outlines the parameters set and basic operation of the algorithm. A measure of the actual time-scale modification performed by the algorithm is given and an estimate of the computations required for shift determination is provided.

Chapter 3 provides a more in-depth look at the parameter set interaction and events which lead to undesirable artifacts in the output signal. The effects of individual parameters are provided and the proper range of values outlined. From this the interactions of the parameters are examined to provide insight into the origin of undesirable artifacts in the output signal. Equations to prevent invalid parameter sets are provided and measures of the amount of overlap and averaging defined. These equations allow the nature and extent of undesirable artifacts in the final signal to be predicted for a given choice of parameters. Additionally, predictability in the shift values for certain parameter sets is detailed and proposed as a

means of reducing the average computational requirements.

Chapter 4 provides an indication of the robustness of the SOLA algorithm by examining its performance on music, speech with multiple speakers, and speech in the presence of correlated and uncorrelated noise.

Chapter 5 presents several methods of reducing the computational requirements of the SOLA technique. The performance of the algorithm using alternate windowing, alignment, and update functions is presented and evaluated. The robustness of the original algorithm leads to several computationally improved versions which provide equal quality.

Chapter 6 introduces a slightly modified version of the SOLA algorithm. This method further simplifies implementation, while maintaining the same output signal quality over the range of desired time-scale modifications.

Chapter 7 summarizes the work performed in this study and draws various conclusions. Additionally, it presents several topics for future investigation.

Appendix 1 contains data from listen tests conducted to examine the effects of various parameter combinations.

Criteria

Due to the subjective nature of the quality measurement, implementations are compared in double-blind A-B preference tests (see [2], Appendix F). Sentences A and B are played in pairs and listeners indicate a preference for either A or B. For comparison purposes, implementation A is said to have a higher quality to computation ratio than implementation B if:

- The computational requirements of A and B are comparable, and the output of A is consistently preferred to that of B in listen tests.
- There is no preference between the output of A or B, and A requires fewer computations.

Chapter 2

The Synchronized Overlap-Add Algorithm

2.1 Introduction

The Synchronized Overlap-Add (SOLA) algorithm was developed by Roucos and Wilgus [11] in an attempt to reduce the number of iterations required for high-quality time-scale modified speech via the Griffin-Lim technique. They sought to accomplish this by providing the algorithm with an initial guess closer to the desired signal than random noise. This section gives the formal definition of the SOLA technique, introduces the parameter set for the algorithm, and provides an overview of its operation.

2.2 Origins

The Griffin-Lim technique starts with an initial guess of random noise for the time-scale modified signal. This signal is then iteratively modified until its short-time fourier transform (STFT) magnitudes taken at a fixed synthesis interval match the STFT magnitudes of the original signal taken at a different analysis interval. Roucos and Wilgus attempted to reduce the number of iterations required for high-quality output by providing an initial guess signal closer to the desired signal than random noise. The initial guess produced by one of their techniques, the Synchronized Overlap-Add, however, was reported to be of such high quality

that no iterations by the Griffin-Lim algorithm were required.

Roucos and Wilgus realized that the overlap-add step of the Griffin-Lim algorithm was working against the desired result. In greatly simplified terms the problem can be summarized as follows: If windowed frames of an input signal are taken using an interframe "analysis" shift of S_a sample points, then simply overlap-added with a different interframe "synthesis" shift of S_s sample points, periods in the overlapping portions of the signal will interfere when $S_s \neq S_a$. The interference may be constructive or destructive, and affects various frequencies differently.

The effect of a simple overlap-add with differing analysis and synthesis shifts will now be examined. Figure 2-3 illustrates a time-signal consisting of evenly spaced impulses. This signal is windowed using a fixed interframe interval, S_a, as shown in the diagram labeled *analysis windowing of input signal* of Figure 2-3. The windows of input taken with a fixed interframe interval, every S_a points, are then added with S_s points between them to decrease the time-scale of the signal ($S_s < S_a$). Note the spacing between impulses has changed in the region of overlap! The overlap-add does not preserve the periodicity of the original signal in the regions of overlap. The signal obtained differs greatly from the original input signal when $S_s \neq S_a$. The signal that results has the desired time-scale, but no longer resembles the original input. This modified signal is referred to as the *rate-modified-unshifted signal* in Figures 2-2 and 2-3.

This is where the concept of phase is important. Each windowed segment of the input signal contains impulses with their original spacing. The position, or phase, of these windows in the signal is changed however by the overlap-add at a different shift. The abrupt changes in the phase at window boundaries lead to discontinuities in the overlap-added signal (*rate-modified unshifted signal*). The local period is no longer continuous throughout the signal; i.e., the regions of overlap in the *rate-modified unshifted signal* in Figure 2-3 contain impulses whose spacing differs from the original signal.

To extend this idea one step further, suppose the impulses in the above example represent the locations of peaks in a cosinusoidal time signal. The period of the cosine corresponds to the spacing between impulses. It is now evident that discontinuities will result at each window boundary in the *rate-modified unshifted signal* in Figure 2-3. The regions of overlap will no

longer resemble the original time-signal as the differing phases of the segments cause constructive and destructive interference in the regions of overlap. Interference and discontinuities lead to perceptible harmonics, clicks, and pops in the output signal.

The interference in the signal arises from the large linear phase inconsistency between successive windows. Each window, $x_w(mS_a)$ where m is the window number, is shifted in time by $m(S_s - S_a)$ during the overlap-add step. Thus adjacent windows differ in phase by $(S_s - S_a)$ samples. Correcting this phase inconsistency by simply shifting each window, $x_w(mS_a)$, by an amount $-m(S_s - S_a)$ returns each window to its original position and results in the original input signal. To modify the time-scale of the signal, a different interframe shift must be used. If the signal contains a single cosinusoid, then each window may be shifted by modulo $\frac{m(S_s - S_a)}{\tau_m}$ (where τ_m is the period of the input signal) without altering the periodicity of the signal. The shift of individual windows will vary, but over the entire signal the average interframe synthesis shift will be S_s. Were the period of the signal to vary slowly, windows could be aligned in this same manner, provided τ_m reflects the local period over the range of windows being overlap-added.

This technique aligns the local periods before they are overlap-added and preserves the phase. Very few signals contain a single frequency however, making the above scheme of little use. Aligning windows with respect to the highest amplitude local period in the overlapping region is the next obvious alternative. Roucos and Wilgus proposed time-aligning windows on the basis of signal similarity before adding them. This is accomplished by maximizing the crosscorrelation of overlapping windows. The crosscorrelation aligns the windows before the overlap-add step, preserving the magnitude and phase (i.e. the local period). They dubbed this process "Synchronized Overlap-Add" or "SOLA" to distinguish it from the Griffin-Lim "Overlap-Add" which performs no shifting.

2.3 Formal Definition

The SOLA algorithm modifies the time-scale of a signal in two steps, analysis and synthesis. The analysis step consists of windowing the input signal every S_a (Shift analysis) samples. The synthesis step consists of overlap-adding the windows from the analysis step every S_s (Shift synthesis) samples. Each new window is aligned with the sum of previous windows before

being added. This reduces discontinuities arising from the different interframe intervals used during analysis and synthesis. The resulting time-scale modified signal is free of detectable harmonics, clicks, and pops.

In the "Synchronized Overlap-Add" algorithm, windows are added synchronously with the local period. The time-scale modified signal, $y(n)$, obtained using the "Synchronized Overlap-Add" of windowed segments, $x_w(n) = w(n)x(n)$ (where $x(n)$ is the input signal and $w(n)$ is the windowing function), is given by:

1. Initializing the signals $y_w(n)$ and $r(n)$.

$$\left.\begin{aligned} y_w(n) &= x_w(n) \\ r(n) &= w(n) \end{aligned}\right\} \text{for } n = 0 \ldots Winlen - 1 \tag{2.1}$$

2. Updating $y_w(n)$ and $r(n)$ by each new frame of the input signal, $x_w(n)$, as follows:

$$y_w(mS_s - k(m) + j) = \begin{cases} y_w(mS_s - k(m) + j) + x_w(mS_a + j) & \text{for } 0 \le j \le L_m - 1 \\ x_w(mS_a + j) & \text{for } L_m \le j \le Winlen - 1 \end{cases} \tag{2.2}$$

$$r(mS_s - k(m) + j) = \begin{cases} r(mS_s - k(m) + j) + w(mS_a + j) & \text{for } 0 \le j \le L_m - 1 \\ w(mS_a + j) & \text{for } L_m \le j \le Winlen - 1 \end{cases} \tag{2.3}$$

$$k(m) = \max R^m_{xy}(k) \tag{2.4}$$

$$R^m_{xy}(k) = \frac{\sum_{j=0}^{L_m-1} y_w(mS_s - k + j)x_w(mS_a + j)}{\sqrt{\left[\sum_{j=0}^{L_m-1} y_w^2(mS_s - k + j)\right]\left[\sum_{j=0}^{L_m-1} x_w^2(mS_a + j)\right]}} \tag{2.5}$$

3. Normalizing $y_w(n)$ by the buffer of appropriately shifted windowing functions $r(n)$ to obtain the final output $y(n)$:

$$y(j) = \frac{y_w(j)}{r(j)} \quad \text{for all } j$$

As outlined in the above equations, $k(m) > 0$, corresponds to a shift backward along the time-axis of the mth frame that maximizes the normalized crosscorrelation R_{xy}^m between the mth window and rate-modified shifted signal composed of windows $0 \rightarrow m-1$. L_m is the number of overlapping points between the new window $x_w(mS_a + j)$ and the existing sequence $y_w(mS_s - k(m) + j)$ for the current frame m, and *winlen* is the number of data points in each window frame $x_w(mS_a + j)$. It is important to remember in subsequent discussions that $k(m)$ positive in the expression $y_w(mS_s - k(m) + j)$ moves the region of overlap backwards along the time-axis. Thus the above equations define an implementation in which each new window is targeted to the maximally advanced (right-most) position on the shift interval and is then shifted backward to test shifts along the interval. Section 2.8 shows how this method is equivalent to other less easily represented methods.

Maximizing the crosscorrelation insures the current window is added and averaged with the most similar region of the reconstructed signal as it exists at that point. The shifting operation insures the largest amplitude periodicity of the signal will be preserved in the rate-modified signal. This signal will be called the *rate-modified shifted signal* to distinguish it from the *rate-modified unshifted signal* obtained by simply overlap-adding.

A time-scale modified signal is constructed by adding overlapping segments of speech such that similar regions in the segments are aligned and then scaling the sum appropriately. The operation does not distort the waveform provided the overlapping portions of the signal are very similar. Averaging the overlapping portions smoothes transitions between waveforms in adjacent windows. This technique preserves the local pitch period while allowing time-scale modification by changing the relative overlap between neighboring windows of the original and time-scale modified signals.

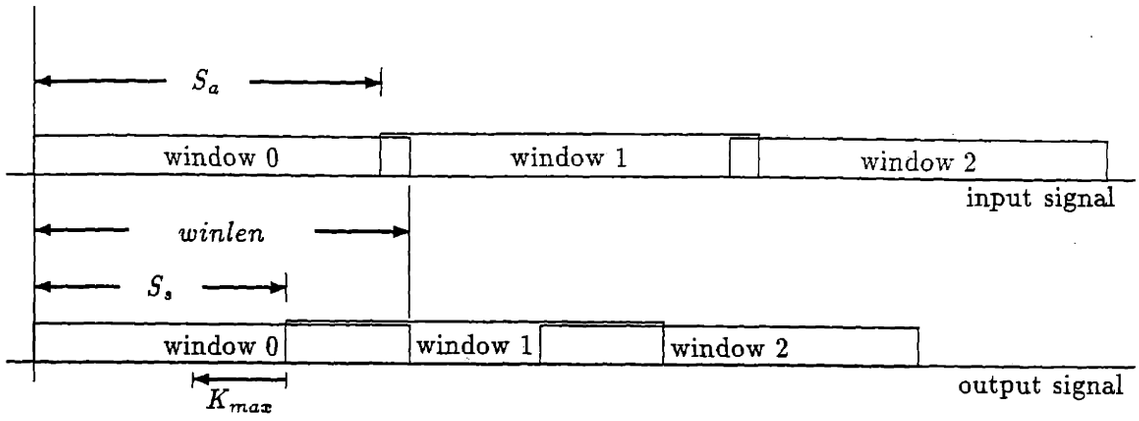


Figure 2-1: Parameters for SOLA algorithm.

2.3.1 Parameters

There are four distinct parameters for the SOLA algorithm: window length, analysis shift, synthesis shift, and shift search interval.

- Window Length ($winlen$): The duration of windowed segments of the input speech. This parameter is identical for both the input and output buffers, and represents the smallest unit of speech manipulated by the algorithm.
- Analysis Shift (S_a): The interframe interval between successive windows along the input signal.
- Synthesis Shift (S_s): The interframe interval between windows along the unshifted output signal.
- Shift Search Interval (K_{max}): The duration of the interval over which a window may be shifted for alignment with previous windows.

For convenience, a time-scale modification factor, $\alpha = \frac{S_s}{S_a}$, is defined. The approximate duration of the modified signal is given by: $\alpha *$ (Duration of input signal). For time-scale compression or speed-up, α will be less than 1; i.e., $S_s < S_a$. For time-scale expansion or slow-down, α will be greater than 1; i.e., $S_s > S_a$. For $S_s = S_a$, $\alpha = 1$ and there is no time-scale change.

2.3.2 Normalized Crosscorrelation

The normalized crosscorrelation (Equation 2.5) is used to evaluate the similarity between overlapping segments. The window to be added is shifted along the shift interval and a normalized crosscorrelation evaluated at each location. The normalized crosscorrelation function is maximal when the largest amplitude periodicities in the segments are aligned. The crosscorrelation is well suited for this application as the human ear tends to focus on the highest amplitude frequencies in the signal.

Experimental experience has shown the number of overlapping sample points, L_m, should be greater than 20. This choice of L_m insures an accurate indication of alignment for uncorrelated, semi-correlated, and highly correlated segments. Although the number of overlapping sample points can be reduced for highly correlated segments,[1] the normalized crosscorrelation must also provide an accurate measure of the similarity between less correlated segments. Note that the normalized crosscorrelation always attains its peak magnitude of ± 1 with only one overlapping point. For Gaussian white-random noise the variance in the correlation function decreases as $\frac{1}{L_m}$, and the standard deviation as $\frac{1}{\sqrt{L_m}}$. To increase our confidence in the indication of alignment provided by the correlation function, the standard deviation of the function should be much less than 1. Thus we desire:

$$\frac{1}{\sqrt{L_m}} \ll 1$$

Call Ola_{min} the minimum number of overlapping points between windows to accurately determine their correlation. Preliminary results indicate that 20 data points provide a very robust indication of the correlation for most speech segments sampled at 8 kHz.

2.3.3 Computational Requirements

The computational requirements of the SOLA technique are determined by the choice of parameter values. The input signal is windowed every S_a points. The number of windowed segments, or frames, per unit time is therefore inversely proportional to S_a. For each frame,

[1] The actual number of points depends on the similarity between the segments. As with matched filter design, the greater the similarity, the less points needed to indicate correlation.

we must determine the proper shift value such that the crosscorrelation with previous frames will be maximized. For each possible shift value on the interval $[0, K_{max}]$, a normalized crosscorrelation must be evaluated to determine which shift gives the highest correlation. The inner product calculations (IPC in equations below) in the normalized crosscorrelation require a total of 5 multiplies and 3 adds for every overlapping point. The minimum overlapping points on the shift interval (MOPOSI in equations) varies from frame to frame, but on average, MOPOSI $= winlen - S_s$. It is important to note the distinction between MOPOSI and Ola_{min} which is the minimum number of points necessary for confidence in the crosscorrelation computation. MOPOSI will always be greater than Ola_{min} for reasons detailed in Section 3.3.2.

Thus we may express the required computations as proportional to:

$$\begin{aligned}
&\propto \left[\frac{\#\text{ frames}}{\text{output sample}}\right]\left\{\left[\frac{\#\text{ shift evaluations}}{\text{frame}}\right]\left[\frac{\text{avg. }\#\text{ computations}}{\text{shifteval}}\right] + [\text{update comps.}]\right\} \\
&\propto \left[\frac{1}{\alpha S_a}\right]\left\{\left[\frac{K_{max}}{1}\right]\left[\text{MOPOSI} + \frac{K_{max}}{2}\right][\text{IPC}] + [\text{MOPOSI}]\,[\text{update computations}]\right\} \\
&\propto \left[\frac{1}{\alpha S_a}\right]\left\{\left[\text{MOPOSI} + \frac{K_{max}}{2}\right][K_{max}]\,[\text{IPC}] + [\text{MOPOSI}]\,[\text{update computations}]\right\} \quad (2.6)
\end{aligned}$$

$$\text{MOPOSI} = winlen - S_s \geq Ola_{min} + K_{max}$$

From Equation 2.6 we see that the order of the computations required is primarily a function of K_{max}, α, MOPOSI $= winlen - \alpha S_a$, and S_a. Although Equation 2.6 is only an approximation of the total number of operations required to implement the algorithm, it illustrates two important points. First, shift determination forms the bulk of the computations required for each frame. Second, S_a and α determine the number of frames per output sample.

2.4 Operation of SOLA for Time-Scale Modification

Time-scale modification is accomplished by changing the interframe shift used for construction relative to the interframe shift used for analysis. Time-scale compression uses a synthesis

shift that is smaller than the analysis shift to construct a time-scale compressed signal (see Figure 2-2). Time-scale expansion uses a synthesis shift that is larger than the analysis shift to construct a time-scale expanded signal (see Figure 2-4).

2.4.1 Time-Scale Compression

Time-scale compression is achieved by decreasing the interframe shift used to reconstruct the modified signal relative to the interframe shift used on the original signal as shown in Figure 2-2. Thus speed-up by a factor of two (halving the time-scale of the signal) is accomplished by using a synthesis shift that is half the analysis shift, $\alpha = 0.5$. The increase in the amount of overlap between frames results in a compressed time-scale, while shifting to maximize the crosscorrelation aligns the periodicities in the segments preserving the periodic structure. Increasing the overlap between segments combines multiple periods into one, reducing the number of periods in the signal while maintaining the characteristics of the waveform provided the overlapping portions are similar.

Figure 2-3 details the sequence of events in the construction of a time-scale compressed signal. The output signal is initialized with the first window, *window 0*, of the input signal. Subsequent windows are targeted to their respective locations and then shifted to maximize their crosscorrelation with the existing output buffer. *Window 1* is targeted S_s points into the output signal and is then shifted appropriately along the shift interval as shown on axis labelled *window 1 shifted* in Figure 2-3. *Window 2* is likewise targeted $2S_s$ into the output buffer and shifted appropriately (see axis labelled *window 2 shifted*, Figure 2-3). The output buffer is then scaled by identically shifted windowing functions (in this example rectangular windows are assumed so $x_w(n) = x(n)$) to give the final output.

The overlay of the windows in Figures 2-2 and 2-3 illustrates the local optimality of the algorithm. Note the shift of subsequent windows is not taken into consideration when determining the shift for the current window.

2.4.2 Time-Scale Expansion

Time-scale expansion is achieved by increasing the interframe shift used to reconstruct the signal relative to the interframe shift used on the original signal, as shown in Figure 2-4.

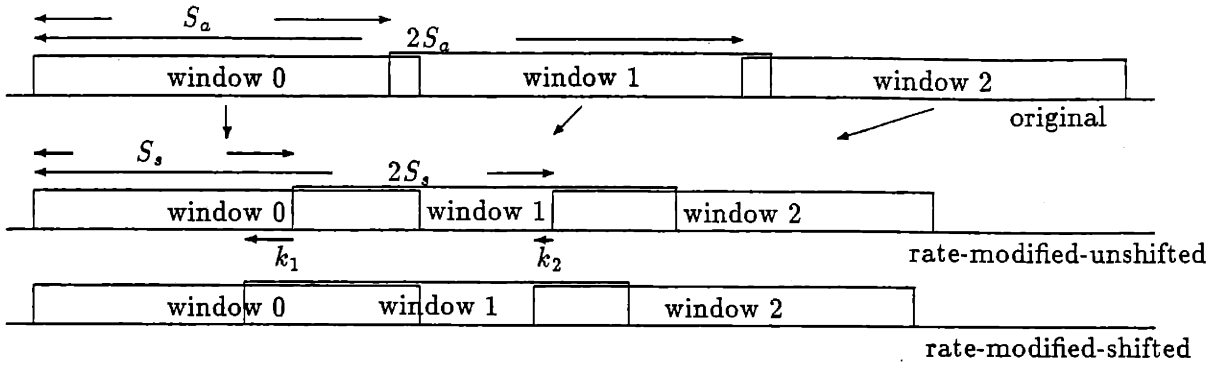


Figure 2-2: Overview of time-scale compression.

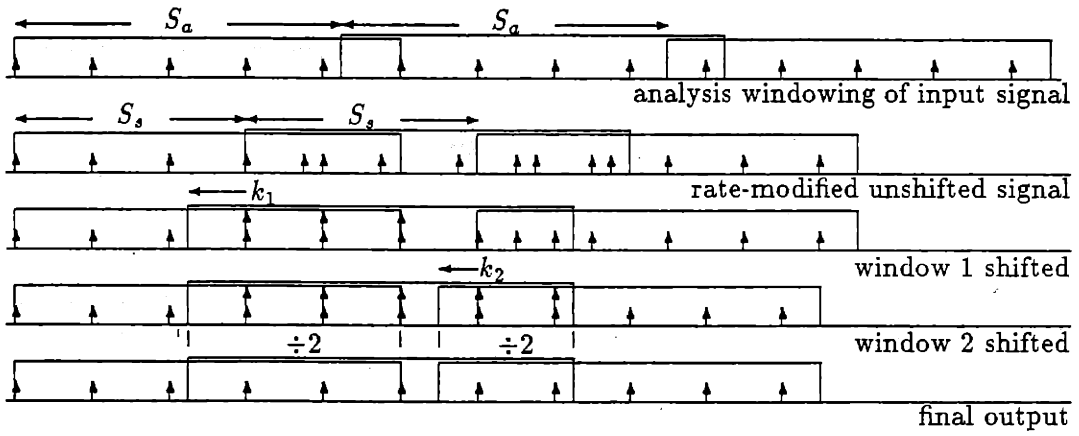


Figure 2-3: Detail of Synchronized Overlap-Add for time-scale compression.

Slow-down by a factor of two (doubling the time-scale of the signal) is accomplished by using a synthesis shift that is twice the analysis shift, $\alpha = 2$. The decrease in the amount of overlap expands the time-scale, while shifting to maximize the cross-correlation aligns the periodicities in the segments preserving the periodic structure. As the overlap is decreased, portions of the input signal are replicated in the output signal (see Figure 2-4). Thus a region of the input signal now appears twice in the output signal. Shifting insures replicated portions are aligned with periodicities in neighboring windows.

Figure 2-5 details the sequence of events in the construction of a time-scale expanded signal. The output signal is initialized with *window 0* of the input signal. *Window 1* is targeted S_s sample points into the output signal and is then shifted appropriately as shown on the axis labelled *window 1 shifted* in Figure 2-5. *Window 2* is likewise targeted $2S_s$ into the output signal and shifted appropriately (see axis labeled *window 2 shifted*, Figure 2-5). Note both *window 0* and *window 1* contain the fourth, fifth and sixth impulses of the input signal. The increase in the interframe shift interval (decrease in the amount of overlap between windows) causes the fourth and sixth impulses to appear in *window 0* and *window 1* respectively. This results in the replication of two impulses. *Window 1* is then shifted to align the fifth impulse from *window 0* with the fourth impulse from *window 1*. In the region of overlap, the fifth impulse from *window 0* is added to the fourth impulse from *window 1* and the sum is divided by two.

The shifting aligns periods in the regions of overlap reducing discontinuities in the signal. (Note the phase change in *windows 1* and *2* in Figure 2-4). The resulting final output is free of detectable harmonics, clicks, and pops since the discontinuities at window junctions have been substantially reduced.

2.5 Actual Time-Scale Modification Performed

Each window is targeted to a position corresponding to its ideal time-scaled position, $m * S_s$ where m is the window number, but is then shifted to align its local periodicities with those of the previous window (see Figures 2-2 and 2-4). The shifts do not accumulate since the target position of a window is independent of any previous shifts. Each shift, however, is dependent on the shift of the window before it. The shifting can be viewed as a forward-moving process

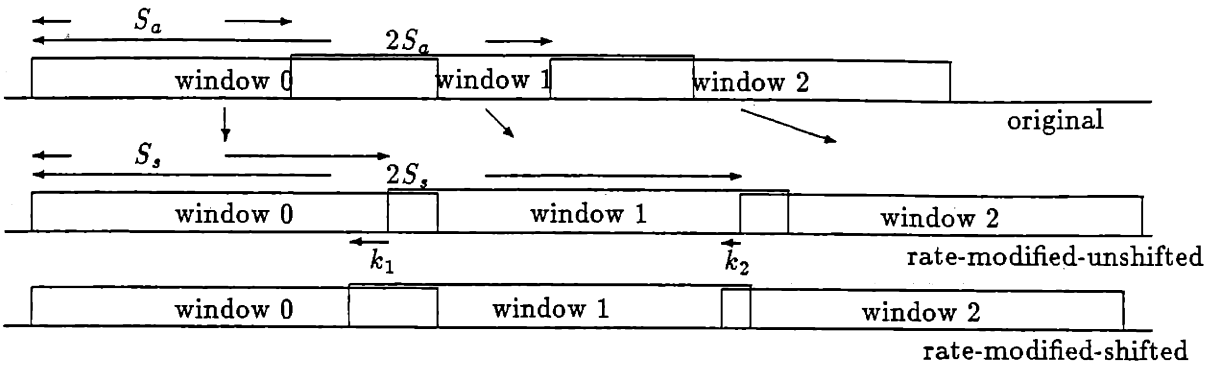


Figure 2-4: Overview of time-scale expansion.

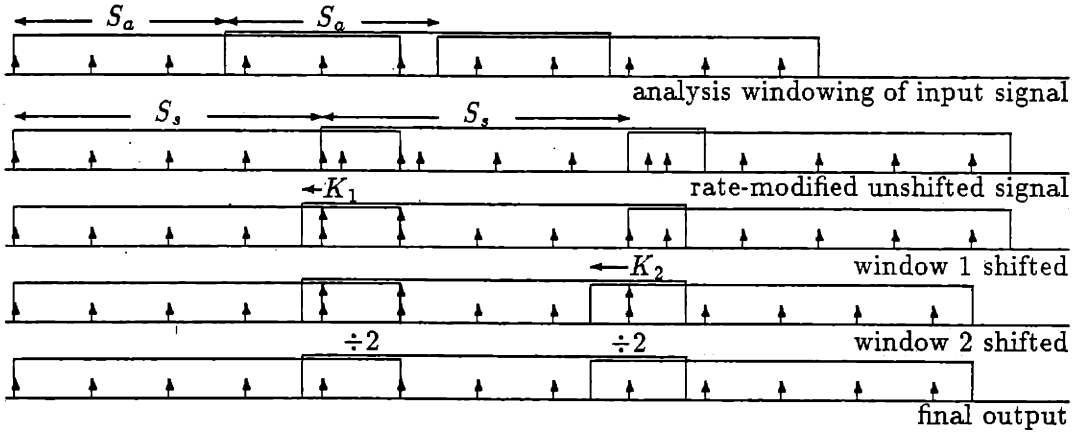


Figure 2-5: Detail of Synchronized Overlap-Add for time-scale expansion.

along the rate-modified-unshifted signal which moves the mth window, W_m an amount less than K_{max} to align it with the sum of the previous $m-1$ shifted windows, W_0 to W_{m-1}. In Figures 2-2 and 2-4, *window 1* is correlated with *window 0* and shifted appropriately. Next, *window 2* is correlated with *window 1* (which has moved) and shifted appropriately. Note, however, that the shift of the last window is the only shift which alters the duration of the output signal.

The duration of the input signal is assumed to be very long compared to the length of a window. In this case the duration of the rate-modified signal is accurately represented by the ratio of the synthesis and analysis shifts, $\alpha = \frac{S_s}{S_a}$. The actual duration of the rate-modified-shifted signal can be computed as follows:

$$\begin{aligned} d_i &= \text{duration of input signal} \\ d_o &= \text{duration of output signal} \\ K_{last} &= \text{shift of last window} \\ \text{number of frames} &= \frac{d_i - winlen}{S_a} \end{aligned}$$

$$\begin{aligned} d_o &= (\text{number of frames})(S_s) + winlen - K_{last} \\ &= (\frac{d_i - winlen}{S_a})(S_s) + winlen - K_{last} \\ &= (d_i - winlen)\alpha + winlen - K_{last} \\ d_o &= \alpha d_i + (1-\alpha)winlen - K_{last} \qquad (2.7) \\ d_o &\approx \alpha d_i \text{ for } d_i \gg winlen \text{ and } d_i \gg K_{last} \qquad (2.8) \end{aligned}$$

2.6 Variable Rate Time-Scale Expansion

In most applications uniform time-scale modification is desired. In these applications α is constant over the entire signal. Variable rate time-scale modification is easily achieved by simply varying the value of α along the signal. This can be accomplished in several ways: the analysis shift along the input can be held constant while varying the synthesis shift, the

synthesis shift can be held constant while varying the analysis shift along the input, or both the analysis shift and synthesis shifts can be varied.

The ability to vary either or both of the interframe shift parameters is desirable and the proper choice depends on the application. If specific portions of the input signal are to be modified with different time-scale factors, changing the analysis shift appropriately along these portions is all that is necessary. Alternatively, if the time-scale modification desired is output signal dependent, the synthesis shift should be changed appropriately. The interframe shift parameters are subject to the same constraints that apply for constant rate time-scale modification, thus varying both S_a and S_s may be required when substantial changes in the time-scale modification factor are desired.

2.7 Global versus Local Maximization of the Crosscorrelation

The overlays of windows in Figures 2-3 and 2-5 expose the local optimization performed by the SOLA algorithm. The shift of a current window is computed without regard for its effect on subsequent window alignment. Reducing the crosscorrelation of a current window, $\max R_{xy}^{m}(k)$, may increase the crosscorrelation of several subsequent windows, $\max R_{xy}^{m+1}(k)$, $\max R_{xy}^{m+2}(k)$, $\max R_{xy}^{m+3}(k)$, increasing the global crosscorrelation. Maximizing the global crosscorrelations would require non-causal processing of the signal, thus eliminating the possibility of performing real-time time-scale modification.

2.8 Direction of Shifts

This section will show that the processes of delaying, advancing, or any combination of the two are sufficient for aligning the windows.

A window can be delayed or advanced in time from its target position to align it with previous windows. Either method will provide the desired result of pitch period alignment. The normalized crosscorrelation, however, can give unreliable results when the number of overlapping points is small (less than Ola_{min}). Regardless of the method employed, the forward-most position on the shift interval must be such that at least Ola_{min} data points

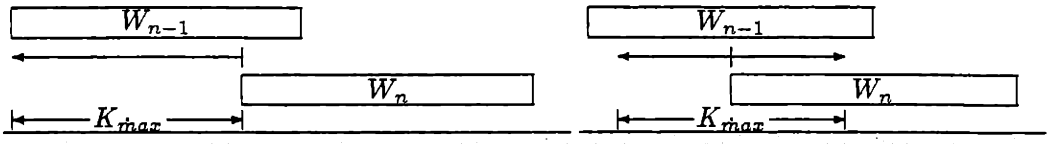


Figure 2-6: Different methods of searching the shift interval.

overlap (see Section 2.3.2). This restriction is further detailed in Section 3.3.2.

Note in Figure 2-6 that identical alignment in the rate-modified shifted signal is obtained by targeting windows to different positions and searching the same shift interval. Each results in a movement of the window such that the periodicities are aligned.

The process of shifting backward along the time axis from a given target point, Ta, is identical to shifting in both directions by $\frac{K_{max}}{2}$ with a target point, $Tb = Ta - \frac{K_{max}}{2}$, on the time axis. Recall that the ratio $\frac{S_s}{S_a}$ determines the rate change, and the actual value of S_s varies for each window. It is important to note that during steady-state operation, the duration of the shift interval is the same in both cases and the distance between shift intervals is S_s in both cases. Thus the two methods are equivalent.

An implementation in which the target positions of the windows corresponded to the forward-most position on the shift interval (i.e., all shifts backward) was compared with an implementation in which the target positions of the windows corresponded to the backward-most position on the shift interval (i.e., all shifts forward). There was no detectable difference between speech produced by the two different methods over the range of compression and expansion, $0.5 \leq \alpha \leq 2.0$.

2.9 Summary

In this section we have introduced the SOLA algorithm originally suggested by Roucos and Wilgus as an initial guess for the Griffin-Lim algorithm. The parameter set was defined and

operation of the algorithm for time-scale expansion and compression was outlined in detail. An indication of the order of computations in terms of the parameters was provided to indicate the relative importance of each parameter's effect on the computations required.

The SOLA procedure for modifying the time-scale of speech offers many advantages over previous time-domain techniques. Pitch extraction is not required; and explicit splicing operations are not performed. The algorithm operates in the time domain and attempts to align periodicities in overlapping segments of speech by maximizing the crosscorrelation between the two overlapping segments.

In many applications it is desirable to perform all window shifting in a single direction. This can greatly simplify implementation. It has been shown that shifting in either direction is adequate for aligning the pitch periods.

Chapter 3

Parameter Set Interactions

3.1 Introduction

In Chapter 2 the parameters for the SOLA algorithm were defined. This chapter will detail the interactions of the parameters and develop the concept of a parameter set. Current literature on the SOLA algorithm [11] [14] [3] [13] provides only ad hoc information about the correct choice of parameters. Investigations into the operation of the algorithm have shown that output signal quality is entirely predictable from the parameters chosen. Equations are presented which explain many of the deleterious effects observed for certain parameter sets. This chapter presents the restrictions on the parameter set imposed for efficient implementation, and restrictions necessary for high quality output. A "solution space" for the parameter set is also given.

The proper choice of parameters depends on the nature of the input signal and the time-scale modification desired. Successful operation of the SOLA algorithm is heavily tied to the nature of the speech signal being processed. This chapter introduces a simplified model for the synthesis of speech to better understand the speech signal.

3.2 The Speech Signal

At this point it is important to present several underlying assumptions being made about the nature of the speech signal. A popular model for human speech consists of an excitation

source and a time varying filter (the vocal tract). The excitation source may be glottal pulses in the case of voiced speech such as vowels, or random noise excitation in the case of frication. The speech signal can therefore be modeled as the output of this time varying filter excited by one or both of the excitation sources. Such a model accurately represents almost all of the waveforms encountered in human speech.

3.2.1 Voiced Speech

The vocal tract transfer function is controlled by the position of the tongue, teeth, and lips. As the position of these does not change instantaneously, the transfer function is for the most part slowly varying. The period of the glottal pulses from the vocal chords varies among speakers across the range 2.5 msec to 12.5 msec. During steady-state portions of the speech signal the vocal tract is excited by several glottal pulses before significant changes occur in the transfer function. This results in a signal which contains several very similar periods and is said to be stationary over short intervals.

3.2.2 Fricated Speech

In portions of speech where the vocal tract is excited by random noise, phonemes such as "S" and "th", the speech signal resembles the random noise excitation source. The waveform in such regions is uncorrelated and contains numerous discontinuities with no periodic structure. The lack of any periodic structure means that added discontinuities in the output signal in these regions are relatively undetectable.

3.2.3 Abrupt Changes in the Speech Signal

Abrupt changes represent a special problem for dividing the input signal into units of speech. Sudden onsets of voiced or fricated phonemes occur at word boundaries and around plosives. Similarly, rapid attenuation or changes in volume violate the stationarity assumption over the window duration.

Ideally no window should contain such transitory portions of the speech signal. Dividing the dissimilar portions of the signal on these boundaries would help maintain the similarity assumption throughout the window. Such a scheme is not easily implemented, however, nor

is it proposed. Rather, the window length is chosen short enough such that violations of the stationarity assumption are limited to regions where the input changes abruptly. The window length must remain fixed throughout the algorithm, thus it is impossible to eliminate transitory portions from all windows. In cases where abrupt transitions are separated by less than a window length, it is impossible to eliminate all of them from a window.

3.2.4 Effect of Different Articulation Rates

The goal of any time-scale modification algorithm is to scale the articulation rate while maintaining natural sounding speech. When humans speak rapidly, the speech contains unequally shortened phonemes. The duration of stops and plosives remains constant, while voiced phonemes and nasals are abbreviated and run together. Silent portions of speech are usually eliminated and words tend to run together. Slowly articulated speech contains extended durations of silence and expanded voiced phonemes.

The SOLA algorithm modifies the time-scale evenly over the entire signal according to the shifts that lead to maximum correlation in the overlapping portions. The time-scale modified speech produced by the algorithm is therefore not identical to that of a human speaker. Nonetheless, the speech remains clearly intelligible at articulation rates exceeding those capable of most humans.

3.3 Parameter Set Restrictions

Each parameter has specific effects on the processing that occurs during the Synchronized Overlap-Add procedure. As previously outlined, blocks of speech (windows) are taken every S_a points along the input signal, targeted to positions every S_s points along the output signal, then shifted to maximize the correlation of the overlapping portions. The goal of this process is the addition or deletion of integer multiples of pitch periods in the output signal. Each parameter contributes to the success or failure of this process. The interaction of the parameters is crucial for obtaining the desired result. A poorly chosen parameter set can lead to needless computations or undesirable results in the output signal such as reverberation or pitch fracturing (see Section 3.3.1). The crucial operation in the algorithm is the alignment

of the windows. Alignment is most important during periodic portions of the signal where errors in alignment are easily detected. To obtain the desired result, the overlapping portions of the windows must be highly correlated. Therefore each window must be similar in periodic structure over the region of overlap with previous windows.

The correct choices for the window length and shift search interval depend solely on the input signal. The correct choices for S_a and S_s parameters however depend on the time-scale modification and quality desired as well as the computational resources available.

3.3.1 Window Length and Shift Search Interval: Input Signal Dependent, α Independent Parameters

Window Length

The SOLA algorithm operates by manipulating windowed segments of speech to modify the time-scale of signals. The window length parameter sets the size of the smallest manipulable unit of speech and therefore has the same duration in both the input and rate-modified output signals. The windows are treated as units of speech, and are assumed to be relatively stationary in character. The window length must be chosen carefully to avoid violating this assumption, namely: short enough that the speech signal characteristics do not change drastically over the duration of the window, and long enough to contain two pitch periods. The correct choice for this parameter is entirely dependent on the input signal's characteristics.

Each window must contain at least two pitch periods, one for alignment and one to advance the output signal. To meet this criteria for male speech the window must be longer than 25 msec (200 samples). Window lengths greater than 40 msec (320 samples) lead to slight reverberation in the output signal and when window lengths approach 50 msec (400 samples) there is noticeable reverberation in the output signal. The reverberation is most pronounced for female speech due to the shorter pitch periods. The source of reverberation is detailed in Section 3.4.

For slow-down the length of the window is further constrained by the synthesis shift, S_s between windows. The duration of each window must be long enough that there are no gaps in the output signal.

The interaction of the window length parameter with the other parameters affects com-

putation and quality in the output signal. These effects will be developed later. Since the correct choice for this parameter is entirely dependent on the input signal, it will have the same value for both expansion and compression.

A formal listen test was performed to determine the preferred window lengths for male and female speech. The results are summarized in Appendix A. These results indicate that different window durations are preferable for male and female speech. Window lengths between 25msec (200 samples) and 37.5 msec (300 samples) are long enough to contain two pitch periods in male speech and yet short enough that reverberation is not noticeable for female speech. Female speech processed using the longer windows required for male speech was not detected as significantly lower in quality (see results of listen test in Figure A-4). Thus the proper choice of window length will provide high quality output for both male and female speech.

A value of 256 sample points was chosen as it fell within the range of 200 and 300 where few listeners indicated a preference (see listen test results in Figure A-4), and provides a block of data conveniently manipulated by most processors.

Shift Search Interval

The duration of the shift search interval, K_{max}, (see Section 2.3.1) limits the distance a window may be shifted along the time-axis for pitch period alignment across window boundaries. To accomplish this, K_{max} must be greater or equal to the largest pitch period encountered. Note the pitch phase in the current window may be one sample ahead of the pitch phase in the previous window at the forward-most position of the shift interval. For this situation the current window must be shifted backward the distance of a full pitch period minus one sample because a forward shift of one sample is not possible. In this scenario the pitch period is aligned with the previous pitch period rather than the pitch period at the forward-most position of the shift search interval provided the correlation with the previous pitch period is higher.

If K_{max} is chosen to be smaller than the largest pitch period encountered, it will not be possible to align the periods in all cases. A discontinuity will result at the window boundary distorting the local period in the signal. This phenomena is referred to as "pitch fracturing" because the local period is fractured at the discontinuity. Figure 3-1 illustrates the effect of

pitch fracturing. Recall the shift, $k(m)$, required to align the period in a current window represents a uniformly distributed random variable on the interval $[0, K_{max}]$, thus proper alignment will be obtained a fraction of the time. The probability of successful alignment is given by:

$$\text{Probability of alignment} = \begin{cases} \frac{K_{max}}{\text{pitch period}} & \text{for } 0 \leq K_{max} \leq \text{pitch period} \\ 1 & \text{for } K_{max} \geq \text{pitch period} \end{cases} \tag{3.1}$$

Note for $|K_{max}|$ greater than the largest pitch period, successful alignment is always possible. The preferred value for K_{max} depends solely on the input signal pitch periods and is different for male and females. Ideally K_{max} should equal the duration of a pitch period. In practice an implementation suited for very long pitch periods, 12.5 msec (100 samples), is inefficient for pitch periods of shorter duration, but does not adversely affect the quality of the output signal for speech with shorter pitch periods.

In Section 3.4.3 it will be shown that the value K_{max} also determines the maximum length of a replicated feature in the output signal. See Section 3.4.3 for a detailed explanation. For this reason K_{max} should not exceed 25.0 msec (200 samples) as the time-scale of the output signal jitters and results in a jerky sound.

3.3.2 Analysis Shift: An α Dependent Parameter

The analysis shift has a very large effect on the quality of the output signal as well as the computational requirements (see Equations 3.2, 3.3 and 2.6). The analysis shift sets the number of frames per unit time. The lower the shift the more frames. Consequently it is desirable to set the shift as large as possible. The analysis shift, synthesis shift and window length interact to determine several important attributes of the resulting signal.

Effect on Averaging and Multiple Window Appearances

In Section 2.3.1 a time-scale modification factor, α was introduced, $S_s = \alpha S_a$. For a given α, S_a determines S_s and vice-versa. The interframe shifts used during analysis, S_a, and synthesis, S_s, interact with the window length, *winlen*, to determine several important attributes of the

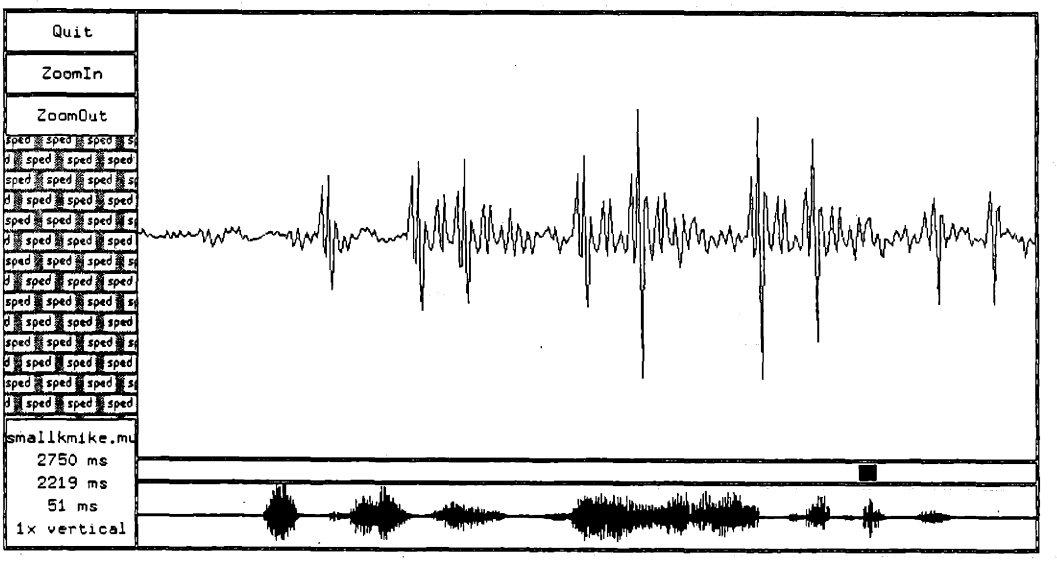


Figure 3-1: Pitch fracturing which occurs when the shift interval (K_{max}) is less than a pitch period's duration. Note the distance between the second and third pitch pulses is significantly less than the distance between the first and second pitch pulses. This "fracturing" of the local period occurs when K_{max} is less than the duration of a pitch period, which prevents proper alignment of the pitch periods in the two segments.

construction process.

The average number of windows containing any given input sample is given by:

$$\text{AWO} = \frac{winlen}{S_a} \tag{3.2}$$

This is illustrated in Figures 3-4 and 3-5 and its importance is detailed in Section 3.4.7 .

Each sample in the final output signal is comprised of one window or the sum of several overlapping windows scaled appropriately. The average number of overlapping windows which give rise to a single sample in the output signal is given by:

$$\text{SWO} = \frac{winlen}{S_s} = \frac{winlen}{\alpha S_s} \tag{3.3}$$

This can be seen in Figure 3-6 and its importance is detailed in Section 3.4.7 .

Note in Figure 3-5 the sample marked x was overlapped by 4 windows during analysis, but samples in the output signal shown in Figure 3-6 are comprised of the sum of only 3 windows since $\alpha > 1$.

Restriction Imposed by Correlation Overlap

The values for S_s and S_a must be greater than or equal to one sample. Additionally, the synthesis interframe interval, S_s, must be small enough that each window overlaps the previous window in the *rate-modified-shifted signal* (see Figures 2-2, 2-3, 2-4, 2-5, and 3-2).

The distance between the starting points of adjacent windows is given by the synthesis shift, S_s, and the difference between each window's shift, $|k(m-1) - k(m)| \leq K_{max}$. The maximum distance between the starting points of adjacent windows is then $S_s + K_{max}$.

The following discussion assumes an implementation in which each window is targeted to the right-most position of the shift interval, then shifted backward (left) along the shift interval to evaluate the crosscorrelation at each position.

Note that the right-most point (i.e. $k(m) = 0$ in Equations 2.2 and 2.3) on the shift interval designated, $Target_m$, for the mth window, W_m, must be Ola_{min} points less than the end of the previous window, W_{m-1}, which has been shifted. This insures that Ola_{min} points are available for all crosscorrelation calculations of the already shifted W_{m-1} and W_m. All

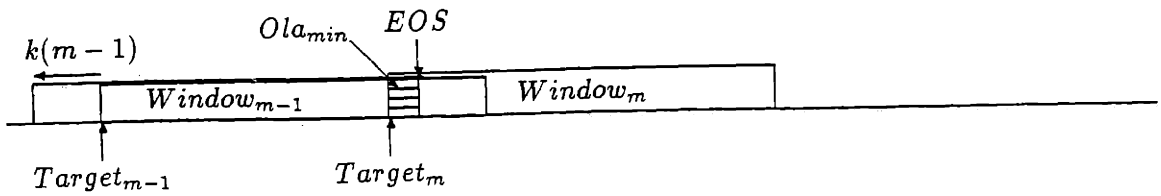


Figure 3-2: Requirements for Ola_{min} overlapping points between windows.

other positions on the shift interval will contain increased regions of overlap. The end of the previous window, W_{m-1}, after shifting is given by $Target_{m-1} + winlen - k(m-1)$.

For the mth window, W_m:

- Define: $Target_m = m * S_s$
- Initial constraint:

$$\text{End of output signal} - Ola_{min} \geq Target_m \tag{3.4}$$

- where:

$$\text{End of output signal} = Target_{m-1} + winlen - k(m-1)$$

Adhering to the aforementioned constraints, restricts the parameter set as follows:

$$\begin{aligned} Target_{m-1} + winlen - k(m-1) - Ola_{min} &\geq Target_m \\ winlen - k(m-1) - Ola_{min} &\geq Target_m - Target_{m-1} \end{aligned}$$

Noting: $Target_m - Target_{m-1} = S_s$; $k(m-1) \leq K_{max}$; and $\alpha = \frac{S_s}{S_a}$ gives:

$$\begin{aligned} winlen - K_{max} - Ola_{min} &\geq S_s = \alpha S_a \\ \frac{winlen - K_{max} - Ola_{min}}{\alpha} &\geq S_a \end{aligned} \tag{3.5}$$

The upper-bound imposed on S_a by Equation 3.5 fixes the analysis resolution required by the algorithm in terms of α, $winlen$, Ola_{min} and K_{max}. Consequently the number of frames and shift determinations per unit time is fixed.

Assuming fixed values for K_{max} and Ola_{min}, for a given time-scale modification factor

α, the window length determines the resolution required. It is important to note from the previously stated constraints (Section 2.3.1) that window lengths longer than 40.0 msec (320 samples) lead to violations of the stationarity assumption in most speech and detectable reverberation in the time-scale modified signal.

Recall that this restriction was imposed to insure Ola_{min} overlapping data points for all possible shifts on the shift interval. Violating this restriction did not produce poor quality output in many cases. Violating Equation 3.4 just means we can no longer search over the range of K_{max} shifts, but instead can only consider shifts in the range:

$$[\max\{0,\ S_s - winlen + k(m-1) + Ola_{min}\},\ K_{max}] \tag{3.6}$$

The interval of shifts containing Ola_{min} samples in the crosscorrelation has been reduced. The reduction in the duration of the shift interval decreases the chances of successful alignment as given in Equation 3.1. Note that K_{max}, which represents the duration of the shift interval in Equation 3.1, is replaced by the duration of the reduced shift interval and the duration is dependent on the previous shift.

$$\begin{aligned} \text{Duration of reduced shift interval} &= K_{max} - (S_s - winlen + k(m-1) + Ola_{min}) \\ &= K_{max} - S_s + winlen - k(m-1) - Ola_{min} \end{aligned} \tag{3.7}$$

When less than Ola_{min} points overlap, the crosscorrelation is zero by definition. If the correlation is positive for shifts in which Ola_{min} points overlap, the corresponding shift will be used. If the correlation is negative for all shifts containing Ola_{min} or greater overlapping points, the algorithm should use the shift which abuts the new window to the output signal to avoid gaps between windows.

Violations of Equation 3.4 increase the likelihood of pitch fracturing in the output signal since the shift interval duration is reduced (Equation 3.7). Thus adhering to Equation 3.4 is more important for male speech with longer periods. Experiments using parameter sets which violate Equation 3.4 provided high quality time-scale modified speech in cases where $\alpha \approx 1$. No degradation in quality was detected for female or male speech when Equation 3.4

was violated with $\alpha \approx 1$.

Analysis Shift for Time-Scale Compression, $\alpha < 1$

The goal in time-scale compression is to correlate the current window with the latter portion of the previous window. Aligning and summing the overlapping portions and scaling appropriately effectively removes an integer multiple of pitch periods. The analysis shift must not be too large or drastic speech signal changes over the interval will cause difficulty in aligning adjacent windows. If the analysis shift is too small however, excessive overlapping between frames will tend to remove the distinctions in the pitch periods and attenuate higher frequencies. This phenomena becomes more pronounced at high compression rates ($\alpha < 0.5$) where S_s is generally less than half of S_a.

The undesirable side effects of high averaging in the output signal for compression result from repeated additions of slightly differing waveforms. The sampling rate is seldom an integer multiple of the pitch period. This leads to phase differences between frequencies in the sampled signal. When multiple samples are averaged to form the output signal, lower amplitude high frequencies can destructively interfere. These effects are less pronounced for the lower frequencies as they are less subject to inter-sample phase changes.

Recall from Equation 3.2 that the average number of overlapping windows giving rise to a sample in the output signal (Synthesis Window Overlap, SWO) is given by:

$$\frac{winlen}{S_s} = \frac{winlen}{\alpha S_a} = \text{SWO}$$

While the number of windows containing any given input sample (Analysis Window Overlap, AWO) is given by:

$$\frac{winlen}{S_a} = \text{AWO}$$

It is desirable to reduce the amount of averaging (SWO) during compression as much as possible. Note that SWO increases faster than AWO for decreasing S_a when $\alpha < 1$. Therefore S_a should be chosen to be as large as possible. Reverberation in the time-scale compressed signal is eliminated by choosing a value for S_a greater or equal to the window length. This gives AWO ≤ 1, insuring that no portion of the input appears in more than one window.

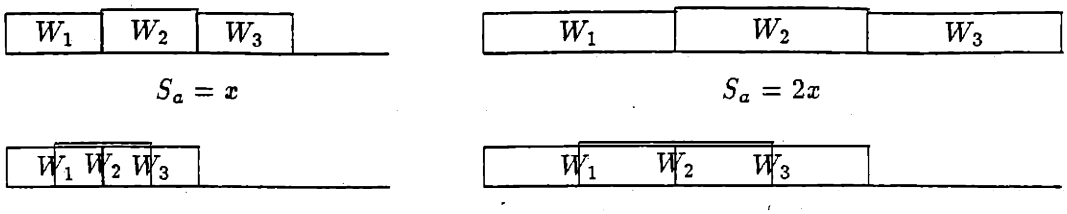


Figure 3-3: Effects of increasing analysis shift during compression.

The proper value for S_a is the largest value such that the windows remain correlated. Too long a shift will violate the highly correlated assumption of stationarity and periodic similarity between consecutive windows. As S_a increases, the similarity between neighboring and next-neighboring windows quickly diminishes. The resulting speech tends to sound choppy due to the greater changes in the signal that occur over the interval $2S_a$. Note in the Figure 3-3 that next-neighboring windows are abutted while the intervening window is overlapped and averaged across the junction.

The proposed $\frac{S_a}{\alpha} = S_a = winlen$ combination for compression by a factor of two ($\alpha = 0.5$) in this case illustrates that next-neighboring windows are abutted in the rate-modified unshifted signal. Although each window is shifted in the final output, the speech signal must remain similar over the interval of the intervening window for proper alignment and good quality. The duration of the overlapping speech segment across the junction is also critical. The importance of the stationarity requirement over a window length is apparent in this example.

The analysis shift, S_a, may actually exceed a window length. In this case, $S_a - winlen$ data points are discarded and do not appear in the output signal. The increase in S_a reduces averaging at high ($\alpha < 0.67$) compression rates (Equation 3.3). The duration of the signal which may be deleted, however, is input signal dependent. Rapidly articulated and highly fricated speech contains short duration key features, the loss of which gives rise to a short chopping-sounding output signal. Care must be taken to insure that no essential information in the input signal is skipped.

It was observed that skipping up to 6.25 msec (50 samples) did not reduce the intelligibility of the output signal. For this case S_a exceeded *winlen* by 6.25 msec (50 samples). Skipping segments greater than 8 msec (64 samples) led to loss of intelligibility and clarity in the

time-scale compressed output.

Conclusion The amount of data that can be skipped will vary with the rate of articulation in the original speech and therefore it is best to set S_a equal to a window length for robust time-scale compression. This insures that no crucial portions of the speech signal will be discarded, while simultaneously preventing reverberation.

Analysis Shift for Time-Scale Expansion, $\alpha > 1$

During time-scale expansion, the decrease in overlap between windows causes portions of the input signal to be replicated, expanding the time-scale (Figures 2-4 and 2-5). Reverberation can not be eliminated in time-scale expanded speech as portions of the input signal are required to appear in multiple windows. Note that for expansion, the analysis window overlap, AWO, will be greater than the synthesis window overlap, SWO. Equations 3.2 and 3.5 clearly support this. Equation 3.2 requires S_a greater than or equal to *winlen* to insure no segment of the input signal appears in multiple windows. To meet the requirement of Equation 3.5, however, requires S_a be less than $\frac{winlen}{\alpha}$ where $\alpha > 1$.

During the shifting process, portions of a current window are correlated with regions of the input signal $(S_s - S_a)$ points further along the time-axis. The distance $(S_s - S_a = S_a(\alpha - 1))$ must be small enough that the regions remain similar in periodic structure. Too large an analysis shift, S_a, forces windows to overlap in regions of low correlation. The quantity $S_s - S_a$ also plays an important role in the duration of the input signal replicated. If excessive, the resulting signal sounds stutterant due to discontinuities caused by the long duration of replicated portions. This effect will be further detailed in Section 3.4.3.

Conclusion: The amount of averaging during synthesis is reduced during expansion. The synthesis shift for expansion must be substantially less than a window length. Consequently, the analysis shift must be even less which increases computations. Large analysis shifts lead to undesirable artifacts in the time-scale expanded signal such as pre-echo and reverberation. The source of reverberation and how it may be reduced are the topics of the next section.

3.4 Source of Reverberation in the Output Signal

3.4.1 Introduction

Reverberation has been noted by several authors [3] [14] as an undesirable side effect of expansion via the SOLA algorithm. The source of this reverberation is identifiable in terms of the parameter set. In this section a detailed description of the source of reverberation in the time-scale modified signal is provided. Using this information, accurate predictions for the quality of the output can be made. The detailed investigation into the mechanism of the SOLA algorithm for slow-down provides insight into methods for reducing the required number of shift computations.

Informal experiments suggest that reverberation arises from repeated appearances of a single "feature" over a small region of the output signal. Such a "feature" must differ from adjacent portions of the signal to be detected as unique. Thus the "feature" described must be longer than a pitch period, and represent a nonstationarity in the signal. Events such as a sudden change in amplitude between adjacent periods or changes in the pitch period waveform represent "features" which occur often in the voice signal during stop consonants and word boundaries. For the critical voiced portions of the signal the occurrence of "features" is largely due to abrupt changes in amplitude. Figure 3-12 illustrates the problem of replicating segments of an input signal whose amplitude varies over a window length: amplitude discontinuities arise in the time-scale expanded signal as earlier portions of the input signal are joined with latter portions of the input signal contained in previous windows. This effect is detail further in Section 3.4.7.

3.4.2 Reverberation during Time-Scale Compression

During time-scale compression, portions of the input signal will appear in multiple windows if the analysis shift is less than a window length. The distance between multiple appearances of the same input signal segment can be large enough in the rate-modified-unshifted signal that overlapping the multiple appearances is not possible. This occurs when the shift to realign the regions contained in multiple windows lies outside the range of possible shift values (i.e., $> K_{max}$) along the shift search interval. In this case, segments of the input signal appear

multiply over a short region of the output signal as a pre and post echo which is perceived as reverberant. This effect is easily eliminated by forcing S_a to be greater than or equal to a window length for speed-up. In all tested cases of time-scale compression this technique provided excellent quality rate-modified speech.

3.4.3 Reverberation During Time-Scale Expansion

During time-scale expansion, portions of the input signal are purposely replicated in the output signal. For time-scale expansion S_a must always be less than a window length to avoid gaps in the output signal and erroneous correlation indications. (see Equation 3.5). Analysis window overlap (AWO>1) is necessary for repetition of segments or pitch periods in the output signal (Figures 3-4 and 3-5). In the rate-modified-unshifted signal, short segments of the input signal are replicated at every window juncture (the segments between the x's in Figure 3-6). The duration of these segments is given by $(S_s - S_a)$ (see rate-modified-unshifted signal in Figure 3-7). The shifting process applies a "borrow from Betty and give to Sue" like procedure to align periodicities in the windows and consolidate the small replicated portions into larger sections which results in the replication of input signal segments. Figure 3-8 illustrates the effects of the shifting process which occur on the intermediate rate-modified-unshifted signal in Figure 3-7. The larger replicated portions do not have to be an integer multiple of the pitch period but will vary in duration between $(S_s - S_a)$ and $K_{max} + (S_s - S_a)$ samples. The rate-modified-shifted signal then contains windows whose phase alignment is continuous over correlated portions of the signal.

Figures 3-4 through 3-8 assume a single feature, x, in the input signal. Note that the number of windows which contain the feature, x, is given by AWO, or $\frac{winlen}{S_a}$, as shown in Figure 3-4. Also note the feature shifts toward the beginning of each window by S_a in successive windows as shown in Figure 3-5.

Each window, W_m, is targeted to the position $n * S_s$ in the rate-modified-unshifted output signal as shown in Figure 3-6.

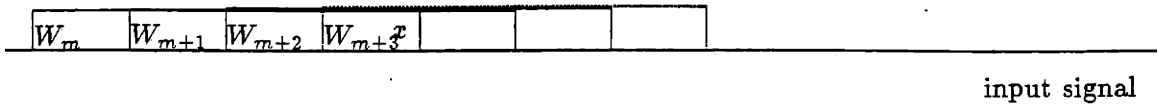


Figure 3-4: Analysis windowing for time-scale expansion.

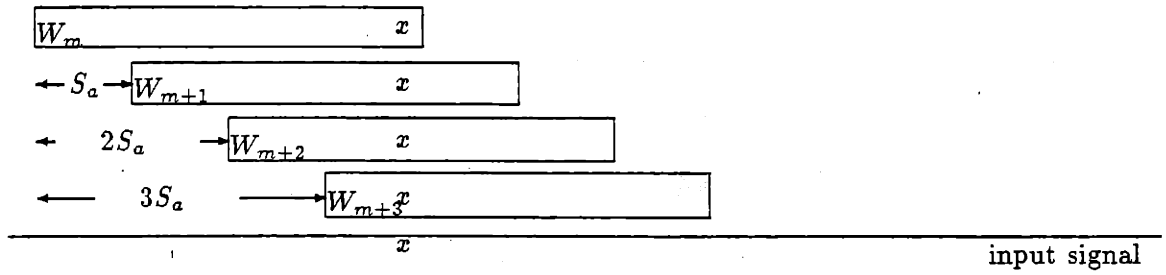


Figure 3-5: Expanded view of analysis windowing of input signal.

The distance between appearances of the feature x in the rate-modified-unshifted signal is therefore given by $S_s - S_a$, and the maximum distance between the first occurrence of x and the last occurrence x is given by $((\frac{winlen}{S_a} = \text{number of windows containing } x) - 1)(S_s - S_a)$ as illustrated in Figures 3-7 and 3-8.

The process of shifting to maximize the crosscorrelation between adjacent windows can be viewed as a forward-moving process (see Section 2.5) which moves from left to right along the rate-modified-unshifted signal to create the rate-modified-shifted signal. The normalized crosscorrelation function will be maximum (i.e. equal to 1) when the overlapping portions of the signals $x(mS_a + j)$ and $y(mS_s - k(m) + j)$ are identical. Thus the algorithm will return a shift value, $k(m)$, that overlaps the feature x in the current window, W_m, with the x appearing in the previous window, W_{m-1}. The shift to accomplish this is: $k(m) = S_s - S_a$, and undoes the expansion. Thus the right hand side of Equation 2.2 becomes:

$$\begin{aligned} y_w[mS_s - k(m) + j] + x_w[mS_a + j] \quad \text{with} \quad & k(m) = S_s - S_a \\ = \quad & y_w[mS_s - (S_s - S_a) + j] + x_w[mS_a + j] \\ = \quad & y_w[(m-1)S_s + S_a + j] + x_w[(m-1)S_a + S_a + j] \end{aligned}$$

At the next frame, the new window is advanced S_a along the input signal, $m = m + 1$ and $k(m+1) = 2(S_s - S_a)$ provided $2(S_s - S_a) \leq K_{max}$. Thus the output signal is given by:

$$\begin{aligned} & y_w[(m+1)S_s + S_a - k(m+1) + j] \text{ with } k(m+1) = 2(S_s - S_a) \\ = \quad & y_w[(m+1)S_s - 2(S_s - S_a) + j] + x_w[(m+1)S_a + j] \\ = \quad & y_w[(m-1)S_s + 2S_a + j] + x_w[(m-1)S_a + 2S_a + j] \end{aligned}$$

Note that for frames m and $m+1$, the output signal is constructed by simply adding the two new windows, W_m and W_{m+1}, to the existing signal, $y_w[(m-1)S_s - k(m-1) + j]$, with the analysis shift, S_a. The windows are overlap-added with the identical interframe shift used during analysis. This shifting process is more appropriately dubbed an "overlay" process since the output signal is reconstructed using the analysis shift, and identical segments of input "overlay" one another in the final output signal.

The shifting operation will overlay multiple occurrences of the feature x on top of one another until the shift required to align x in a given window with the overlay "pile" of x's exceeds K_{max} (see Figure 3-8). At this point the shift returned will be the best possible alignment of the new window with the end of the previous series of shifted windows. The feature x in this window now becomes a "seed" for the next "overlay pile" and all subsequent windows containing x will be shifted onto this "pile" until the required shift exceeds K_{max}—in which case the process repeats—or the feature no longer appears in subsequent windows.

Duration of input signal replicated

It is interesting to note that since the overlay process merely replicates portions of the input signal until the shift to do so exceed K_{max}, it may appear that using a larger value of $(S_s - S_a) = S_a(\alpha - 1)$ would increase efficiency. The quality of the signal suffers however when $S_a(\alpha - 1)$ exceeds K_{max}. This is apparently due to the long length of input signal replicated.

The values of $S_a(\alpha - 1)$ and K_{max} fix the distance between piles of features in the output signal. Thus $S_a(\alpha - 1)$ determines the minimum portion of input signal replicated and $S_a(\alpha - 1) + K_{max}$ determines the maximum size of a replicated feature. To create a new pile, the distance from an occurring feature to an existing pile must be greater than K_{max}. As the shift to align this feature is outside the range of possible shift values, a shift inside the range of shift values with the highest correlation is used. This shift must be between 0 and K_{max}. Assuming the previous window, W_{m-1} was shifted exactly K_{max} to align the feature it contained with the previous pile, leads to the following results: A shift of $k(m) = 0$ for the current window will give the largest distance between the previous pile and new seed; this distance is $K_{max} + (S_s - S_a)$ and represents the portion of the input signal that is replicated in the final output. Alternatively a shift of $k(m) = K_{max}$ will result in the smallest distance between piles; in this case both W_m and W_{m-1} have been transposed by K_{max} and the distance between the pile and new seed remains $(S_s - S_a = S_a(\alpha - 1))$. The distance between piles, d, and thus the duration of input signal replicated, is given by:

$$\begin{aligned} (S_s - S_a) &\leq d \leq K_{max} + (S_s - S_a) \\ S_a(\alpha - 1) &\leq d \leq K_{max} + S_a(\alpha - 1) \end{aligned} \tag{3.8}$$

In speech where the multiple pitch periods occur over the interval $[0, K_{max}]$, multiple periods will be replicated as a unit during time-scale expansion. The replication of multiple pitch periods in this case increases the chance of perceived reverberation since amplitude differences in adjacent periods are preserved. This effect is detailed in Section 3.4.7.

Note that x may represent a single sample or a sequence of any number of samples. The feature x need not be a distinguishable feature. In all cases the current window will be maximally correlated with the existing signal when identical portions of the input signal overlap. As x may be any sequence of samples, it is apparent that this process will occur repeatedly. A sequence of samples, y, immediately following x will initiate the same process. As the sequence of samples x no longer appears in the current window, the new sequence of samples y plays the same role. The overlay process occurs continually, merely transposed along the time-axis.

This effect became apparent in observations of the crosscorrelation function for small analysis shifts. Each crosscorrelation frame contained a single distinctive peak for both periodic and random noise portions of the signal. The position of the peak in the frame changed by an amount $(S_s - S_a)$ in subsequent frames. The distinctive spike during random noise and silence portions of the output signal indicated that this high correlation was not due to periodicities in the signal. Figure 3-9 contains several crosscorrelation functions separated by short segments of zeros to distinguish the crosscorrelation functions of individual frames. The distinctive spikes correspond to shifts which overlap common regions of the input signal. The position of each spike moves an amount $(S_s - S_a)$ between frames since the shift to a overlay common region in the current window with a previous "pile" increases by $(S_s - S_a)$. In frames which do not contain the distinctive spike, no overlay with previous windows was possible.

3.4.4 Effect of Parameter Set on Feature Replication

Table 3.1 is provided to give an indication of the number of features produced by the interactions of the four parameters which lead to input signal replication during time-scale expansion.

From Table 3.1 we see that the number of features in the final output is determined by the interaction of all four parameters. For integral time-scale expansion, i.e. $\alpha = 2, 3, ..$ integer,

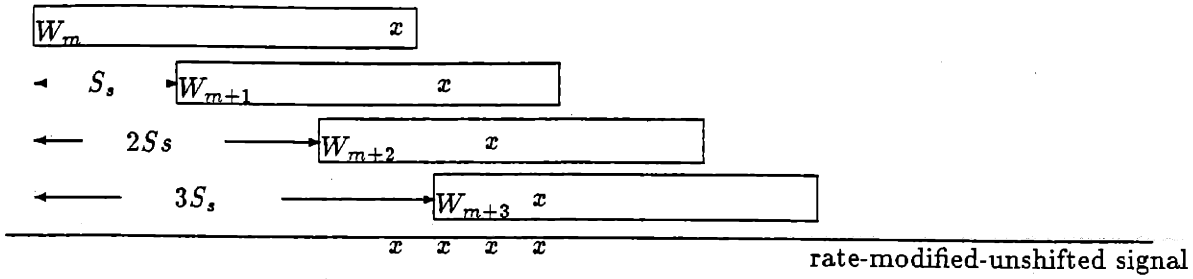


Figure 3-6: Expanded view of rate-modified-unshifted signal.

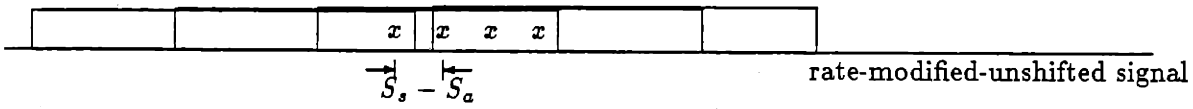
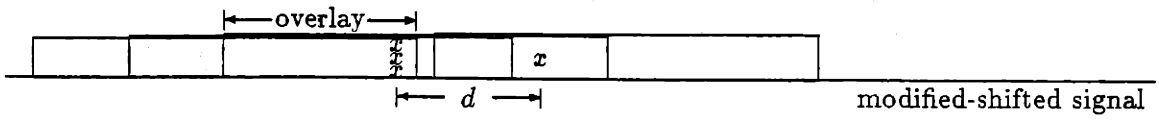


Figure 3-7: Replicated segment duration in rate-modified-unshifted signal.



$$(S_s - S_a) \leq d \leq K_{max} + (S_s - S_a)$$

Figure 3-8: Duration of replicated input segments is given by the distance between piles.

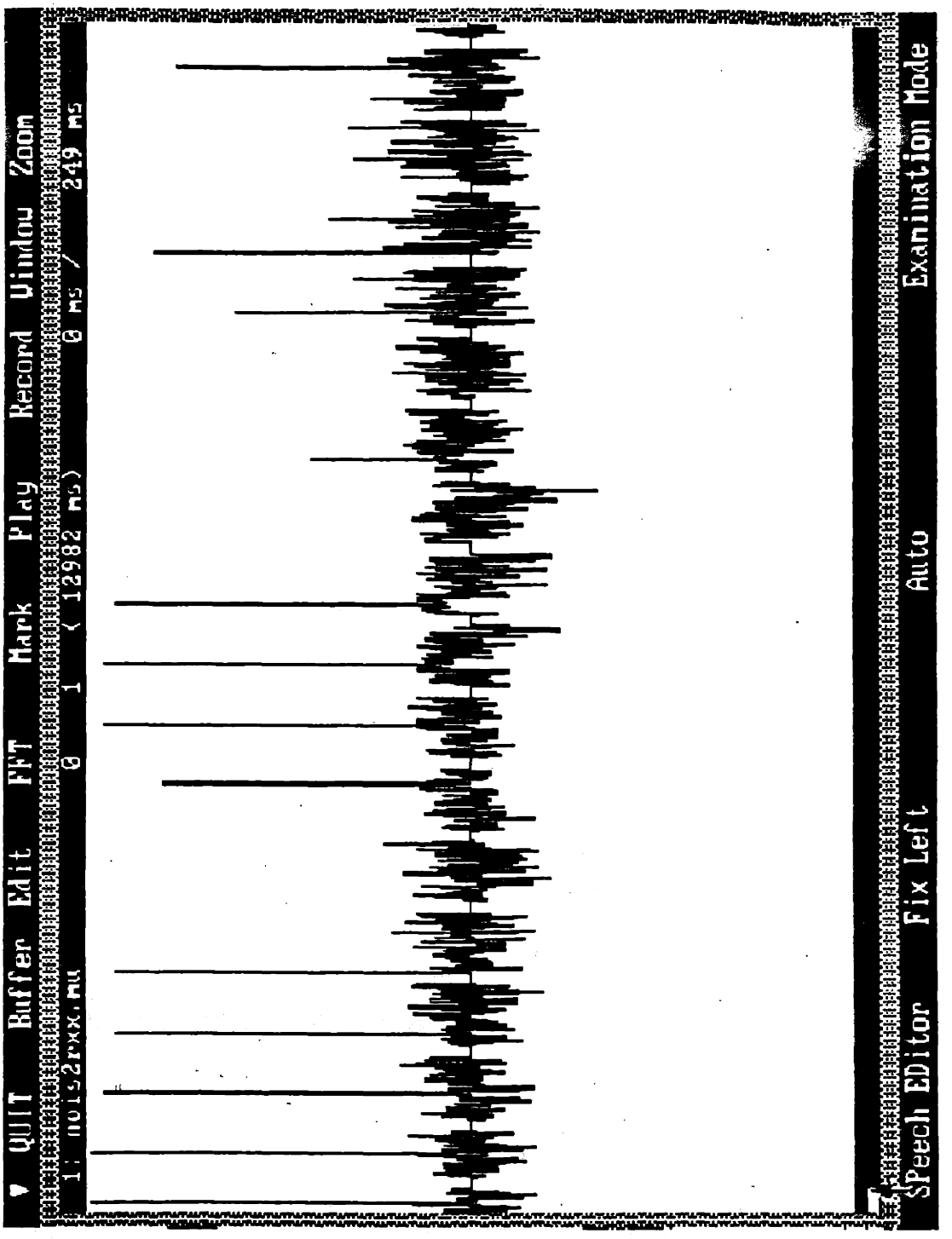


Figure 3-9: Crosscorrelation functions of several windows of random noise used by the SOLA algorithm for shift determination. Note the distinctive spikes which move by an amount $(S_s - S_a)$ between frames. This corresponds to the shift value increase necessary to overlay common regions of the input signal appearing in multiple windows.

α	*Winlen*	*Sa*	*Ss*	*Kmax*	$\frac{winlen}{Sa}$	Feature spread	Average # features
2	400	150	300	100	2.67	251	2.51
2	400	125	250	100	3.20	275	2.75
2	400	100	200	100	4.00	300	3.00
2	400	70	140	100	5.71	330	3.30
2	400	50	100	100	8.00	350	3.50
2	400	40	80	100	10.00	360	3.60
2	400	20	40	100	20.00	380	3.80
2	400	10	20	100	40.00	390	3.90
2	400	5	10	100	80.00	395	3.95
2	300	90	180	100	3.33	210	2.10
2	300	70	140	100	4.29	230	2.30
2	300	50	100	100	6.00	250	2.50
2	300	40	80	100	8.00	280	2.80
2	300	20	40	100	15.00	280	2.80
2	300	10	20	100	30.00	290	2.90
2	300	5	10	100	60.00	295	2.95
2	300	1	2	100	300.00	299	2.99
2	200	40	80	100	5.00	160	1.60
2	200	20	40	100	10.00	180	1.80
2	200	10	20	100	20.00	190	1.90
2	200	5	10	100	40.00	195	1.95
2	200	1	2	100	200.00	199	1.99

Table 3.1: Feature replication for time-scale expansion-by-two via several combinations of parameters. Entries are given in sample numbers (8 kHz sampling rate) with each sample corresponding to 125 μsec.

the number of features in the output signal after shifting should be exactly α. The data above shows that several combinations result in a number of features which exceeds α. The presence of these extra features is the source of perceived reverberation in the output signal.

Several favorable combinations appear: those in which the maximum spread of the features in the rate-modified-unshifted signal is an integer multiple of K_{max}. In this case there should be an integer multiple of full "piles" after the shifting operation. A full "pile" is one that contains n features where n is the number of overlapping windows giving rise to the output signal in a particular region. Of these favorable combinations, only the sets with $winlen = 200$ samples and $winlen = 300$ samples approach the desired number of replicated features with substantial envelope averaging.

We desire the maximum spread of features in the rate-modified unshifted signal to be an integer multiple of K_{max}:

$$[(\text{features in unshifted signal}) - 1](S_s - S_a) = \text{Maximum spread in unshifted signal}$$

$$\left[\frac{winlen}{S_a} - 1\right][S_a(\alpha - 1)] = (winlen - S_a)(\alpha - 1) = NK_{max}$$

$$\frac{(winlen - S_a)(\alpha - 1)}{K_{max}} = N \text{ an integer} \tag{3.9}$$

The fractional remainder of the left hand side of Equation 3.9 gives the fullness of the incomplete piles in the shifted signal. No fractional remainder corresponds to completely filled piles, while 0.5 corresponds to initial or final piles which are half full. It is obvious that slight changes in the value of K_{max} make it relatively easy to meet this goal. The ideal parameter sets appear to be those combinations which have as large an S_a as possible while maintaining an average number of features in the output signal equal to α.

Unfortunately this is not the case. The observed output using these parameters is stutterant, and amplitude envelope discontinuities are readily detected in the output. This is attributed to two factors: the long duration of replicated segments (fixed by $S_a(\alpha - 1)$), and low averaging. Discontinuities occur when the averaging factor is extremely low (SWO$\approx$ 1.5) because there is no smoothing in the transitions between un-overlapped and overlapped portions of the signal. The averaging factor improves with smaller synthesis shifts ($S_s = \alpha S_a$),

but the corresponding decrease in analysis shift, S_a, increases computations. These two contrasting relationships work against each other and reinforce the quality at computational cost tradeoff. Note the influence of the time-scale modification factor, α. The beneficial output signal averaging, SWO= $\frac{winlen}{\alpha S_a}$, decreases with increasing α for a fixed value of S_a, and thus fixed computation. This indicates that the quality of the output signal will suffer as α increases without a corresponding increase in computation (decrease in S_a). The computation/quality tradeoff must be weighted differently for different time-scale modification factors. No choice of parameters will satisfy both for all scale changes.

3.4.5 Pre-Echo in Time-Scale Expanded Signals

If the length of replicated segments in the time-scale expanded signal is longer than 12.5 msec, these segments are detected as distinct repetitions of speech fragments in the output signal. This leads to stuttering and reverberation in the expanded signal. Figure 3-10 exemplifies this effect. Note the fragment of speech present before the onset of the utterance. This leads to clicks and stuttering in the output signal because too long a segment of the input signal is replicated rather than single pitch periods. These effects significantly detract from signal quality. Methods to reduce this effect are explored in Chapter 5.

3.4.6 The Role of "SWO" in Minority Feature Attenuation

Let us now step back and examine the effects of "SWO" and "piles" in the context of the expansion process described so far. During time-scale expansion with $(S_s - S_a) \leq K_{max}$, the variability, K_{max}, in the actual synthesis shift used allows multiply-overlapping windows to regain their analysis configuration (See Figure 3-8). Each new window is shifted to a position S_a points further along the time-axis than the previous window (i.e. its position during analysis). Such a configuration tends to maximize the crosscorrelation provided the output signal resembles the input signal over the shift interval. This leads to segments of the unnormalized output signal consisting of multiple overlays of some region of the input signal. The term "overlay" is used to describe the case when the same feature appears in multiple windows and the window are positioned such that these features are aligned as in Figure 3-8. When this region of the output is normalized, we obtain an identical segment of the input

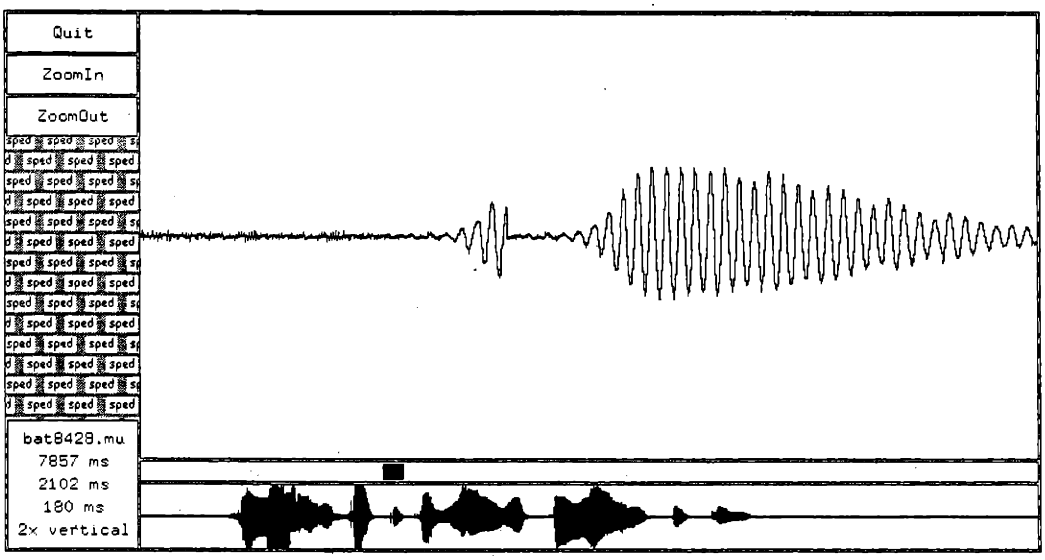


Figure 3-10: Pre-echo due to excessively large analysis shift ($S_a = 12.5$ msec, $winlen = 37.5$, $K_{max} = 12.5$ msec, $S_s = 2$).

signal.

When it is not possible for a window to obtain the offset with previous windows used during analysis, no overlay with previous windows will occur. Rather the most similar regions are overlapped. Although subsequent windows may overlap this region, overlays are no longer possible as the region no longer corresponds to a segment of the input. Overlays may occur in portions outside the region of overlap of a newly added window, however, since this region represents an unaltered segment of the input.

Many regions of the output signal are comprised of multiply-overlapped windows in which one or more of the overlapping windows contain different portions of the input signal. The sum of these similar regions from different sections of the input signal are divided by the total number of overlapping windows. If such a region consists of the sum of 6 overlays of a feature (or segment) x from the input signal and a similar but different feature (or segment) y, the resulting region of normalized output will strongly resemble the feature (or segment) x. Note in this case the feature (or segment) y has been attenuated and feature (or segment) x predominates in the normalized output. Although both x and y are similar, the minority feature y is attenuated. In a region consisting of 3 overlays of y and 3 overlays of x, the output will contain characteristics of both x and y in equal proportion.

3.4.7 The Role of "Piles" and "SWO" in Transitions, Perceived Reverberation, and Minority Feature Attenuation

Now that the role of the "piles" and the concept of "overlay" have been examined, let's explore additional benefits of using low analysis and synthesis shifts. During expansion with $(S_s - S_a) \leq K_{max}$, the algorithm repeatedly overlays several windows then crosscorrelates when overlaying is no longer possible for the new window. The process of overlaying then begins anew with the un-overlapped portion of the window added. This *Expand on Demand* process is represented pictorially in Figure 3-11.

In the modified regions of the output signal (regions where overlays with previous windows were not possible), a staircase fade between original regions occurs. The distance between steps is S_a. For small S_a, the staircase approaches a linear fade. Note in Figure 3-11 the start of the first modified region contains three overlays of a previous input signal segment and one

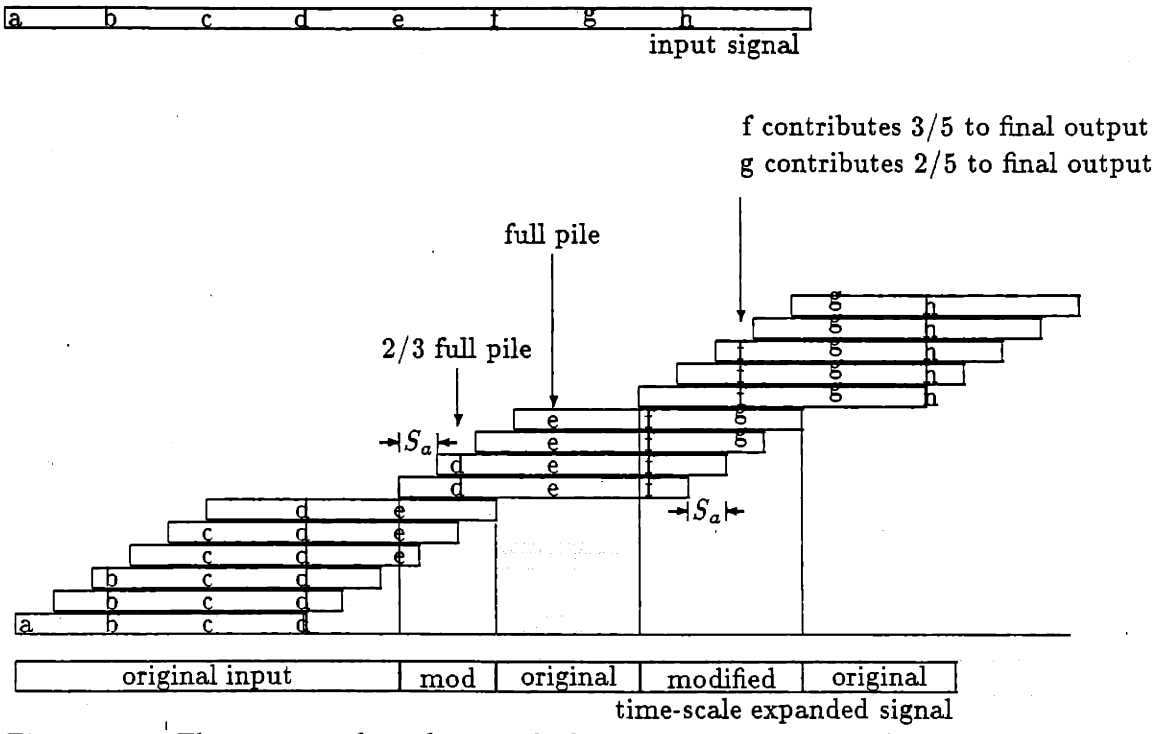


Figure 3-11: The process of overlaying which occurs during time-scale expansion with small analysis and synthesis shifts.

window of more recent input signal with similar characteristics. The end of this modified region contains a single window of the previous input signal segment and three overlays of the more recent input signal segment.

The transition between "overlay" regions and modified regions can result in an amplitude discontinuity when S_a is too large. The overlay region which corresponds to a segment of the input signal, is abutted with a similar periodic region of the new window. When SWO is low ($\frac{winlen}{\alpha S_a} < 4$) the transition to the new window is abrupt because little averaging occurs. In many cases the periodic region of the new window will have significantly different amplitude. A large amplitude difference leads to a discontinuity in the time-scaled signal. Figure 3-12 illustrates this phenomena. Amplitude discontinuities are readily apparent in the time-scale expanded signal. When large analysis shifts are used, little averaging occurs in the modified signal; and the amplitude of each windowed segment of input is preserved. Pre-echo associated with large analysis shifts is also readily apparent in the expanded signal.

The increased quality associated with small S_a during time-scale expansion can be attributed to these effects which occur in the modified regions of the output signal. Note that a longer window increases envelop averaging (increases SWO) by $winlen/\alpha S_a$, while the number of features (indicative of AWO) is increased by only $winlen/S_a$. This indicates a longer window will give better results for a fixed analysis shift. For large analysis shifts, $S_a > 6.25$ msec (50 samples), during time-scale expansion, $S_s = 2S_a$, the quality of the time scale expanded signal produced using $winlen = 25$ msec (200 samples) was noticeably improved when compared to the signal produced using $winlen = 37.5$ msec (300 samples). This improvement is due to minority feature attenuation of pre-echo which is not noticeably improved by using different updating schemes with $winlen = 25$ msec (200 samples) (see Section 5.9.4 and 5.9.3).

It is interesting to note that the high-quality associated with low analysis shifts results from the nature of the reconstruction process rather than a more detailed analysis of the input signal. An analysis shift of 2.5 msec (20 samples) corresponds to an advance of less than a single pitch period for all but the highest-pitched speakers. The subsequent synthesis shift is approximately a single pitch period in length for female speakers, and less than half a single pitch period for male speakers when $\alpha = 2$. Operation with such a low analysis shift is clearly inefficient since only a fraction of the signal changes between windows. The

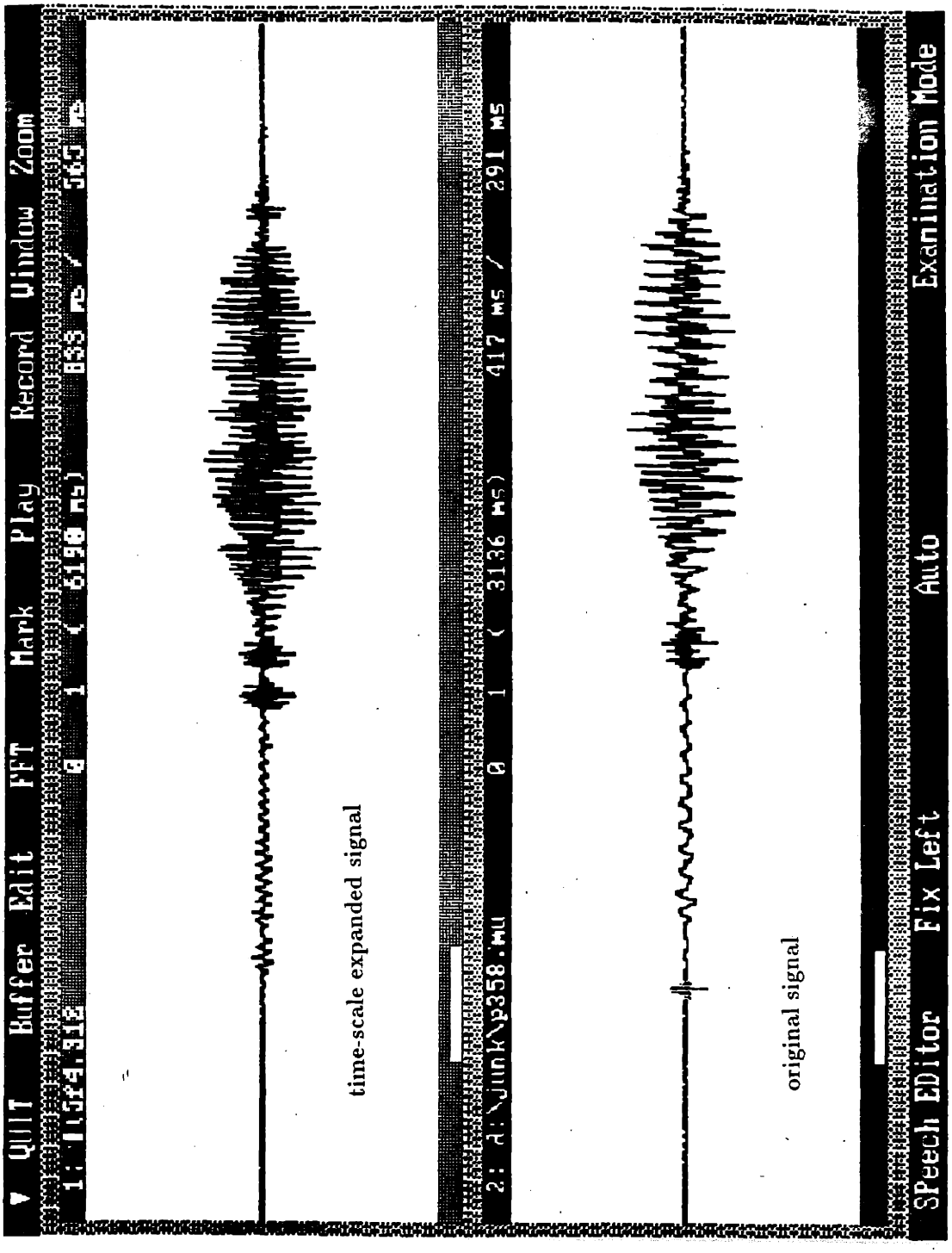


Figure 3-12: Original and time-scale expanded-by-two signals. The time-scale expanded signal (using *winlen* = 40 msec, S_a = 12.5 msec, S_s = 25 msec, and K_{max} = 12.5 msec) displays considerable pre-echo and amplitude discontinuities.

real improvement in quality arises from the interaction of the parameters leading to overlays, minority feature attenuation, and smooth transitions in the rate-modified signal. These results were first observed for time-scale expansion of a signal which contained a single impulse using various analysis shifts. Figures 3-13 through 3-17 illustrate the effects of minority feature attenuation and the distance (time) between replicated features. Note in the figures that the distance between impulses ("piles") increases with increasing analysis shift. This follows directly from Equation 3.8. The attenuation of minority features, the leading and lagging replicated features in the figures, decreases with decreasing SWO (increasing analysis shift). This follows directly from Equation 3.3.

3.4.8 Summary

Each sample in the rate-modified shifted signal is scaled by the number of overlapping windows summed to create that sample. Thus all windows contribute equally. Shifts that do not overlay previous windows, those truly utilizing the normalized crosscorrelation calculation, lead to the creation of new "piles" some distance $((S_s - S_a), K_{max} + (S_s - S_a)]$ from previous "piles". In most cases the window length will be greater than $2K_{max}$ (Section 2.3.1) so that the tails of windows comprising the previous pile overlap with the window containing the new "pile". The sum of the overlapping portions are divided by the total windows contributing to the sum, weighting a feature in the final output by its number of occurrences in the unscaled, rate-modified signal. The greater the number of average overlapping windows, SWO, the greater the attenuation for incomplete "piles". The quality of the output signal is increased for higher resolutions because as the number of overlapping windows contributing to the output signal increases the attenuation of incomplete or half empty piles increases.

3.5 Prediction

The process described in Section 3.4.3 indicates that a portion of the shift values returned by the algorithm are entirely predictable when a given feature appears in multiple windows and the distance between features appearances in the rate-modified-unshifted signal is less than K_{max}. Section 3.3.2 shows that for slow-down, portions of the input signal will always be

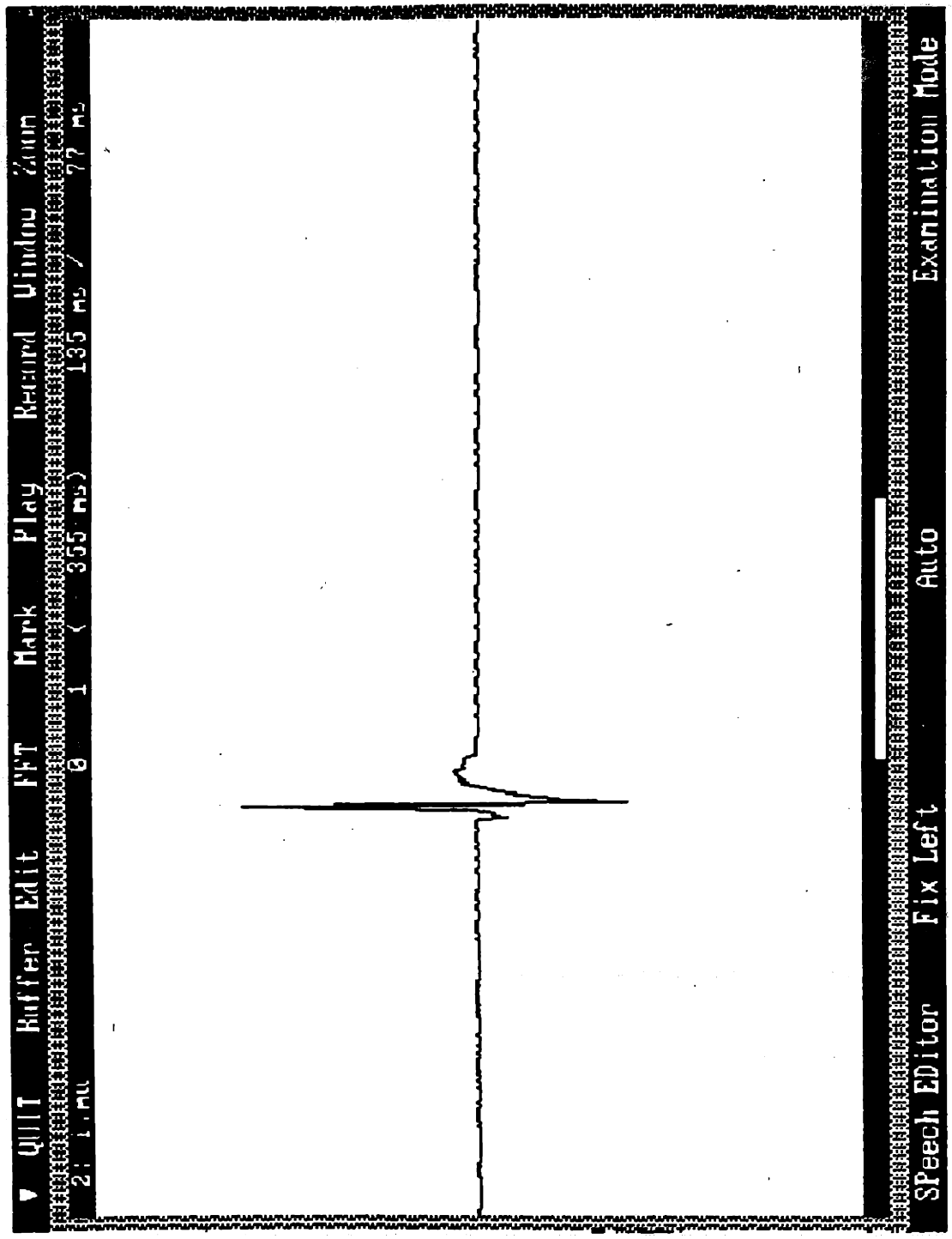


Figure 3-13: Original impulse used in time-scale expansion tests.

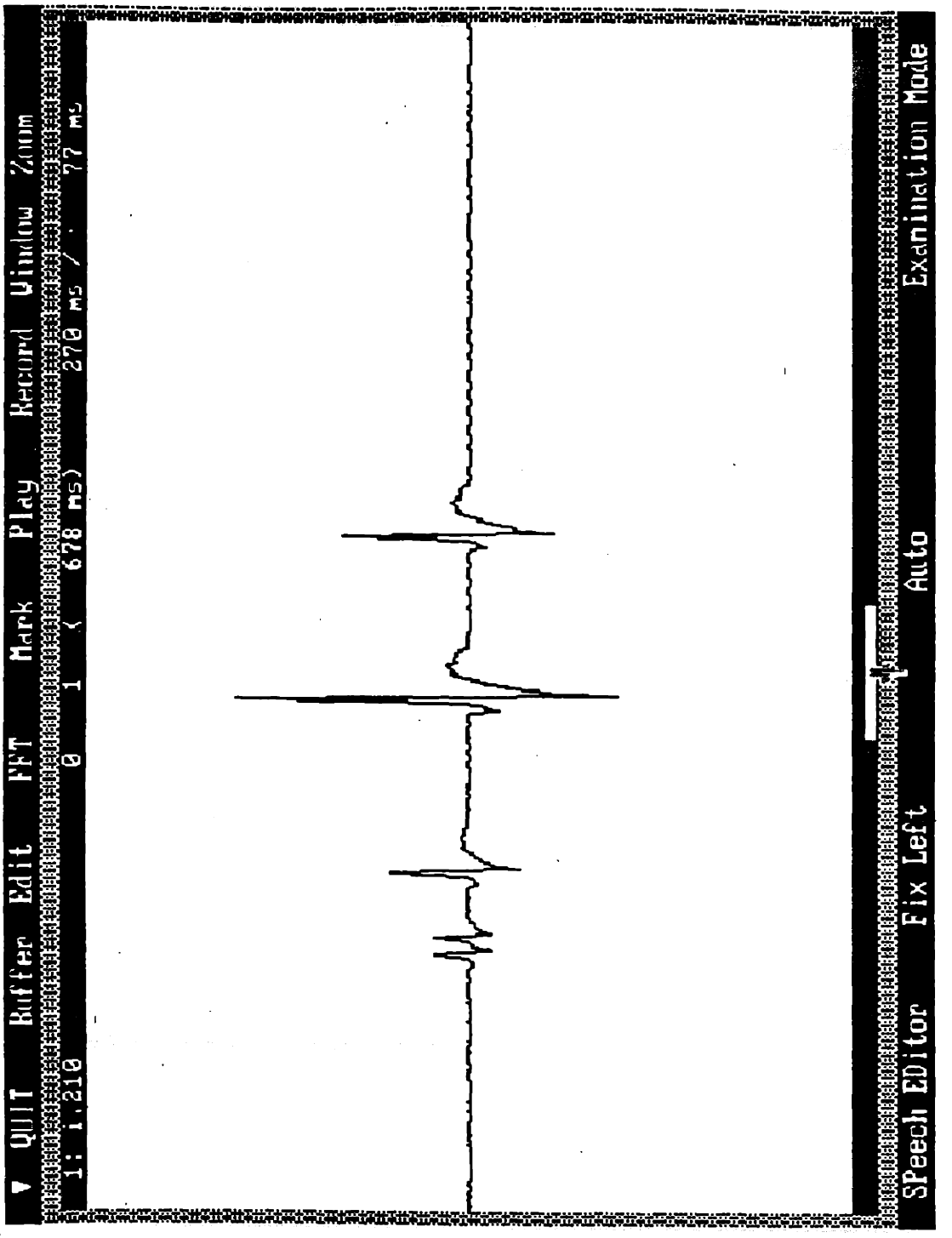


Figure 3-14: Time-scale expanded-by-two output of impulse using *winlen* = 25 msec, S_a = 1.25 msec, S_s = 2.5 msec, K_{max} = 12.5 msec.

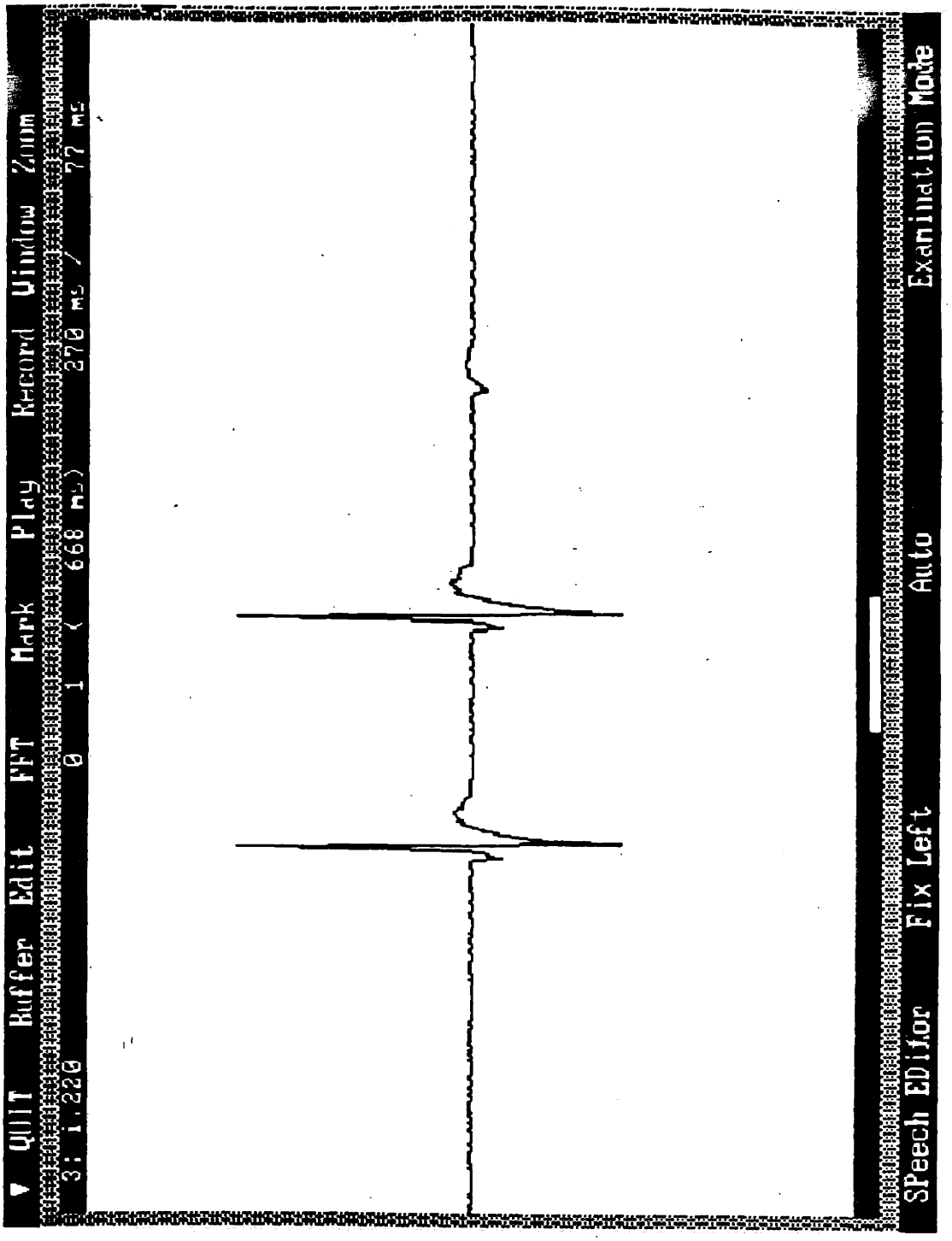


Figure 3-15: Time-scale expanded-by-two output of impulse using $winlen = 25$ msec, $S_a = 2.5$ msec, $S_s = 5.0$ msec, $K_{max} = 12.5$ msec.

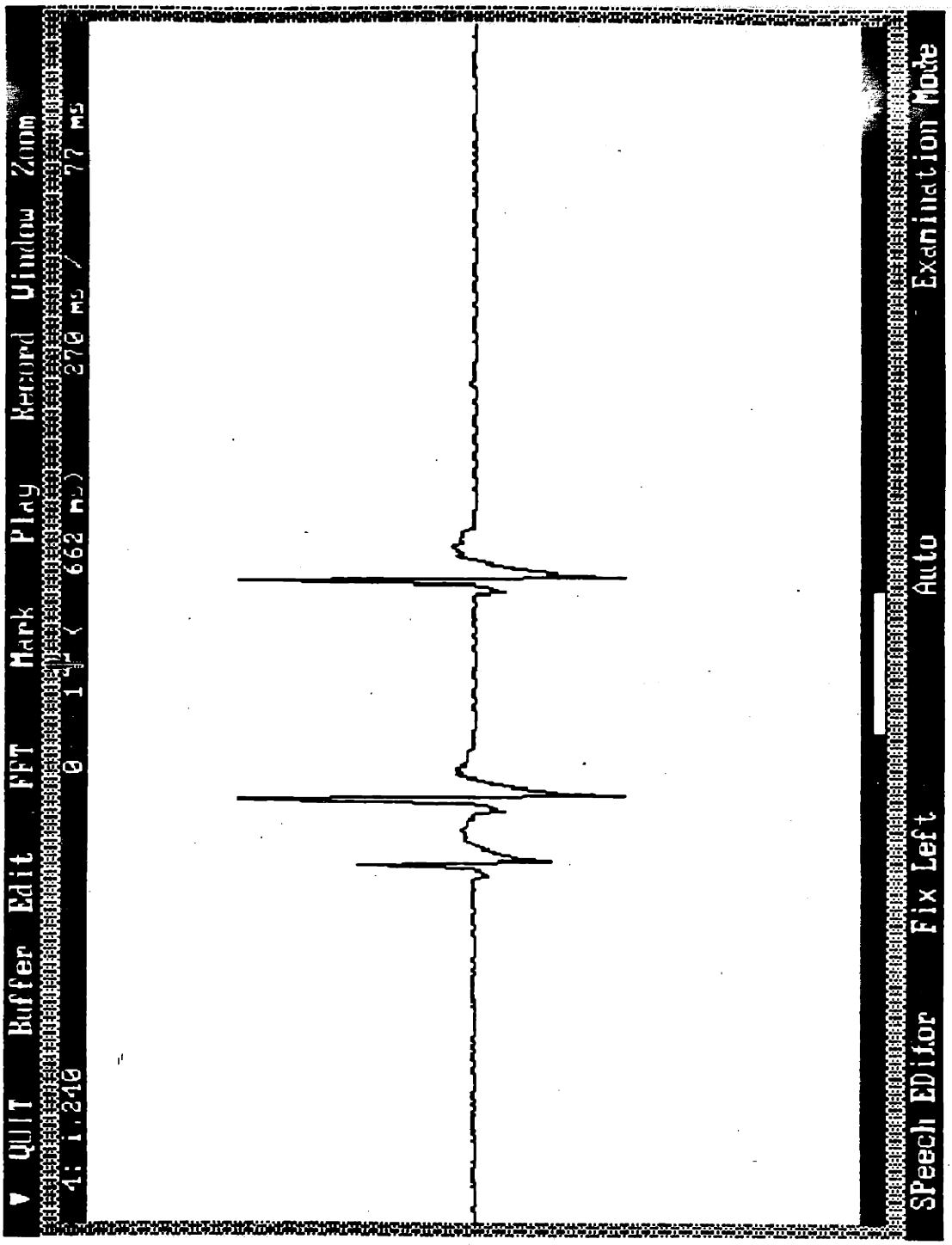


Figure 3-16: Time-scale expanded-by-two output of impulse using *winlen* = 25 msec, $S_a = 5$ msec, $S_s = 10$ msec, $K_{max} = 12.5$ msec.

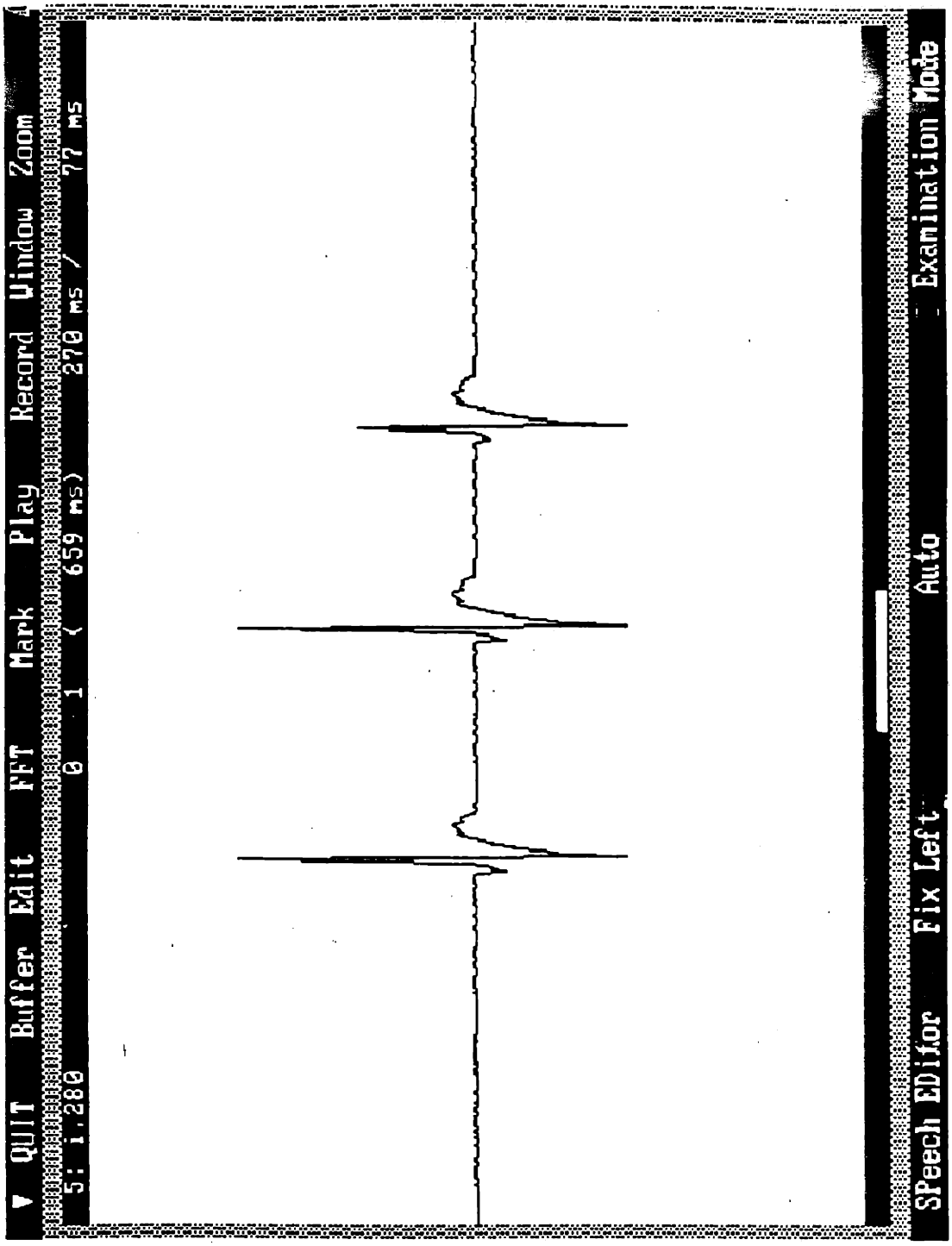


Figure 3-17: Time-scale expanded-by-two output of impulse using $winlen = 25$ msec, $S_a = 10$ msec, $S_s = 20$ msec, $K_{max} = 12.5$ msec.

contained in multiple windows, and the analysis shift may be chosen so the distance between features allows prediction.

The distance between successive feature appearances in the rate-modified-unshifted signal is $(S_s - S_a)$. A backward shift $k(m) = S_s - S_a$ for the current window, W_m, will align the feature appearance in the current window with the feature appearance in the previous window, W_{m-1}. If the previous window has been shifted backward by an amount $k(m-1)$, the current window must be shifted by an amount $k(m) = k(m-1) + (S_s - S_a)$ to align the feature appearances in W_m and W_{m-1}. If this distance is less than K_{max}, then the normalized crosscorrelation calculation is unnecessary for determining the shift of the current window as the value maximizing the crosscorrelation can be predicted. Thus:

$$k(m) = \begin{cases} k(m-1) + (S_s - S_a) & \text{for } k(m-1) + (S_s - S_a) \leq K_{max} \\ \max R_{xy}^m(k) & \text{for } k(m-1) + (S_s - S_a) > K_{max} \end{cases} \tag{3.10}$$

In a similar manner the shifts for time-scale compression may also be predicted. However, this would only arise in an implementation where multiple windows contain the same feature and the shift to overlay features is within the shift search interval. In such an implementation, small analysis shifts would permit overlays to occur on the shift interval and thus certain shifts would be predictable. For time-scale compression the analysis shift is in general large enough $(S_a \approx K_{max})$ that prediction cannot and need not be utilized.

During time-scale compression the distance between appearances for compression is also given by $(S_s - S_a)$ only now $S_a > S_s$. The features move backward in time with each appearance. For compression it was shown that no features should appear in more than one window and the analysis shift is such that the quantity $(S_s - S_a) = S_a(\alpha - 1)$ is greater than K_{max} for reasonable choices of S_a and K_{max}.

3.6 Concluding Summary of Parameter Set Equations

Now that the interactions of the parameters have been outlined, the solution space for the parameter set can be defined. The constraints fall into two classes: those arising from the nature of implementation, algorithm defined constraints; and those which determine output

signal quality, quality defined constraints. The two classes of restrictions compete in many instances and define the quality versus computation tradeoff.

This section recapitulates the previously outlined constraints. In summary we have the following set of linear equations:

3.6.1 Algorithm Defined

Maximum window displacement: Fixed value.

$$K_{max} > \text{Largest Pitch Period in Speech Signal}$$

$$K_{max} > 110 \text{ samples}$$

$$K_{max} > 13.75 \text{ msec}$$

Number of overlapping points for indication of correlation: Fixed value.

$$Ola_{min} > 30$$

$$\frac{winlen - K_{max} - Ola_{min}}{\alpha} \geq S_a$$

Order of Computation: *Goal:* Minimize.

(computation for indication of correlation) * (range of shift values) * (number of frames)

$$\bigcirc\left[\frac{(winlen - \alpha S_a)(K_{max})^2}{S_a}\right] = \bigcirc\left[\frac{(winlen - S_s)(K_{max})^2\alpha}{S_s}\right]$$

3.6.2 Quality Defined

Time-Scale Compression

Averaging in output signal: *Goal:* Minimize.

$$\frac{winlen}{S_s} = \frac{(winlen)\alpha}{S_a}$$

Multiple window appearances of input signal segments: *Goal:* Eliminate.

$$\frac{winlen}{S_a} = 1$$

Reverberation in output signal is eliminated by preventing any input feature from appearing in multiple windows.

Time-Scale Expansion

Averaging in output signal: *Goal:* Maximize.

$$\frac{winlen}{S_s} = \frac{(winlen)\alpha}{S_a}$$

Multiple window appearances of input signal segments: Always greater than 1.

$$\frac{winlen}{S_a} > 1$$

Eliminating reverberation in output signal by preventing any input feature from appearing in multiple windows is not possible. Reverberation is reduced, however by high averaging during output signal construction.

3.7 Summary

Chapter 2 presented the SOLA algorithm and defined the four parameters inherent in the algorithm. In this chapter the concept of a parameter set was defined as constraints on parameter combinations were introduced. Only certain combinations of parameters provide the desired high-quality time-scale modified speech. Events associated with reverberation, stuttering, and poor quality in the time-scaled signal were shown to be consequences of parameter interactions in poorly chosen parameter sets. The source of perceived reverberation in the time-scale modified output signal produced by the SOLA algorithm was examined in detail. The concepts of "features", "piles", and "overlays" were introduced and used to support the observations of output signal quality for various parameter sets. The effects of low analysis shifts, shift interval duration, and window length on reverberation have been provided in close

detail. The observed improvement for time-scale expansion with low analysis shifts was supported by the theorized interaction of the parameter set. Understanding the interaction of the parameter set for time-scale expansion led to a method of predicting the shift values in many cases, reducing the number of costly shift determinations.

Chapter 4

Robustness of SOLA algorithm

This chapter is intended to give a relative measure of the robustness of the SOLA algorithm for both time-scale compression and expansion. Several signals were processed in an attempt to "fool" the algorithm. Throughout these tests the algorithm has proven to be very robust.

4.1 Speech Containing Noise

In telephony applications speech is often corrupted by varying amounts of random noise. This noise is usually apparent as a slight background hiss in the signal. Even under ideal circumstances digitized speech will contain a very slight amount of background noise associated with quantization during the analog to digital conversion. Some amount of random noise in present in almost all speech signals. Occasionally telephony speech will contain correlated noise. Correlated noise is simply an echo of the original signal delayed by some amount. This type of noise occurs less frequently than random noise in modern communication systems, but can significantly degrade signal quality.

A robust time-scale modification technique for use in telephony applications must be able to operate on signals containing noise. Ideally such a technique would provide uniform quality across the range of modifications desired regardless of the noise contained in the signal. The performance of the SOLA algorithm on signals corrupted by significant amounts of noise is detailed below.

4.1.1 Speech with White Random Noise

Random noise is by nature highly uncorrelated. Voiced speech on the other hand is highly correlated. Thus the normalized crosscorrelation used to determine the shift values remains relatively unaffected by random noise in speech signal. The crosscorrelation function of noisy speech no longer resembles a smooth continuous waveform for voiced portions of the signal. Random noise in the speech signal leads to noise in the correlation waveform, however, the characteristics of the correlation waveform such as period and amplitude are preserved. Errors in alignment due to noise in the correlation waveform are masked by the random noise in the speech signal. Those portions of the speech signal composed of random noise are not affected by the added random noise as discontinuities in such signals remain undetected for the most part.

Tests Performed

To determine what effect, if any, random noise has on the SOLA algorithm, random noise was added to utterances before and after time-scale modification. The random noise was added such that the segmental SNR[1] varied over the signal from a maximum of 10.53 to a minimum of -43.80. Measured with 0.192 second damping[2], the damped segmental SNR had a maximum of 4.21 and a minimum of -32.19.

In the following tests, time-scale expansion was accomplished using a window length of 25 msec (200 samples), analysis shift of 2.5 msec (20 samples), synthesis shift of 5 msec (40 samples), and a maximum shift value of 12.5 msec (100 samples) on speech sampled at 8 kHz. Time-scale compression was accomplished using a window length of 32 msec (256 samples), analysis shift of 32 msec (256 samples), synthesis shift of 16 msec (128 samples), and a maximum shift value of 12.5 msec (100 samples).

Both male and female speech were time-scale compressed and expanded via the SOLA algorithm. White random noise was then added to these time-scale modified signals to create several "standards" for comparison purposes. Random noise was also added to the original

[1]The power in each signal was computed over segments of 32 msec (256 samples).

[2]Segmental SNR with 0.192 second damping was calculated by averaging 6 consecutive single segment SNR values.

utterances from male and female speakers to create "noisy utterances". These "noisy utterances" were time-scale modified (compressed and expanded) by the same factors as the standards. Comparisons of the time-scale modified "noisy utterances" and the "standards" indicate that random noise has no effect on the quality of the speech produced by the algorithm. For both expansion and compression the speech corrupted by noise suffered no loss in intelligibility during processing. The noise floor was slightly modified during time-scale expansion however.

Time-scale expansion via the SOLA algorithm produces slightly noticeable changes in the noise. Replicating portions of the input signal is inherent to the operation of the algorithm during expansion. This replication occurs at quasi-periodic intervals along the output signal. The replication becomes slightly audible as a subtle warble in the random noise background of utterances containing significant background noise. The effect is most pronounced for male speech expanded by a factor of two. This effect was not observed for female speech or time-scale compressed male or female speech. The detectable change in the noise background for male speech which was not apparent for female speech is most likely due to the greater fluctuations along the shift interval associated with the longer local period in the signal. Specifically, the presence of a single or partial period in the replicated portions of the input signal may place added emphasis on the background noise as a single period takes on more characteristics of the noise than multiple periods.

The change in the random noise background during time-scale expansion prompted experiments using only random noise as input to the SOLA algorithm. The same warbling was noted, and increased when the output of one expansion was used as input for subsequent expansions. After several expansions the output was distinctly periodic. This result is expected for time-scale expansion as portions of the input signal are replicated in a quasi-periodic manner. Subsequent expansions lock on to these quasi-periods and reinforce the slight correlation. The highest amplitude frequency with a period smaller than K_{max} will be successfully aligned via the crosscorrelation and consequently reinforced as other frequencies are added without regard to phase and tend to be attenuated due to destructive interference.

Operation of the SOLA algorithm on signals containing random noise provided significant insight. The results obtained by repeating processing of random noise exposed the quasi-

periodic nature of the SOLA algorithm. During time-scale modification of a speech signal, these quasi-periods in the algorithm operate in synchrony with the local periods in the signal and remain undetected. Observation of the crosscorrelation function of random noise also exposed the predictability of shift values during time-scale expansion with low analysis shifts (See Section 3.4.7).

4.1.2 Speech with Correlated Noise

Reverberation or echo in a signal is highly correlated with the original signal. This correlated noise alters the signal and affects the crosscorrelation used to determine the shift values. The delayed attenuated signal interferes constructively and destructively with periods in the original signal. This interference can significantly alter the periodic structure of the resulting signal. Subsequent time-scale modification of reverberant signals should at best preserve this altered periodic structure. Recovering the original signal from a reverberant signal is an active research topic known as echo cancellation.

When two periodic signals are added, the period of the resulting signal can approach the greatest common multiple of the two individual periods. Reverberation in speech can lead to signals whose local period may exceed K_{max}. Theoretically this is cause for concern, however, tests performed indicate that possible fracturing of periods greater than K_{max} occur rarely in telephony speech and remain undetected in time-scale modified reverberant signals.

Just as the periodic structure of a reverberant signal will differ from the original, the crosscorrelation of a reverberant and original signal will differ. The crosscorrelation function is corrupted by the correlated noise and the characteristics of the function are significantly changed. The time-scale modified speech is of the same perceived quality as the reverberant input signal. The algorithm will align the greatest amplitude local periods in the signal. Thus the original signal and echo signal are aligned with themselves rather than each other assuming the original signal has greater amplitude in the region of overlap. The alternative aligns the original signal with echo and the echo signal with nothing, which should not have as high a crosscorrelation.

Tests Performed

To create a reverberant signal, an echo signal was constructed by delaying an utterance by 32 msec (256 points for 8 kHz sampling) and scaling it by 0.3. This echo was then added to the original signal to create the reverberant test signal. The length of the delay was chosen such that the reverberation was easily detected without rendering the test signal unintelligible. The performance of the algorithm in this test should reflect performance for different delays.

Both male and female reverberant test signals were constructed in this manner. These signals were then time-scale expanded using a window length of 25 msec (200 samples), analysis shift of 2.5 msec (20 samples), synthesis shift of 5 msec (40 samples), and a maximum shift value of 12.5 msec (100 samples) on speech sampled at 8 kHz. Time-scale compression was accomplished using a window length of 32 msec (256 samples), analysis shift of 32 msec (256 samples), synthesis shift of 16 msec (128 samples), and a maximum shift value of 12.5 msec (100 samples).

No increase or decrease in reverberation was detected for male or female speech after compression or expansion. The performance in this test is harder to gauge accurately as no "standards" exist. Creating a reverberant signal from the compressed or expanded original signal yields substantially different output. Thus subjective measures must be used.

No period-fracturing due to local periods exceeding K_{max} in male speech was detected in the output signal although a small number of plots of the crosscorrelation functions did not contain complete periods. While this limited test set does not rule out the possibility of such occurrences being detected, it indicates that most instances of fracturing involving frequencies lower than 80 Hz remain undetected in the reverberant signal. Section 4.2 explores this observation in further detail.

4.2 Speech Containing Multiple Speakers

Speech containing multiple speakers represents a complex waveform whose maximum local period can significantly exceed the sum of the two individual periods. This presents a problem when both speakers have low pitch which can be partially corrected by increasing K_{max}. Note from Section 3.4.3 however that the value of K_{max} affects the duration of the input signal

replicated during time-scale expansion. Therefore K_{max} must remain less than a window length to insure uniform time-scale modification of the output signal.

Tests Performed

Test files containing two speakers were created by adding utterances from two male speakers, and utterances from a male and female speaker. These two test files were then processed and compared with "standards" obtained by time-scale expanding and compressing the utterances individually before adding them. The test files containing multiple speakers were processed using the same parameter set as speech from a single speaker.

The SOLA algorithm provided high-quality time-scale modified speech containing multiple speakers over the desired range of compression and expansion. No degradation due to period-fracturing was detectable in the utterance containing two male speakers. It was noted that tracking a single utterance in multiple utterances by different speakers can be quite difficult. The ability to track a single utterance was not improved in the time-scale expanded signal, and was virtually impossible in time-scale compressed signals.

Speech processed with K_{max} equal to twice the pitch period actually resulted in speech of lower quality due to the long duration of replicated segments of the input signal. Refer back to Section 3.4.3 for details.

The performance of the algorithm using parameters for single speakers on speech containing multiple speakers is likely due to the fact that frequencies below 70 Hz are sharply attenuated in telephony speech. Even in studio-quality recordings the absence of frequencies whose periods approach two male pitch periods are often undetected. Frequencies in this range do not normally occur in speech and thus tend to be ignored when listening to speech. If the local period in multiple-speaker utterances exceeds that of K_{max} (i.e. below 80 Hz), some period fracturing will occur. Recall however that the crosscorrelation will successfully align the highest amplitude frequency above 80 Hz. The results obtained from this test indicate that only frequencies above 80 Hz are detected by the human ear when listening to speech in telephony environments, or frequencies below 80 Hz are undetected due to their attenuation.

4.3 Complex Non-Speech Signals

Non-Speech signals such as music containing electric guitar and piano were expanded and compressed via the SOLA algorithm. The performance of the algorithm on non-speech signals was better than expected. The crosscorrelation aligns the highest amplitude frequencies present in adjacent windows. In most cases these higher amplitude frequencies provide the most essential information used by the human ear. The music was processed using the same parameter sets as for voice, allowing frequencies down to 80 Hz to be aligned. The resulting output was easily recognizable and of good quality. In telephony applications, frequencies below 70 Hz are sharply attenuated, thus any misalignment of these low-amplitude frequencies will result in extremely small discontinuities.

4.4 Robustness of the Crosscorrelation For Shift Determination

In cases where the crosscorrelation function is distorted or excessive precision rounding occurs, the 2nd highest peak may be selected for the shift value. This can have two effects depending on the duration of the signal's period. Recall that the crosscorrelation function is periodic with the local period in the overlapping portions of the windows. The function is therefore periodic with the pitch pulse excitation. Figures 4-1 and 4-2 represent graphs of the crosscorrelation of several frames of voiced speech. In the graphs, the crosscorrelation functions of two windows in a given frame are separated by short segments of zeros.

If the largest amplitude periodicity in the signal is less than half K_{max}, the 2nd highest peak will most likely correspond to alignment of the current window with an alternate period. This can be seen in the crosscorrelation function of windows of voiced female speech in Figure 4-2. The shift returned is offset by an amount equal to the duration of the periodicity in the signal. This will provide good alignment in most cases and not give rise to drastic discontinuities.

If the largest amplitude periodicity in the signal is approximately K_{max}, only one period of the crosscorrelation will occur on the interval [0, K_{max}]. The longer duration of the period in this case results in a rounder peak. This can be seen in the crosscorrelation function of windows of voiced male speech in Figure 4-1. In this case the second highest peak will occur

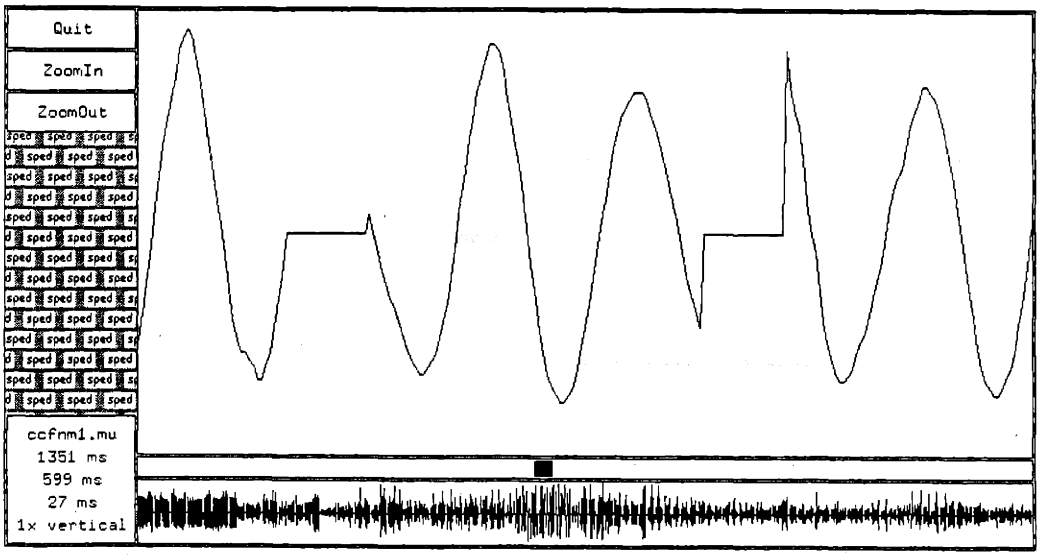


Figure 4-1: Crosscorrelation function of several frames of voiced male speech.

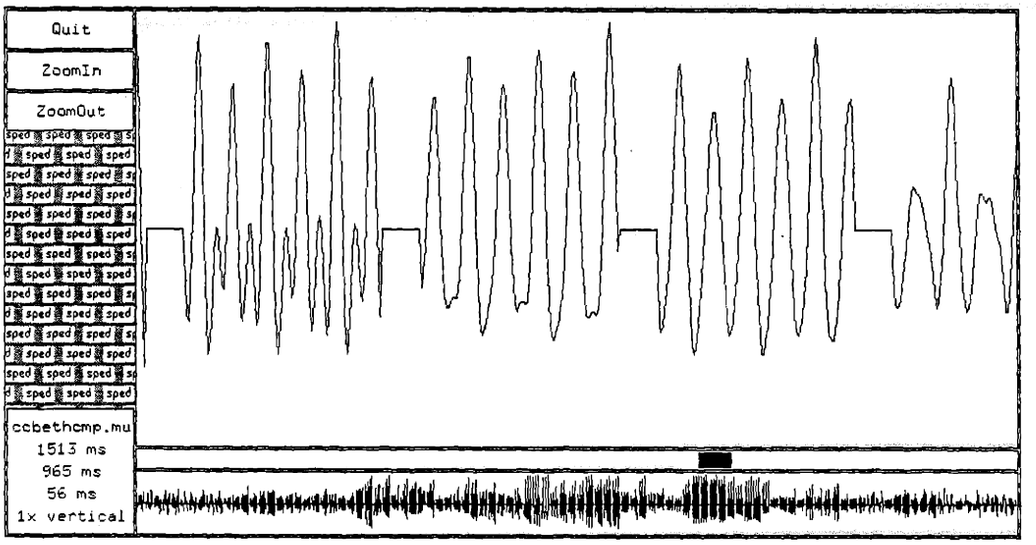


Figure 4-2: Crosscorrelation function of several frames of voiced female speech.

one sample to either side of the highest peak. This shift returned will be offset by one sample. This small misalignment does not have a drastic effect however as such a delay represents a small percentage phase lag for the longer duration period.

To simulate errors, the highest peak of the correlation function was masked during processing. The speech obtained from this processing was indistinguishable from the speech processed with the standard implementation. This result is exploited in Chapter 5 to reduce the computational requirements for shift determination.

4.5 Conclusion

This Chapter has shown the SOLA algorithm to be very robust for speech corrupted by noise which is present in many telephony applications. The algorithm performance was not significantly affected by random or correlated noise. In addition, the algorithm performed equally well on bandlimited, telephony-quality music.

Note that no objective measure is available for evaluating the performance of the algorithm with the standards. This is an unfortunate consequence of the locally optimal nature of the shifts selected by the algorithm. An accurate signal to noise ratio calculation requires identical phases in the reference and test signal.

Chapter 5

Computational Savings

5.1 Introduction

This chapter details several methods which reduce the SOLA algorithm's computational requirements for time-scale compression and expansion. The methods are focused on the shift determination step of the algorithm since this forms the bulk of the computational load.

The algorithm's robustness as detailed in Chapter 4 indicates that it may be possible to reduce the number of computations while maintaining high quality time-scaled speech. Ideally the algorithm should be "tuned" to operate on speech signals, making full use of inherent redundancies. It is also desirable to exploit the human ear's inability to detect certain errors that can arise in the modified speech.

As mentioned earlier, shift values have increased importance in portions of speech where the signal is stationary and slowly varying, such as vowels. The signal is highly correlated, making errors in alignment more pronounced and easily detected. Those portions of speech which are aperiodic, such as fricatives, generally arise from random noise excitation of the vocal tract. Alignment of these portions is less critical as the signal lacks any distinct periodic structure. Reductions in computation can be accomplished by exploiting these speech signal properties.

An unfortunate consequence of dealing with a subjective measure such as perceived quality is that no easy criteria for evaluation exists. The shifts returned by different alignment functions may differ radically while the speech produced is indistinguishable. This makes an

objective evaluation of the performance of computationally reduced functions difficult. In the following sections numerous functions which reduce the required computations for shift determination are demonstrated. In all cases the resulting speech was compared with that obtained using the standard crosscorrelation technique. Attempts to quantitatively measure the performance are difficult beyond the following measures: "indistinguishable", "slightly worse but barely detectable", "noticeably reduced quality", and "objectionably degraded".

5.2 Predicting Shifts for Time-Scale Expansion

Chapter 3 detailed the source of reverberation and clicks in time-scale expanded signals. It was noted that in many cases the shifts which maximize the crosscorrelation are entirely predictable. Exploiting this predictability allows using lower analysis shifts, S_a, without an accompanying increase in computation. This is possible because the number of predictable shifts also increases. The peak load of the algorithm, however, is not reduced as it may still be necessary to determine shift values via the crosscorrelation for several successive windows.

Observations of the shift values returned by calls to the crosscorrelation procedure showed that in a few cases, approximately 3 to 5 percent of the time, the shifts returned did not correspond to those predicted. For example, when $(S_s - S_a) = 20$ successive shifts should increment by 20 (i.e. 20 40 60 80...). Occasionally a shift would be violate the pattern, only to have the next shift complete the pattern (i.e. 20 57 60 80...). Rarer still, approximately 1 percent of the time, there were occasional instances of violations leading to the initiation of new patterns (i.e. 20 57 77 97..). The cause of such violations in the predictability of the signal is likely due to the fact that windows are added and their sum is used in subsequent correlation evaluations. Because the sum may change the nature of the signal (such is the case when dissimilar windows are added) the shift with the maximum correlation may change. These excursions from the predicted patterns may correspond to alignment with previous pitch periods or alignment with the summed signal. Note when SWO is high, the crosscorrelation will be operating on an output signal composed almost entirely of overlapped previous windows. This signal will in many cases differ significantly from the original input and thus the crosscorrelation may not be maximum when "features" overlap.

Due to the rare instances of prediction rule violations, an implementation which predicts

the shifts according to the rule outlined in Chapter 3 (Equation 3.10) was implemented. The speech obtained from this technique was of high quality and indistinguishable from speech produced without exploiting the predictability. The average computational load is substantially reduced by applying the prediction rule when low analysis shifts are used. The reduction in the number of critical shift determinations has the added benefit of reducing the importance of the correct shift determination for low analysis shifts, as fewer shifts are determined by the alignment function. In all tested cases, predicting the shifts provided high quality output and maintained the local period in the signal.

Prediction essentially allows the algorithm to operate in the *Expand on Demand* fashion where small analysis shifts are used without a significant increase in computations. A straightforward extension to the SOLA algorithm utilizing prediction would calculate the distance to the next window whose shift would not be predictable. Intervening segments would merely be copied to the output buffer. This method resorts to shift determination via crosscorrelation and updates to the output buffer only when shifts are not predictable.

5.3 Average versus Peak Computational Load with Prediction

Recall that shifts were determined to be predictable when the shift which overlays identical segments is contained on the shift interval, or when:

$$\begin{aligned} |K_{n-1}| + (S_s - S_a) &\leq |K_{max}| \\ |K_{n-1}| + S_a(\alpha - 1) &\leq |K_{max}| \end{aligned}$$

We would like to estimate the number of predictable shifts obtained from a nonpredictable shift. To determine this we examine the number of predictable shifts that will occur for each value of a nonpredictable shift. The interval $(0, K_{max})$ is therefore divided into N equally spaced segments starting at the value K_{max}. The number of segments, N, is given by $\lceil \frac{K_{max}}{(S_s - S_a)} \rceil$. The width of the first $N-1$ segments is given by $(S_s - S_a)$, the width of the Nth segment is

given by the fractional remainder of $[\frac{S_s - S_a}{K_{max}}]$.

We assume the value of the shift which maximizes the crosscorrelation is uniformly distributed along the shift interval after several frames. Thus all shifts occur with equal likelihood, and the probability that a shift falls within a specific interval N is (width of N)/K_{max}.

The number of shifts predictable from a previous shift depends solely on the value of the previous shift. For the interval $(K_{max} - (S_s - S_a), K_{max}]$, no shifts are predictable. The number of predictable shifts increases by one with each segment N as the shift moves away from K_{max}. A shift which falls in the Nth segment (segment containing $k(m) = 0$) will result in $K_{max}/(S_s - S_a)$ predictable shifts.

The expected value for the number of predictable shifts from an unpredictable computed shift is approximately:

$$\begin{aligned} E(x) &= \sum_{n=0}^{\frac{K_{max}}{S_s - S_a}} [1 - \frac{n(S_s - S_a)}{K_{max}}] \\ E(x) &= \sum_{n=0}^{\frac{K_{max}}{S_s - S_a}} [1 - \frac{nS_a(\alpha - 1)}{K_{max}}] \\ E(x) &= \frac{K_{max}}{2S_a(\alpha - 1)} \end{aligned} \tag{5.1}$$

Equation 5.1 indicates that the expected number of predictable shifts goes as $1/2S_a$. When utilizing prediction, the fraction of shifts which must be computed is then

$$\begin{aligned} \frac{1}{E(x)+1} &= \frac{1}{\frac{K_{max}}{2S_a(\alpha-1)} + 1} \\ &\approx \begin{cases} 1 & \text{For } \frac{K_{max}}{2S_a(\alpha-1)} \ll 1 \\ \frac{2S_a(\alpha-1)}{K_{max}} & \text{For } \frac{K_{max}}{2S_a(\alpha-1)} \gg 1 \end{cases} \end{aligned} \tag{5.2}$$

Thus the computational cost associated with increasing the number of frames ($1/S_a$ in Equation 2.6) is offset by the decrease in the number of shifts which must actually be calculated when $2S_a(\alpha - 1) \ll K_{max}$.

5.4 Reduced Shift Resolution

The initial implementation of the crosscorrelation procedure computes a crosscorrelation value for each possible shift in the range 0 to K_{max}. Pitch periods take on values between 3.75 msec and 125 msec (30 to 100 samples at a sampling rate of 8 kHz). In general, the sampling rate is not an integer multiple of the pitch period. Thus the number of samples per pitch period will vary by one or two samples. Tests have shown that restricting shifts to n samples, where n takes on values 2 or 3, results in time-scaled speech which is indistinguishable from the higher resolution provided when $n = 1$. Reduced shift resolution effectively subsamples the correlation function. Thus a major peak may be missed if it is extremely narrow. Such narrow peaks are unlikely in the crosscorrelation function of speech however. The reduced resolution will affect alignment of speech with shorter pitch periods the greatest, because a shift of one sample represents a larger relative offset. Lower-pitched speech is less affected since the misalignment is small relative to the pitch period. The coarse alignment of the windows can cause the period of voiced segments in the time-scaled output to jitter by $\approx \pm n/2$ samples. Reducing the resolution also forces misalignment of the higher harmonic frequencies from the pitch pulses.

Several assumptions are made in restricting the resolution of the shift values. The most important assumption is that misalignment of higher frequency harmonics does not substantially reduce the perceived quality. Each increase in n represents a "decimation" of the possible sample alignments. Thus higher frequencies are no longer capable of being aligned. This effect increases into lower frequencies with increasing n, as frequencies above (sampling rate)/$2n$ are aliased lower into the spectrum. Note however that these frequencies will not always be aligned in the standard implementation, since the highest amplitude frequencies dominate the crosscorrelation. The higher amplitude periods in the signal are not always integer multiples of the higher frequencies' shorter periods, nor are these shorter periods integer multiples of the sampling period. (See chapter 3 on output signal averaging during time-scale compression for a discussion of these effects).

The previous assumption is valid if the highest amplitude frequencies in the signal are centered well below the point where this "aliasing" occurs. Thus assuming pitch pulses occur at frequencies below 400 Hz, reducing the shift resolution by 3 means the phase misalignment

between periods at 400 Hz will not exceed 36 degrees. If the pitch pulses occur below 100 Hz, reducing the shift resolution by 3 will give rise to a phase misalignment not greater than 13.5 degrees.

Results

Time-scale modified male and female speech using reduced shift resolution with $n = 2$ was indistinguishable from speech obtained with $n = 1$ over the range of $\alpha = 0.5$ to $\alpha = 2$. Time-scaled speech processed with $n = 3$ was noticeably lower in quality than $n = 1$, yet clearly intelligible. The reduction in quality was most evident in the time-scale expanded by 2 signal.

Reducing the shift resolution beyond every third sample led to a noticeable degradation in quality for time-scale expanded ($\alpha = 2$) female speech. For very high pitched speakers the signal takes on a harsh coarse sound. This effect was not as pronounced for time-scale expanded ($\alpha = 2$) male speech with substantially lower pitch. The quality of time-scale compressed speech ($\alpha = 0.5$) was far less affected by reduced shift resolutions. However all values of $n \geq 4$ gave noticeably degraded time-scale compressed and expanded speech with varying degrees of intelligibility.

5.5 Data Reduction for Correlation

The number of data points used in the correlation calculation may also be substantially reduced without affecting the resulting crosscorrelation indication. Recall that speech signals are composed largely of voiced and unvoiced segments. The voiced segments are largely periodic and relatively stationary over short durations. The unvoiced regions tend to resemble random noise and are aperiodic. Again, due to the length of the periods for voiced speech (30 to 100 samples at 8 kHz sampling rate), the correlation calculation may be computed using decimated data. Decimating the data in the correlation relies on similar assumptions that were made for reducing the shift resolution. We assume that frequencies which are aliased lower into the spectrum will not substantially interfere with the alignment of higher amplitude pitch periods. Decimation occurs in both sequences, so similar aliasing occurs in each, and the crosscorrelation may not be significantly affected.

The amount of decimation however depends on the spectral content of the signal. If an anti-aliasing filter is applied to the input speech signal before the crosscorrelation step, bandlimiting the input signal to 400 Hz, decimation by 10 is possible before aliasing of the 400 Hz sinusoidal signal occurs. Bandlimiting the signal in this case reduces the complex speech waveform to a sinusoid waveform. This sinusoidal signal, sampled at 8 kHz, may be decimated by 10 without aliasing affecting the shift determination. The decimated signal is adequate for representing the overall trends in the sinusoidal waveform. Tests on unfiltered input (bandlimited to 3.8 kHz), however, indicate that limiting the decimation factor to 4 or less maintains a confident indication of alignment. Each decimation corresponds to a folding of each windowed segment's spectrum. Decimation by n of a signal sampled at 8 kHz corresponds to superimposing the signal's spectrum at $\frac{8\text{ kHz}}{2n}$, thus decimation by 4 results in a superimposed (aliased) spectrum at 2000 Hz.

Experiments using the standard shift resolution and different decimation factors for the crosscorrelation indicate no detectable difference in output quality for decimation of the data by 2, 3, or 4. Decimation beyond a factor of 4 was not explored due to the diminishing returns in terms of computational savings and increased susceptibility to noise.

5.6 The Normalized Crosscorrelation Function

The normalized crosscorrelation is designed to provide a measure of the relative correlation between two segments. An unnormalized correlation product varies with the amplitude of the data points. Thus the unnormalized function can give a misleading indication of alignment; i.e., the highest correlation product may not correspond to the most correlated alignment. To correct for the effects of increasing or decreasing amplitudes in the segments to be correlated, the correlation product is divided by the power in the two segments. This removes the amplitude bias that can arise from amplitude variations over several periods and restricts the crosscorrelation function to the interval $[-1, +1]$.

5.6.1 Adaptations to the Normalized Crosscorrelation

The normalized crosscorrelation function used to determine the shift values proves to be computationally expensive. The denominator term accounts for more than two thirds of the total computations involved, and includes a computationally intensive square root operation. Computing these in real-time with a typical hardware implementation proves difficult. Several methods of reducing the computational requirements of the normalized crosscorrelation were explored.

Using an unnormalized crosscorrelation function with a variable number of points provided poor shift values with a noticeable reduction in quality. Calculating the denominator term once and using it for subsequent shift values has no normalizing effect since only the maximum over an interval is selected as the best shift value. Therefore any fixed value for the denominator will be no better than setting it equal to one. One possible method would be to update the denominator periodically over the range of shift values since the signals are assumed to be slowly varying over voiced regions.

5.6.2 Periodic Updates to Denominator

The method of periodic denominator updates proved very effective. By updating the denominator in the normalized crosscorrelation every 15 shift values, the amplitude bias is drastically reduced. The correlation products are divided by a function which is staircase in nature, yet proves effective because the change in power over the shift range of 15 samples in most cases is small. Recalculating the denominator is highly redundant since the same samples are used for calculating the power. The sum of the square of the samples need not be recalculated for each shift. Rather, the value of the previous sum can be stored between shifts and modified appropriately before the square root operation. If the number of overlapping points increases with each shift, the denominator sum from the previous shift is updated by adding the square of the additional samples before the square root operation.

Another alternative to the standard denominator is to use the power in one of the windows rather than both to normalize against changing amplitudes in the signals. Although this method is less justifiable, the results were of such quality that this method could not be dismissed. Either choice of windows is equally justified.

5.6.3 Two Alternatives to Square Root Function

As the square root of an unbounded number is a rather slippery calculation using standard ALU operations, an effective alternative is desirable. Using the product of the sum of the absolute value of samples in the denominator rather than the square root of the product of the sum of the squares of the samples provides equal quality output. Thus:

$$\sum_{j=0}^{L_m-1} |y(mS_s - k(m) + j)| \sum_{j=0}^{L_m-1} |x(mS_a + j)| \tag{5.3}$$

replaces

$$\sqrt{\sum_{j=0}^{L_m-1} y^2(mS_s - k(m) + j) \sum_{j=0}^{L_m-1} x^2(mS_a + j)} \tag{5.4}$$

Another method exploits the simplicity of computing $\log_2$ on binary numbers. A $\log_2$ can be implemented by simply determining the bit position (b-p) of a binary number which contains the most significant 1 (MS1). The rightmost bit-position is defined as position 0, with subsequent bit-positions incrementing to the left. Using the well known identity:

$$\log[\sqrt{x}] = \frac{1}{2}\log x$$

and exploiting the simplicity of computing the $\log_2$ of a binary number we have:

$$\sqrt{x} = 2^{\log_2 \sqrt{x}}$$

where

$$\log_2[\sqrt{x}] = \frac{1}{2}\log_2 x = \frac{1}{2}(\text{bit position of MS1 in binary number})$$

$$2^{\frac{1}{2}\log_2 x} = 2^{(\text{bit position of MS1})/2}$$

Additional precision to the $\log_2$ approximation can be obtained by appending an arbitrary number of the bits immediately to the right of the most significant 1 to the number representing the bit position of the most significant 1.

Recall that the purpose of the square root is to correct for variation in the power in the segments, thus the nearest integer power of 2 should suffice and the bit position approximation

$\sqrt{x}$	x	$\log_2 x$	b-p of MS1	$2^{\text{(b-p of MS1)}}$	$2^{\lceil \text{(b-p of MS1}/2)\rceil}$
4.47	20	4.32	4	16	4
7.07	50	5.64	5	32	8
10.00	100	6.64	6	64	8
22.36	500	8.97	8	256	16
31.62	1,000	9.97	9	512	32
70.71	5,000	12.29	12	4,096	64
100.00	10,000	13.28	13	8,192	128
223.60	50,000	15.61	15	32,768	256

Table 5.1: Comparison of $\log_2$ approximation for square root.

should be adequate. This scheme provides a simple approximation for the square root in the normalized crosscorrelation. The error in this scheme is exponential, as the square root of numbers on the interval $(2^n, 2^{n+1}-1)$ is given by the square root of 2^n. This scheme provided good results for time-scale compression and only slightly lower quality than the standard normalized crosscorrelation for time-scale expansion.

5.6.4 Unnormalized Correlation with Fixed Data Points

Fixing the number of data points used in an unnormalized correlation provided speech that was indistinguishable from the normalized variable length correlation. In this method, only the numerator is used for evaluation of a shift's correlation. Although this function is a crude approximation to the normalized function, the quality of output does not appreciably suffer and the computations and complexity are drastically reduced. This implementation requires careful choice of parameters as the minimum number of overlapping points for all possible shifts must be used. See Section 5.8 for further information on using fixed lengths during shift determination.

5.7 Alternate Functions for Aligning Windows

The previously outlined methods for computational savings reduce the complexity associated with computing a normalized crosscorrelation-like function. The goal of shifting, however, is to obtain the alignment that maximizes signal similarity in terms of magnitude and phase.

Many alternative functions give an indication of similarity between signals. Several of these were tested and evaluated. Unfortunately the gains in simplicity for many cases are not easily realized since current signal processing hardware has been optimized for multiply-accumulate operations.

The following class of functions was explored as an alternative to the computationally intensive normalized crosscorrelation. Several of the functions offer distinct advantages in that a sum or difference need only be normalized by the number of overlapping points used in the calculation. This results in a bounded divide operation (division by a number on a pre-computed bounded interval). Bounded division can be implemented using multiplication by a fraction. A look-up table for the appropriate multiplication value is all that is required. Alternatively the number of points used to compute the sum or difference may be fixed, eliminating the need for any normalization over the range of shift values. The minimum number of overlapping points for all shifts on the shift interval must be used in this case, consequently all available data will not be used for all shifts.

5.7.1 Least Difference

Rationale

If two signals are identical, the difference between them will be zero when they overlap exactly. If the signals are similar in periodic structure, the accumulated absolute difference as a function of lag will be minimum at a point which aligns the periodicities. Such a function requires no normalization and may prove adequate for aligning similar windows of speech. The accumulated difference is given by:

$$D_{xy}^{m}(k) = \sum_{j=0}^{L_m-1} \left| \frac{y(mS_s - k(m) + j) - x(mS_a + j)}{L_m} \right| \tag{5.5}$$

There are several distinct advantages of the least difference function. First no square root is required, and no multiplications are necessary. The function still requires a divide to normalize for the varying amounts of overlapping points used in the sum, but this divide is bounded by the range of overlapping points for shifts on the shift interval. This bounded divide can be implemented using multiplication by fractions. Alternatively, this divide may

be eliminated by fixing the number of points used to the lowest number encountered, in which case all sums may be compared equally.

Results

The least difference method provides good results for male speech, but somewhat lower quality for female speech. The reduced quality arises from stutterant sounding fricatives during short portions of the speech signal; the speech obtained, however, was highly intelligible.

Using a fixed number of points results in extremely poor quality when the number of overlapping points used to compute the difference is less than a pitch period. The time-scale compressed speech obtained using the least difference function was indistinguishable from that obtained using the normalized crosscorrelation function. Time-scale expanded speech varied in quality ,however, with the performance for time-scale expansion of female speech being much worse than male speech. The resulting female speech was noticeably lower in quality than speech obtained from the standard algorithm. The decreased performance for shorter pitch periods is most likely due to the fact that small alignment errors have greater affects on smaller pitch periods.

5.7.2 Same Sign Total

Rationale

After several attempts at further reducing the required computations, it became apparent that an ideal function should give an indication of the alignment/correlation between windows without requiring a denominator term to normalize the function. Alignment affects the quality most during periodic portions of the speech signal. These regions of speech represent voiced segments which have periods between 3.75 and 12.5 msec (30 and 100 samples at 8 kHz sampling rate). If we assume the pitch period is the highest amplitude frequency in these regions, it is valid to assume that the shift which results in the highest number of agreeing signs will also align these periods.

$$R_{xy}^{m}(k) = \sum_{j=0}^{L_m-1} \text{sign}\{y(mS_s - k(m) + j)\}\,\text{sign}\{x(mS_a + j)\} \tag{5.6}$$

This method weights all samples equally which eliminates the need for normalizing the function by the power in the signals. The technique makes full use of the periodic structure of those portions most sensitive to alignment. In essence, the complicated speech waveform is rendered into a square wave of fixed amplitude whose zero crossings match those of the speech signal. The number of agreeing signs is identical to a crosscorrelation on this unity amplitude square wave. The resulting function is therefore a good approximation to the more complicated crosscorrelation yet requires no multiplications.

The key operation performed on the data is an exclusive or (XOR) on the sign bits of the data. As only the sign bits are used, an efficient scenario would involve stripping sign bits from the data as it enters and loading them into a buffer of bit length ($winlen + K_{max}$). A similar buffer would hold the sign bits of the shifted output buffer. The desired shift then corresponds to the bit offset between buffers providing the largest number of 0's (false for XOR) in the XOR result. Many DSP chips perform this type of "population count" of bits on numbers in a single instruction. Note that such an implementation would allow operation on blocks of the input data rather than single samples (8 samples for byte operations, 16 for word operations etc..).

Alternatively the signal could be pre-processed to +1 or -1 for all samples. A single bit multiply-accumulate would correspond to the number of agreeing signs; and assuming less than 256 overlapping points, only 8 bits plus a sign bit would be required for the accumulation sum.

Results

The speech produced by the same sign total technique is indistinguishable from the speech obtained using the standard crosscorrelation function for time scale compression and of slightly lower quality than the standard crosscorrelation function for expansion by a factor of two. The technique however is not nearly as robust as the normalized crosscorrelation to the methods of decimation and reduced shift resolution. Decimating the data used for the sum and reducing the resolution of the shifts resulted in speech of significantly lower quality than comparably decimated crosscorrelation techniques. Note that a shift of one sample in this scheme eliminates two agreeing signs per zero crossing! Thus the robustness of this technique to decimation

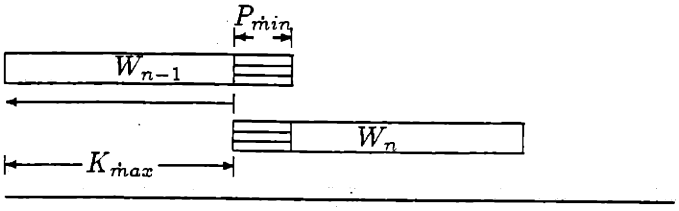


Figure 5-1: Number of data points available for fixed point shift determinations.

depends largely on the number of zero crossings in the signal.

5.8 Fixing the Number of Overlapping Points in Shift Evaluations

Using a fixed number of overlapping data points has several effects that must be noted. In implementations with variable numbers of overlapping points, the value of the alignment function (crosscorrelation, least difference, etc...) obtained reflects the correlation or similarity of all overlapping points.

If the minimum number of overlapping points, P_{min}, across the shift interval is used, the value obtained is not indicative of the similarity of all overlapping points, but rather a fraction of the overlapping points. Only P_{min} points in each window will be used in all alignment evaluations even though substantially more points will be added. The effect of the addition of points other than the P_{min} used in the evaluation, is not taken into account. The measure obtained represents only a fraction of the effect of adding the window if an insufficient number of points are used. It is therefore necessary to raise the number of minimum overlapping points to insure a substantial portion of the points in the window are used to evaluate the alignment for all shifts.

Recalling from Section 3.3.2, the number of overlapping points between the current window, W_n and the rate-modified shifted signal is:

$$\text{overlapping points} = winlen - S_s - k(m-1)$$

As $k(m-1)$ may take on values between $(0,\ K_{max})$ we have:

$$\text{minimum number of overlapping points} = Winlen - S_s - K_{max}$$

Table 5.2 provides an indication of the minimum number of overlapping points available for different parameter combinations. The minimum number of overlapping points increases

α	$Winlen$	S_a	S_s	K_{max}	number of points used in correlation
2	400	100	200	100	100
2	400	050	100	100	200
2	400	020	040	100	260
2	300	050	100	100	100
2	300	020	040	100	160
2	300	010	020	100	180
2	200	035	070	100	030
2	200	020	040	100	060
2	200	010	020	100	080
0.5	400	400	200	100	100
0.5	400	200	100	100	200
0.5	400	100	050	100	250
0.5	400	050	025	100	275
0.5	300	300	150	100	050
0.5	300	200	100	100	100
0.5	300	100	050	100	150
0.5	300	050	025	100	175
0.5	200	200	100	100	000
0.5	200	100	050	100	050
0.5	200	050	025	100	075
0.5	200	020	010	100	090

Table 5.2: Number of points available for fixed point alignment functions.

with increasing window length or decreasing analysis shift. Such changes are limited, however, due to the undesirable effects in terms of reverberation and computation respectively.

Conclusion

The fixed-length methods provide good quality output for many of the alignment functions when the proper choice of parameters is used. The quality of the output obtained for a given

number of points is highly reflective of the robustness of the method employed. Of all alignment functions which do not require a denominator term for normalization, the unnormalized crosscorrelation method from Section 5.6.4 provided the best results using a fixed number of overlapping points in shift evaluations since it was the most robust of methods with similar computational efficiency.

5.9 Update Functions

As the benefit of using low analysis shifts became apparent, a weighting function to provide the same effect was desired. Several update functions were tested in an effort to reduce the required computations and storage for implementation.

5.9.1 Equal Contribution Weighting Versus Time-Order Dependent Weighting

Equal Contribution Weighting

Summing all overlapping samples in the rate-modified shifted signal prior to normalization weights the contribution from all windows equally. This has been the method of choice for several authors [14] [11] [13]. Such an implementation requires a divide and a separate buffer containing identically shifted overlapping window functions to determine the proper scaling factor for a given sample. Maintaining the normalization buffer requires construction of an additional signal, $r(n)$, in parallel with the unnormalized rate-modified-unshifted speech signal, $y(n)$ (See Equation 2.3). This places additional memory and computational burdens on the resources used to implement the algorithm.

The division operation required for equal contribution weighting varies in complexity with the windowing function used. For rectangular windows, the denominator is a small integer, and a lookup table can be used to convert the divide to multiplication by a fraction. When other windowing functions are used, however, a true division will be necessary since the sum of the overlapping shifted window functions can take on numerous values. True division can be costly to implement on some DSP chips.

Conclusion

Equal contribution weighting effectively doubles the storage required to implement the algorithm and increases the number of operations.

Time-Order Dependent Weighting

Averaging each new window with the existing signal during each frame was explored as an alternative to equal contribution weighting. This method eliminates the need for a parallel normalization buffer by incorporating a time or order dependent weighting. In this scheme the weighting function, $f(j)$ may be applied to a window multiple times. If a window overlaps a portion of the output signal constructed by the addition of two previous windows, the weighting function is applied again, thus a recursion of the weighting operation. The operations associated with Equations 2.2, 2.3, and 3 are replaced by

$$y(mS_s - k(m) + j) = \begin{cases} (1 - f(j))y(mS_s - k(m) + j) + f(j)x(mS_a + j) & \text{for } 0 \leq j \leq L_m - 1 \\ x(mS_a + j) & \text{for } L_m \leq j \leq Winlen - 1 \end{cases} \tag{5.7}$$

The function $f(j)$ is a weighting function such that $0 \leq f(j) \leq 1$. Since the weighting function is less than one, each application of the weighting function reduces the contribution of previous windows when multiple windows overlap.

Recursive Addition Updating

The simplest type of time-order dependent weighting is to add the overlapping portion with the existing signal and immediately divide by two. Thus $f(j) = 1/2\ \forall j$. Subsequent overlapping windows are then added to the previously constructed signal and the sum divided by two. Averaging the windows at each frame incorporates a time/order dependent weighting when multiple windows overlap (SWO ≥ 2). The first window is weighted by $\frac{1}{2^n}$ where n is the total number of overlapping windows contributing to a given section of the signal (i.e., the number of applications of $f(j)$ to a region of the output signal). In this case the most recent window added in the region of overlap is always weighted by $\frac{1}{2}$. This technique offers two distinct

advantages: the division by 2 operation can be accomplished by shifting the sum of the output signal and new window one bit to the right, and the need for a separate normalization buffer is eliminated. Such a scheme can only be accomplished when rectangular windowing functions are used for the speech. More complicated windowing functions require keeping track of the overlapped windowing functions.

Results

The speech obtained using recursive weighting with $f(j) = \frac{1}{2}\ \forall j$ was compared with speech obtained using equal contribution weighting. Recursive weighting provided speech that was slightly lower in quality than the original algorithm for time-scale compression ($\alpha = 0.5$) and not noticeably different for time-scale expansion ($\alpha = 2$). The difference in performance is most likely due to the higher number of overlapping windows during compression and the deleterious effects of recursive weighting which occur for higher values of SWO associated with compression.

5.9.2 Conclusion

Recall from Section 3.4.6 that in equal contribution weighting, portions of the unnormalized output are weighted by the sum of overlapping windows. This attenuates the least frequently occurring features in the sum. Recursive windowing, however, attenuates the most frequently weighted window. This effectively removes incomplete "piles" or sections of overlay at the leading edge of multiple feature occurrences in the output signal as desired, but eliminates any attenuation of incomplete piles at the trailing edge of multiple overlay segments. There is no longer a staircase fade between modified and original segments. The last window added will always have a significant contribution to the final output and may or may not contain the features present in previously overlay segments.

Preliminary tests were inconclusive as to which caused greater perceived reverberation in time-scale expanded-by-two signals containing four multiple feature appearances: equal contribution weighting which gives two unattenuated features (due to full piles) occurring between attenuated features (due to incomplete piles), or recursive weighting which results in three essentially unattenuated features. The order dependent weighting scheme that occurs

with recursive windowing has the undesirable effect of discarding the information in the first overlapping windows. In many cases, crucial features in the speech signal may be unduly attenuated. Stop consonants and plosives may be so reduced that they are undetected in the output signal, thus detracting from the perceived quality. This attenuation increases with increasing SWO, making time-scale compression more susceptible to these deleterious effects.

During compression, weighting all windows equally is desired since the contribution from each window is equally important and smooth transitions result. During expansion it is desirable to place added emphasis on the most recently overlapped window and reduce the contribution from previous windows. Previous windows may contain distinct input signal features which cannot be aligned with corresponding features in the current window. This can lead to reverberation and pre-echo when large analysis shifts are used (see Figure 3-10. Reducing the contribution of previous windows can increase the perceived quality when large analysis shifts are used by attenuating pre-echo.

Setting $f(j) = 1\ \forall j$ reportedly provided time-scale expanded speech of high quality [13]. Speech obtained using this technique was highly intelligible but of lower quality for time-scale expansion than equal contribution weighting or recursive weighting with a linear update function. Time-scale compression using this technique was significantly lower in quality than equal contribution weighting or recursive weighting with a linear update function.

5.9.3 Windowing Functions and Weighting

It was noted that a major source of reverberation and reduced quality in the time-scale expanded signal was due to amplitude changes in the input signal at word boundaries. Significant gains in signal quality were noted when triangular windows were used across pitch periods in Time-Domain Harmonic Scaling [4]. Makhoul also reported improvements in signal quality using triangular windows [3]. Each sum of overlapping samples is divided by the sum of overlapping window functions, thus the contribution of a window in the sum depends on its distance from the center of the window in the input signal and its shift relative to the center of neighboring windows. Figure 5-2 illustrates a simple case where two triangular windows overlap. In the region of overlap the signal is divided by the sum of windowing functions. Note the contribution of speech from a window decreases with increasing distance from the

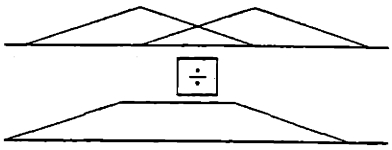


Figure 5-2: Triangular Weighting.

center of its window in the region of overlap. In cases where there is no overlap the speech is normalized by the windowing function and is identical to the original input.

The effectiveness of triangular windowing is greatest during compression when regions of the output signal are comprised of several overlapping windows. During time-scale expansion, however, the overlap between windows is decreased. In this case many portions of the time-scaled signal consist of a single window or overlapped windows with the same interframe interval used during analysis[1]. Thus many regions of the output signal correspond to portions of the input signal exactly. By using triangular windowing functions, the transition to the new window is more gradual because the contribution of the new window is gradually increased in the region of overlap. Linearly increasing the contribution of the leading edge of the new window reduces discontinuities associated with the transition between windows.

It was noted that the speech obtained for time-scale compression using triangular windows was no better than rectangular windows. No distinguishable improvement was obtained using triangular weighting, and in certain cases the quality was worse.

The primary motivation for using a more complicated window was to improve the time-scale expanded signal quality by reducing envelope discontinuities. Triangular weighting does smooth the boundaries between overlapping windows, but as shown in Figure 5-3, the windowing function does not significantly reduce the pre-echoing discontinuities associated with large analysis and synthesis shifts. Note the time-expanded signals obtained using rectangular (corv100.mu) and triangular (cortv100.mu) windows in Figure 5-3 both exhibit the pre-echoing and signal discontinuities associated with too large an analysis shift. The triangular windowing does exhibit smoother amplitude envelope transitions which improve the signal quality

[1]Recall from Sections 3.4.6 and 3.4.7 that small analysis shifts result in regions of overlay. The net result is a replicated portion of signal is identical to input signal after normalization.

slightly. However the presence of substantial pre-echo in both signals (see Figure 5-3) lead to objectionable choppiness.

Summary

Triangular windowing was shown to smooth envelope discontinuities in the time-scale expanded output signal improving perceived quality. However, amplitude envelope discontinuities are apparent only when large analysis shifts ($S_a \geq 10$ msec) are used during time-scale expansion. These effects are not detectable at lower analysis shifts. The reduction in signal quality using analysis shifts greater than 10 msec (80 samples) is caused primarily by pre-echoing which is not reduced when triangular windows are used. Triangular windowing carries the added burden of maintaining a parallel buffer of overlapping windowing functions that must be used to scale the rate-modified shifted signal to obtain the final output. Because of the increased complexity associated with triangular windowing and the relatively small gains achieved during time-scale expansion, this process was abandoned for real-time hardware implementations. I conclude that this scheme, with its added complexity, offers no improvement over simply decreasing the analysis shift.

5.9.4 Linear Recursive Weighting

It has been shown that discontinuities can arise at the boundary of overlay and modified regions in the rate-modified shifted signal because no smoothing occurs when SWO is low. Triangular windowing does provide a more gradual transition in the window on the right-hand side of the boundary when multiple windows overlap. Indeed this was the primary motivation for pursuing a more complicated weighting scheme! As an alternative to triangular windowing, recursive linear weighting was explored.

Linear recursive weighting eliminates the need for a parallel normalization buffer by linearly weighting the overlapping portions before adding them. Linear recursive weighting uses the following weighting function in Equation 5.7 to update the output signal:

$$f(j) = \frac{j}{L_m} \tag{5.8}$$

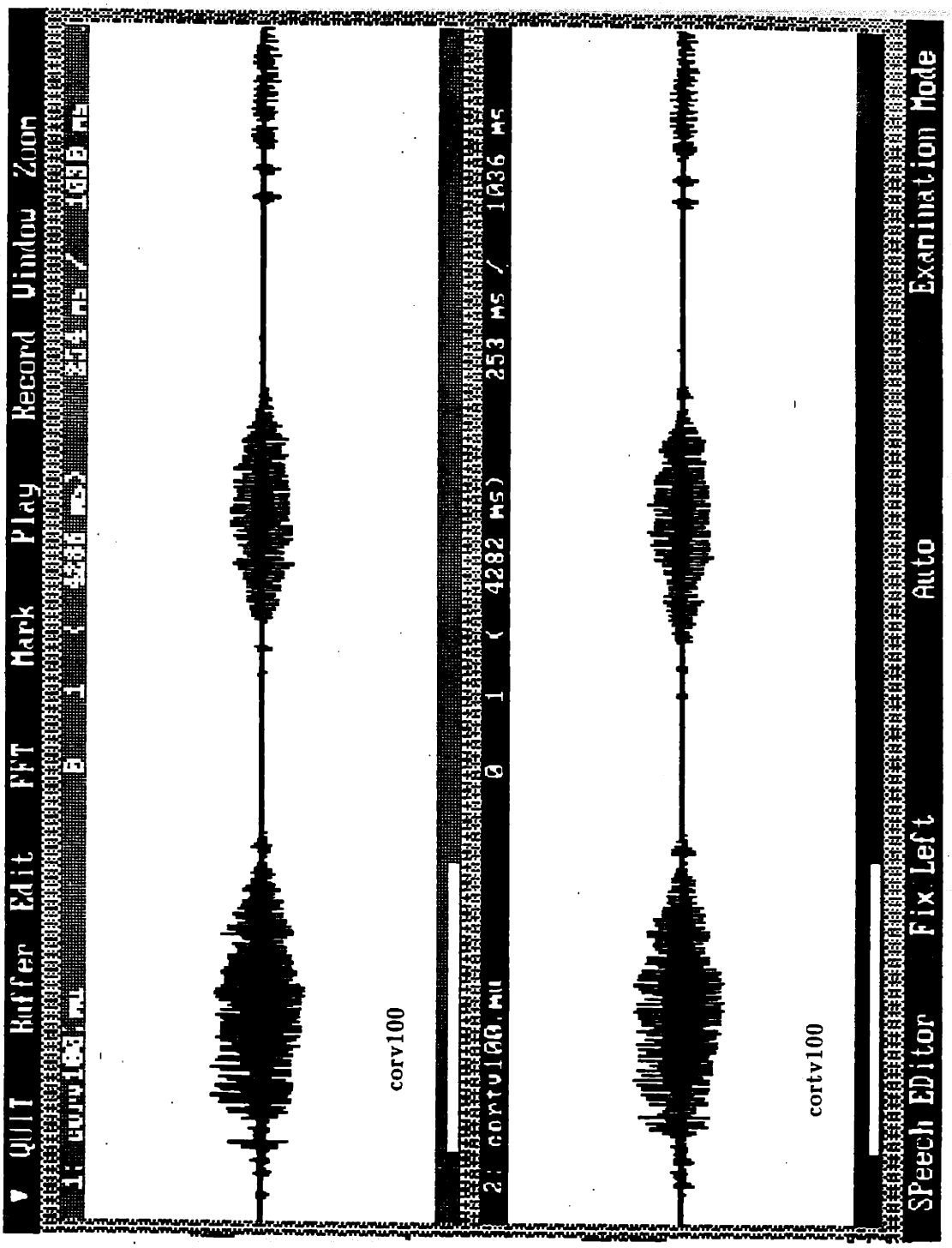


Figure 5-3: Time-scale expanded-by-two speech using triangular windowing (cortv100) and rectangular windowing (corv100). Both signals processed using *winlen* = 40 msec, S_a = 12.5 msec, S_w = 25 msec, K_{max} = 12.5 msec.

The quantity L_m represents the number of overlapping points between the windows and is bounded on a finite interval. A look-up table may be used to obtain $f(j)$ for all possible j and L_m, thus no division is required. As before, this scheme results in a time/order weighting of the data in overlapping windows.

Although the number of overlapping points will vary from window to window, the total number of overlapping points cannot exceed a window length. A look up table containing appropriate scale factors for a window length's worth of overlapping points may be used when fewer points overlap by skipping appropriate entries. Although this leads to added complexity, the gain in quality justifies the added computations for very high quality applications. In Chapter 6, methods which fix the overlap region are introduced, further simplifying implementation of this update technique.

Results

Recursive linear weighting was compared with equal contribution weighting. No significant improvement was noted for speed up ($\alpha = 0.5$) using linear recursive weighting when compared with equal contribution weighting. A slight lowering in the spurious noise floor was barely detectable. This is likely due to the decreased influence of data near the window edges.

Recursive linear weighting improved the output signal quality for expansion ($\alpha = 2$, $\alpha = 1.5$) such that the analysis shift could be doubled with the resulting output remaining equal or superior in quality to the standard equal contribution weighting scheme. Recursive linear weighting did not, however, prevent pre-echoing associated with large analysis shifts ($S_a \geq 10$ msec) used during time-scale expansion.

Note that utilizing prediction allows a decrease in the analysis shift with only a small increase in the average computational load from the overhead associated with each frame. This reduces the emphasis on using large analysis shifts to save computations. Linear recursive weighting, however, offers the added benefit of reducing peak load and overhead, since fewer frames are used over a given section of the input.

5.10 Hardware, Computation, and Implementation Issues

The tradeoffs for implementing the alignment function must be fully investigated for determining the optimal implementation for real-time hardware. Specifically, issues such as overflow, underflow, data access, and memory maintenance drastically affect the number of operations required for implementation.

The number of operations necessary to implement the algorithm is extremely architecture dependent. To illustrate this dependency, the computational requirements of three shift-determination methods are compared in terms of the different operations required. To determine the true cost of each implementation, the following measures are defined:

- α: The number of shift calculations.
- β: The number of operations associated with computing the numerator of the crosscorrelation (multiplying the signals $x(n)$ and $y(n-k)$).
- γ: The number of operations associated with computing the denominator (i.e., power in $x(n)$ by squaring and summing the samples).
- δ: The number of operations associated with the division operation (δ_b and δ_u refer to bounded and unbounded division respectively).
- ϵ: The frequency of the denominator calculation.
- ε: The operations required to compare the current shift's alignment function value with the existing maximum alignment function value on the shift interval (i.e., picking the peak).
- ζ: The operations required to load data from memory into the internal registers.

Using the above representations, the approximate number of operations for the standard normalized crosscorrelation exceeds:

$$\text{Cost} \approx \alpha[\zeta(\beta + 2\epsilon\gamma) + \text{ops required for } sqrt() + \varepsilon + \delta_u]$$

A modified version of the algorithm, designated *cross33up1trm* for the function it calls, uses data decimation by 3, a shift resolution of 3, periodic updates to the denominator function,

and the power in the new window only for the denominator calculation. The approximate number of operations for the *cross33up1trm* version exceeds:

$$\text{Cost} \approx \frac{\alpha}{3}\left[\zeta(\frac{\beta}{3} + \frac{\epsilon}{5}\frac{\gamma}{3}) + \frac{\varepsilon}{3} + \delta_u\right]$$

The *crosssign* version of the SOLA algorithm counts the number of agreeing signs in the region of overlap and normalizes this quantity by the number of overlapping samples. The approximate number of operations for the *crosssign* version exceeds:

$$\text{Cost} \approx \alpha\left[\frac{1}{16}(\zeta + \text{operations for 16-bit XOR}) + \frac{\varepsilon}{2} + \delta_b\right]$$

This estimate reflects implementation of the *crosssign* version on an architecture which performs 16-bit XOR in a single instruction cycle (typical of most DSP microprocessors) and an implementation in which only the sign bits of samples are used in shift determinations. The sign bits are "packed" into 16-bit integers for the XOR operation which reduces the operations required to load and operate on the data by 1/16. The operations required to compare values of the alignment function are reduced by 1/2 since the number of agreeing signs cannot exceed 256 (2^8) for overlapping segments of less than 256 samples. This means only single bytes need be compared.

5.10.1 Integer and Fixed Precision Implementations

The SOLA algorithm is well suited for integer implementation on current DSP chips. The crosscorrelation represents the most difficult operation to implement. The large unbounded nature of the sum in the multiply-accumulate used to evaluate the correlation proves the most cumbersome! A 16-bit linear representation for the speech signal results in 32-bit intermediate product terms during the crosscorrelation computation. Provided the number of product terms (i.e. number of overlapping data points) is less than 256, the accumulated sum of each correlation will not exceed a number easily represented by 40-bits.

Precision becomes an issue only after the divide associated with the normalization step. This issue can be easily side-stepped using one of the various correlation-indicating functions that do not require unbounded division. Small errors due to rounding or truncation of the

16-bit speech data are not detected by the human ear.

5.10.2 Storage Requirement

The storage requirement for implementing the SOLA algorithm can be reduced to two buffers. One contains the new window of data to be added to the output signal, the other contains the most recently constructed portion of the output signal.

The buffer containing the new window to be added is used in the correlation calculation and then added to the output buffer with the appropriate shift. The new-window buffer requires a fixed length equal to the window length used and is overwritten with new data each frame.

The output signal buffer is used for both the correlation computation and the update procedure. Since the position of the window to be added is bounded by the shift interval, only a portion of the newly constructed output signal needs to be stored. The output signal buffer requires a fixed length equal to $(winlen - S_s + K_{max})$. To allow time-scale expansion and compression over a wide range of synthesis shifts (S_s) using fixed hardware resources, a length of $(winlen + K_{max})$ should be used.

After determining the proper shift for the current window, the existing signal must be updated. Construction of the output signal using one of the previously outlined recursive weighting update functions can be accomplished using a circular buffer with $(winlen - S_s + K_{max})$ data elements. The output signal is generated as follows:

1. The output buffer is initialized with the first frame of input. The initialization is accomplished by copying the first window of data clockwise into the output buffer starting at the 0 or 12 o'clock position.

2. The correlation of the new-window buffer with the output buffer is evaluated for each shift on the interval $[0, K_{max}]$.

3. Add the new window to the output buffer with the shift that maximizes the correlation between buffers.

4. S_s data points from the output buffer are output as finished speech. The data points

are output from $(Target(m-1) - K_{max})$ to $(Target(m) - K_{max})$. This is final output since it will be unaffected by the addition of subsequent windows.

5. Zero the output buffer from $(Target(m-1) - K_{max})$ to $(Target(m) - K_{max})$.

6. Loop back to step 2 using the next window of data in the new-window buffer.

This implementation offers two principal advantages: fixed memory requirements and fixed worst-case computational loads. Assuming 16-bit data is used, the storage required will be less than 1250 bytes. The computations required will be less than 100 operations per output sample, on the order of 50 operations per output sample for shift determination and 50 operations per output sample for data transfer and memory management.

5.11 Summary

In Chapter 5 methods which reduce the computational requirements of the SOLA algorithm were examined. Numerous update equations were tested in an attempt to increase the analysis shift and thus reduce the required number of computations. None however afforded the ability of exceeding an analysis shift of approximately 8.75 msec (70 samples) for expansion without a noticeable decrease in quality. This hard limit is imposed by the mechanisms of the SOLA algorithm for time-scale expansion.

The combined savings in computation using decimated data, reduced shift resolution, and modified alignment functions allowed substantial reductions (by more than a factor of 50) in computational requirements. During the more computationally intensive time-scale expansion, exploiting prediction in the algorithm allowed small analysis shifts to be used without increasing the average computational load. Predicting shifts reduced the average computational load by approximately a factor of 4. The total reduction in computation amounts to a peak load reduction of $\frac{1}{50}$ for compression and expansion, and an average load reduction of $\frac{1}{200}$ for expansion. These reductions do not significantly reduce the perceived quality for time-scale modified speech signals.

The number of computations required for shift determination exploiting these reductions is approximately 40 to 50 operations per output sample. The total required operations will vary with the implementation and update function used, as well as the quality desired.

Chapter 6

SOLA-b: A Modified SOLA Algorithm

6.1 Introduction

The previous chapter was dedicated to modifications of the SOLA algorithm which would reduce computation and aid in implementation. Substantial reduction in computational load was shown to be possible without an associated loss of quality or robustness provided by the original SOLA technique. Implementing the SOLA algorithm for operation on real-time signal processing architectures exposes some key obstacles which can easily be side-stepped with minor modifications to the original algorithm.

Recall that each new window is displaced from its target position in the output signal. It's final position may lie anywhere within the shift interval. Thus the synthesis interframe shift varies with each frame. The variable synthesis shift during the construction of the output buffer gives rise to a signal whose growth rate varies frame to frame. The fixed distance between target positions of new windows leads to a variable number of data points available for each crosscorrelation calculation and variable length overlapping regions.

The variable length overlap region requires the weighting function, $f(j)$ in Equation 5.7, be defined over a variable length interval. This complicates use of linear update functions, in which the values of $f(j)$ are dependent on the total length of the overlap interval. The variability in the number of data points available for the crosscorrelation and update functions

requires knowledge of the previous shift value to determine the current number of data points in the output signal buffer. This increases storage and computational requirements since the number of data points must be calculated and stored between frames. Additionally, the computations and time required to generate a given number of output samples will vary, requiring that the worst-case latency be used.

An implementation in which the region of overlap is fixed significantly reduces complexity. All values of the weighting function, $f(j)$, are fixed. The computational requirements and output signal generation time are fixed. Such gains are easily achieved by varying the analysis interframe shift and fixing the synthesis interframe shift. This new algorithm, a variation of the original SOLA, will be called the SOLA-b algorithm.

6.2 SOLA-b Definition

The goal of the "Synchronized-Overlap Add" technique was to maximize the similarity between overlapping portions during construction of the output signal. This goal was achieved by fixing the analysis shift and varying the synthesis shift. The goal can also be realized by varying the analysis shift along the input signal while fixing the synthesis shift. In this implementation the first p samples in each new window overlap the last p samples in the output signal. The starting position of each analysis window is allowed to vary dynamically so the crosscorrelation of the first p points in the new window with the last p points in the output signal is maximized.

The time-scale modified signal, $y(n)$, obtained using the SOLA-b technique for aligning input signal segments, $x(n)$, is given by:

1. Initializing the signals $y(n)$.

$$y(n) = x(n) \text{ for } n = 0 \ldots Winlen - 1 \tag{6.1}$$

2. Updating $y(n)$ by each new frame of the input signal, $x(n)$, as follows:

$$y(mS_s + j) = \begin{cases} (1 - f(j))y(mS_s + j) + f(j)x(mS_a + k(m) + j) & \text{for } 0 \leq j \leq P - 1 \\ x(mS_a + k(m) + j) & \text{for } P \leq j \leq Winlen - 1 \end{cases} \tag{6.2}$$

$$k(m) = \max R_{xy}^m(k) \tag{6.3}$$

$$\begin{aligned} R_{xy}^m(k) &= \frac{S\left[\sum_{j=0}^{L_m-1} y(mS_s + j)x(mS_a + k + j)\right]^2}{\left[\sum_{j=0}^{L_m-1} x^2(mS_a + k + j)\right]} \\ S &= \text{sgn}\left[\sum_{j=0}^{L_m-1} y(mS_s + j)x(mS_a + k + j)\right] \\ L_m &= P \ \forall m \end{aligned} \tag{6.4}$$

This technique offers several benefits. At each frame the output signal is extended a fixed amount, $winlen - p$. Thus the region of overlap is constant, the number data points in the crosscorrelation is fixed for all shifts. As previously mentioned this allows fixing the values of the update function, $f(j)$, so they may be pre-computed and accessed when needed. In addition, the same data points in the output signal (the last p data points) are used in each crosscorrelation evaluation, thus only the first p data points of the sliding analysis window change in the correlation calculation between shifts. This leads to the simplified normalized

crosscorrelation. The original normalized crosscorrelation

$$
\begin{aligned}
R_{xy}^{m}(k) &= \frac{\sum_{j=0}^{L_m-1} y(mS_s+j)x(mS_a+k+j)}{\sqrt{\left[\sum_{j=0}^{L_m-1} y^2(mS_s+j)\right]\left[\sum_{j=0}^{L_m-1} x^2(mS_a+k+j)\right]}} \\
Ola_{min} &\leq L_m \leq winlen
\end{aligned}
\tag{6.5}
$$

can be simplified to:

$$
\begin{aligned}
R_{xy}^{m}(k) &= \frac{\sum_{j=0}^{L_m-1} y(mS_s+j)x(mS_a+k+j)}{\sqrt{\left[\sum_{j=0}^{L_m-1} x^2(mS_a+k+j)\right]}} \\
L_m &= winlen - S_s = P \ \ \forall m
\end{aligned}
\tag{6.6}
$$

The quantity $\left[\sum_{j=0}^{L_m-1} y^2(mS_s+j)\right]$ in Equation 6.5 remains constant duringthe shift determination of the current window. Since only the maximum of the normalized crosscorrelation is desired during each shift determination, this constant can be removed with no effect on the location of the maximum, only its value. This reduces the number of multiply-accumulate operations by 1/3.

This crosscorrelation used in the SOLA-b algorithm may be simplified further. The $\sqrt{\cdots}$ operation may be eliminated by squaring the numerator and denominator to give:

$$
\begin{aligned}
R_{xy}^{m}(k) &= \frac{S\left[\sum_{j=0}^{L_m-1} y(mS_s+j)x(mS_a+k+j)\right]^2}{\left[\sum_{j=0}^{L_m-1} x^2(mS_a+k+j)\right]} \\
S &= \text{sgn}\left[\sum_{j=0}^{L_m-1} y(mS_s+j)x(mS_a+k+j)\right] \\
L_m &= winlen - S_s = P \ \ \forall m
\end{aligned}
\tag{6.7}
$$

Note that this simplified crosscorrelation provides identical results since, for our purposes,

it is mathematically equivalent to the standard normalized crosscorrelation. Again, the value L_m is fixed in this new implementation.

Also note that for the proper choice of parameters, no region of overlap will contain a previous region of overlap. In this case the crosscorrelations and shift determination can be computed using only the input signal. The last p data points at the end of every window will be identical to the last p data points of the output signal at every step of the output signal construction. Thus we can compute the location in the input signal corresponding to the start of the last p samples in a window, W_m. Note this location depends on the choice of S_a and the shift $k(m)$ associated with W_m. The location corresponding to the start of the next window, W_{m+1}, is then computed. Note that the starting location of W_{m+1} depends solely on the value of S_a. Thus a buffer of input signal samples of length $K_{max} + 2P$ is adequate for determining the shift of a window.

6.3 Comparison of SOLA-b and SOLA

6.3.1 Tests Performed

Performance of the SOLA-b technique was compared to the SOLA for time-scale compression and expansion of male and female speech.

Time-Scale Compression

Male and female speech were time-scale compressed via the standard SOLA algorithm using a window length of 32 msec (256 samples), analysis shift of 32 msec (256 samples), synthesis shift of 16 msec (128 samples), and a maximum shift value of 12.5 msec (100 samples). The output obtained was compared with the same male and female speech obtained using the SOLA-b method. Identical values for the parameter set were used for the SOLA-b method. The speech obtained from the two differing methods was indistinguishable for both the male and female speech.

To test performance at moderate compression ($\alpha \approx 1$), male and female speech was processed via the SOLA and SOLA-b algorithms using identical parameter sets. The parameter set used in this test was specified by a window length of 32 msec (256 samples), analysis shift

of 32 msec (256 samples), synthesis shift of 25 msec (200 samples), and a maximum shift value of 12.5 msec (100 samples). No difference was detected in the speech obtained using the differing methods.

Time-Scale Expansion

Male and female speech was processed via the SOLA and SOLA-b algorithms using identical parameter sets. For the parameter set specified by a window length of 32 msec (256 samples), analysis shift of 12.5 msec (100 samples), synthesis shift of 25 msec (200 samples), and a maximum shift value of 12.5 msec (100 samples), the SOLA-b algorithm produced speech that was superior in quality to speech using the SOLA technique. The speech however was not of high quality due to the ill conditioned parameter set for the SOLA algorithm which effectively reduces the shift interval by violating Equation 3.4.

Performances of the two techniques were compared using other parameter sets: a window length of 32 msec (256 samples), analysis shift of 5 msec (40 samples), synthesis shift of 10 msec (80 samples), and a maximum shift value of 12.5 msec (100 samples). For this parameter set, there was no distinguishable difference in quality between the two methods.

6.3.2 Performance: Quality and Computation

It is important to note that the computational requirements are not equal in both algorithms. Namely the fixed number of overlapping points in the SOLA-b algorithm requires less cross-correlation computations than the corresponding SOLA algorithm. This is due to the fact that shifting the synthesis windows backward along the shift interval in the SOLA algorithm increases the number of overlapping points between windows. The SOLA-b technique of shifting the analysis windows results in the same number of overlapping points for all shifts along the shift interval.

The SOLA-b algorithm is free of many restrictions associated with the SOLA algorithm. SOLA-b is not constrained by Equation 3.4 for high-quality operation. Using SOLA, the window length parameter is restricted to be larger than $K_{max} + Ola_{min} + S_s$ (see Section 3.3.2, equations 3.5, 3.6, 3.7), since the growth of the output signal varies by up to K_{max} for each frame. The larger window length required due to parameter set interactions leads to larger

overlap regions than necessary for the crosscorrelation evaluations. In the SOLA-b technique the signal is extended a fixed amount with each frame, and the number of overlapping points is determined by the minimum necessary for the crosscorrelation computation. Furthermore, since only a minimum number of overlapping sample points (Ola_{min}) are required in the crosscorrelation calculation (Section 3.3.2), only Ola_{min} data points need be evaluated during all shifts along the shift interval. This results in substantial computational savings since SOLA requires at least $Ola_{min} + K_{max}$ overlapping points when evaluating the left-most shift along the shift interval.

The computational requirements for SOLA are given by:

$$\begin{aligned} &\propto \left[\frac{\#\text{ frames}}{\text{output sample}}\right]\left\{\left[\frac{\#\text{ shift evaluations}}{\text{frame}}\right]\left[\frac{\text{avg. }\#\text{ computations}}{\text{shifteval}}\right] + [\text{update comps.}]\right\} \\ &\propto \left[\frac{1}{\alpha S_a}\right]\left\{[K_{max}]\left[\text{MOPOSI} + \frac{K_{max}}{2}\right][\text{IPC}] + [\text{MOPOSI}][\text{update computations}]\right\} \end{aligned} \tag{6.8}$$

$$\text{MOPOSI} = winlen - S_s \geq Ola_{min} + K_{max} \tag{6.9}$$

MOPOSI must be greater than $Ola_{min} + K_{max}$ to prevent a gap in the output stream when the previous window is maximally shifted. This is a direct consequence from Equation 3.5. Equation 6.9 sets a lower bound on the window length for the SOLA algorithm: $winlen \geq Ola_{min} + K_{max} + S_s$. Note that for SOLA-b, the sole requirement on window length is that it be Ola_{min} points greater than S_s, which results in a smaller number of overlapping points for most parameter sets. Thus the computation requirements for SOLA-b become:

$$\begin{aligned} &\propto \left[\frac{1}{\alpha S_a}\right]\{[K_{max}][winlen - S_s][\text{IPC}] + Ola_{min}[\text{update computations}]\} \\ &\propto \left[\frac{1}{\alpha S_a}\right]\{[K_{max}][Ola_{min}][\text{IPC}] + Ola_{min}[\text{update computations}]\} \end{aligned} \tag{6.10}$$

Using Equation 6.8 and Equation 6.10 for the SOLA and SOLA-b algorithms respectively, with comparable parameters (more specifically, identical frame rates), we can compare the

performance of both algorithms in terms of their computational requirements: IPC/frame (inner product computations per frame) and Update/frames (updates per frame).

From the equations above it is clear that for a given frame rate, the SOLA algorithm requires K_{max} more minimum overlapping points than SOLA-b. Furthermore, the number of points in each crosscorrelation evaluation increases as the window is shifted. Because the number of additional points is fixed for a given frame rate, the computational overhead can be calculated exactly.

Using Equations 6.8 and 6.10 with $K_{max} = 100$ and $Ola_{min} = 30$ and noting that the IPC for SOLA-b are 2/3 the IPC for SOLA using the modified crosscorrelation, SOLA-b requires 1/9 the computations for shift determination. The advantage of the SOLA-b algorithm increases with smaller values of Ola_{min}. Values as low as 20 overlapping sample points have been used without degradation of the output signal, at which point the SOLA-b requires 1/13 the number of IPC computations in the SOLA algorithm.

Additionally, SOLA-b requires fewer update computations per frame. SOLA-b updates only Ola_{min} points per frame and copies S_s to the output while the SOLA technique updates $K_{max} + Ola_{min}$ points on average and copies S_s.

To test the effect of shortened window lengths on performance of the SOLA-b algorithm the following parameter sets were compared.

$winlen = 256$		$winlen = 110$
$S_a = 40$		$S_a = 40$
$S_s = 80$	versus	$S_s = 80$
$K_{max} = 100$		$K_{max} = 100$

Note that the number of frames is the same in each parameter set, but the number of overlapping points and thus crosscorrelation computations has been significantly reduced using a shorter window length. The output obtained using the shorter window length was of slightly better quality than the output using the more computationally intensive window length. This pleasant result most likely results from the decreased amount of averaging that occurs when shorter window lengths are used.

Overlap-Replace Update Function Using SOLA-b

The high quality obtained using small regions of overlap ($= Ola_{min}$) in the SOLA-b algorithm prompted experiments using an overlap replace rather than update scheme. This scheme, proposed by Wayman, Reinke, and Wilson [WRW89], overwrites the overlapping region of the signal with the new window. The performance of this technique for time-scale compression was poor. Relatively large overwrite regions lead to bad splices as the signal changes significantly across window boundaries. The performance of this technique for time-scale expansion was significantly lower than the overlap-add using SOLA-b for the two parameter sets used:

$$\begin{array}{rcl} winlen = 256 & & winlen = 110 \\ S_a = 40 & & S_a = 40 \\ S_s = 80 & \text{and} & S_s = 80 \\ K_{max} = 100 & & K_{max} = 100 \end{array}$$

6.4 Summary

This chapter introduced a modified version of the SOLA algorithm, SOLA-b, which greatly simplifies implementation. It provides time-scale modified speech of equal quality with significantly reduced computational requirements. The SOLA-b algorithm operates most efficiently when $winlen - S_s = Ola_{min}$. In this case the SOLA-b update resembles splicing more than averaging, since most new data points are appended to the output buffer rather than averaged.

Chapter 7

Conclusions and Topics for Future Investigations

This Chapter presents conclusions based on the body of work presented in this thesis. No new information is presented in the conclusion section of this chapter, rather it serves to highlight important findings. Several topic for future investigation and future applications are presented in the "Future Investigations" section.

7.1 Conclusions

7.1.1 Parameter Selection for SOLA Algorithm

Listen tests (See Appendix A) indicate that 25 msec window lengths were preferred for female speech, while 37.5 msec window lengths were preferred for male speech. A window length of 32.0 msec (256 samples) is compatible with both male and female speech with no loss in quality.

Time-Scale Compression

An analysis shift equal to the window length should be used. This choice eliminates reverberation in the output signal by insuring no segment of input appears in more than one window.

Time-Scale Expansion

An analysis shift of not more than 6.25 msec should be used in conjunction with shift prediction to reduce computation.

Time-Scale Compression with Subsequent Expansion for Data Reduction

The analysis shift used for compression should be less than that for normal expansion due to the reduced stationarity of the signal. If possible, the window length used should contain between two and three pitch periods.

Alignment Function

The quality of speech obtained by decimating data by a factor of three and reducing the shift resolution by 2 provided speech which was indistinguishable from the standard technique with one-sixth the computations. This technique is highly recommended.

7.1.2 Computational Requirements

The modified versions of the SOLA algorithm and the SOLA-b algorithm can easily process telephony quality speech (bandlimited to 4 kHz and sampled at 8kHz) in real-time using currently available signal processing chips. For signals sampled at higher rates, increased processing speed may required parallel processing for real-time applications.

7.1.3 Intelligibility Enhancement

Ad hoc experiments performed by time-scale expanding rapidly articulated speech which was difficult to understand indicated no increase in intelligibility. If a segment of speech is difficult to understand as a result of slurring or careless articulation, time-scale expansion will not improve its intelligibility.

7.1.4 SOLA Performance on Broad-Band Speech

The SOLA algorithm was used to time-scale compress speech sampled at 44.2 kHz. The resulting speech was of high quality and did not expose any abnormalities in the output that

might have been masked by the operation on telephony-quality speech.

7.1.5 The Winning Algorithm

From the experiments conducted during this thesis, the SOLA-b algorithm provided the best quality:computation ratio of the standard and reduced computation versions of the SOLA algorithm. The SOLA-b algorithm operates most efficiently when the number of overlapping points is small. The linear weighting update function significantly improves quality when large analysis shifts are used. The benefits of linear updating are less apparent when small analysis shifts are used.

7.2 Future Investigations

7.2.1 Introduction

This section introduces topics for further investigation. The SOLA algorithm has been shown to provide high-quality time-scale modified speech with relatively low computational requirements. This thesis explored several techniques for increasing the quality to computation ratio, and revealed parameter interactions that account for reverberation in the output signal.

7.2.2 Increased Intelligibility of Time-Scale Modified Signals

To develop more natural sounding time-scale modification, techniques which preserve the time-scale of critical phonemes while modifying others may be explored. Presumably additional pre-processing of the speech signal may be useful in detecting critical phoneme segments of the signal. The crosscorrelation provides information about the local pitch and may be useful in making voiced/unvoiced decisions.

7.2.3 Parallel Processing for Shift Determination

The task of determining the shift value which maximizes the crosscorrelation of a particular window lends itself to parallel processing techniques. The crosscorrelations may be computed in parallel to reduce latency and increase throughput in the algorithm. The parameter set may be chosen such that the shifts of each window can be determined from the input signal.

7.2.4 Input Signal versus Rate-Modified Signal Operations

One specific drawback of the SOLA algorithm is its inherent operation on the rate-modified unshifted signal rather than the input signal for the shift determinations. This prevents pre-computing appropriate shift values for the construction of the output signal. The TDHS algorithm forwarded by Malah [4] offers just such an advantage, as the pitch is estimated in a pre-processing pass on the input signal and used to perform insertion or deletions as required. For a limited range of parameter sets, the shift values in the SOLA and SOLA-b algorithms can be determined from the input signal alone. The range of time-scale modifications is limited however.

In interactive applications it is desirable to perform time-scale modification at different rates during various portions of the speech signal. A hybrid algorithm combining pitch information from the input signal with the displacement of windows in the rate-modified unshifted signal would allow predicting the shift which would maximize the correlation with previous windows. During the less critical random noise sections of the signal the time-scale could be preserved, or windows could be added without shifting.

During voiced portions of the signal, the shift that maximizes the correlation between the current and previous window is given by the sum of the fractional period at the end of the previous window and the fractional period at the beginning of the current window modulo the pitch period.

Performing time-scale modification via the SOLA algorithm provides information in the from of shift values which indicate the distance between similar portions of the speech signal. It may be possible to utilize information obtained from a computing the shifts at low analysis resolutions to infer the shifts that would be appropriate for construction of the modified signal using different synthesis shifts. Pre-computing all shift values at a low analysis shift and adding pairs of shift values does not result in the same shift obtained by doubling S_a because the offset between consecutive windows in the rate-modified unshifted signal is increases by S_s.

7.2.5 Pitch Synchronous or Hybrid TDHS algorithm

Time-Domain Harmonic Scaling makes use of the local pitch period to determine the amount of shift between frames. This technique is advantageous in time-scale expansion of speech

with long pitch periods. With knowledge of the local pitch period, the algorithm can advance an integer multiple of the local period before updating the current pitch estimate. The SOLA algorithm advances a fixed amount regardless of the local pitch. Utilizing pitch information to determine the analysis shift may provide an algorithm with improved performance.

7.2.6 Speech Boundary Detector

Observations of the crosscorrelation across the frame sequence led to the several possible applications of similar functions. Since the crosscorrelation provides an accurate indication of the similarity between signals irrespective of their amplitudes, the correlation of a single unit with neighboring units could be used to give an indication of the duration of a single unit type in the speech signal.

7.2.7 Speech Redundancy Reduction

One interesting application of time-scale compression for coding and speech processing involves reducing redundancies in the speech signal via SOLA time-scale compression. The normalized crosscorrelation function gives a quantitative evaluation of signal similarity. If consecutive frames above a certain correlation value were deleted from the signal many of the redundancies could be eliminated. The goal of such processing is to obtain a series of individual pitch periods, each differing from its neighbors by some threshold value. The number of redundant periods removed could be stored and used later to construct a time-expanded signal that resembled the original. The reconstructed signal would undoubtedly suffer in quality, but possibly remain intelligible.

7.2.8 Intelligibility Enhancement

Speech recognition efforts continue to improve the ability to detect important features in a continuous speech signal. Accurate knowledge of specific phoneme boundaries could be combined with SOLA time-scale modification to produce more naturally sounding time-scale modified speech and possibly increase intelligibility. Preserving critical portions of the speech signal while time-scale modifying others would provide a powerful means for enhancing intelligibility.

7.2.9 Data Compression

Several attempts have been made to reduce the data required for speech coding [14] [3]. The limiting factor in compression is the duration of the shortest phoneme. It may be possible to increase the quality by varying the amount of compression/expansion with the signal. Long stationary segments in a speech signal are prime candidates for compression. Short rapidly changing segments, however, are difficult to compress and re-expand without significant alteration.

7.2.10 Voice Mail Applications

Time-scale modification of speech is particularly attractive in voice messaging systems. A high-quality time-scale modification feature would allow users to speed through long messages without missing information. Users familiar with the messaging system could speed-up prompts from the voice messaging system to save time. Message segments containing important information, such as names, addresses, or phone numbers, could be time-scale expanded to aid in transcription.

Issues

Several issues surround time-scale modification in voice-messaging. The first and foremost is whether such a feature would be utilized. Although the benefits of such a feature are clear, several other issues play significant roles in its utilization. The user interface should allow simple and seamless transitions between time-scale modified and normal speech without pauses or deletions. Also, to be useful, the quality of the time-scale modified speech should be such that users would not have to replay the message at normal speed to be assured of its content.

Merging Time-Scale Modification with Data Compression-Decompression

By far the most interesting and challenging problem that remains in real-time applications involving data-compressed stored speech is incorporating the SOLA technique before decompression. Most voice messaging systems employ some sort of data compression-decompression scheme to reduce storage requirements of voice messages. Data compression-decompression

often forms the bulk of the computational requirements in voice messaging systems. In addition to the computational burdens of performing time-scale modification, speed-up (time-scale compression) requires decompressing the voice data at an increased rate. Speed-up by a factor of two requires decoding of the data at twice the normal rate. This can present significant difficulties because the computational requirements of the messaging system are more than doubled.

If the SOLA algorithm were to operate on the compressed data representing the speech signal before decompression, a significant reduction in computations may be possible. Sub-band coding is one technique which may allow this. Sub-band coding divides the speech signal into frequency bands, then dynamically allocates bits to represent information in the sub-bands accordingly. The division into sub-bands represents a linear, time-invariant operation. Moreover, the simple overlap-add update employed by the SOLA algorithm is also linear and time-invariant. It may be possible to perform the overlap-add in the sub-band domain prior to decompression. This would allow time-scale compressed messages to be data-decompressed at a normal rate.

Many sub-band coders decimate the data in the sub-bands of the signal. Decimation restricts the shift resolution of the SOLA algorithm in such a scheme because no intra-sample overlap-adding is possible for data in the sub-bands. The time-scale modification performed, however, is dependent only on the ratio S_s/S_a, thus the decimation rate does not limit the choice of modification factors.

Appendix A

Listen Tests

Due to the subjective nature of the quality measurement, implementations were compared in double-blind A-B preference tests. Sentences A and B were played in pairs and listeners indicated a preference for either A or B.

Each test pair of sentences A versus B represents an utterance selected at random from a group of 8 utterances. Sentences A and B are two versions of the selected utterance processed using a parameter set A and B respectively. The number of trials reflects the number of different test pairs (a single utterance) used to compare parameter sets A and B. In cases where more than one test pair is used, the order in which the pairs are presented to the listener is forced to be random and equal plus or minus one. This is done to counteract any order-dependent bias associated with the comparison.

All tests were performed on speech bandlimited to 3.8 kHz and sampled at 8 kHz. Numbers appearing in parentheses after durations in milliseconds refer to the corresponding number of samples of a signal sampled at 8 kHz unless otherwise noted. Eight different utterances collected from eight different speakers (4 male and 4 female) were used for the listen tests.

A.1 Listen Test I

The first listen test was used to determine the preferred window length and analysis shift for time-scale compression by a factor of two. Four groups of parameter sets were chosen for this test:

P-set #	α	*winlen*	S_a	S_s	K_{max}
1	$\frac{1}{2}$	100	100	50	100
2	$\frac{1}{2}$	200	200	100	100
3	$\frac{1}{2}$	300	300	150	100
4	$\frac{1}{2}$	400	400	200	100

This selection allowed comparisons of different analysis shifts on different utterances to determine the point at which degradation in the output signal became detectable. The goals of this test are two-fold. First, the amount of averaging, AWO=1 and SWO=2, is held constant throughout the parameter sets, and multiple feature appearances are eliminated to isolate the effects of analysis shift and window length. A relative measure of the stationarity of each speech signal can be inferred from the results. Additionally, the affects of different window lengths for the units of speech used in the algorithm can be gauged.

Because the affect of block size is most easily isolated using time-scale compression, the listen test was restricted to time-scale compression by a factor of 2 ($\alpha = 0.5$). To limit the number of test sentence pairs evaluated by the listeners, the following pairs were compared:

Sentence A	Sentence B	Number of trials
P-set #3	P-set #1	6
P-set #3	P-set #2	8
P-set #3	P-set #3	3
P-set #3	P-set #4	7

The foundation of this test was based on the results of earlier findings indicating that 12.5 msec (100) was inadequate for male speech while 50 msec (400) was excessive for female speech. The preferred value was therefore predetermined to fall within the range of 25 to 37.5 msec (200 to 300).

A.1.1 Summary

The results presented in Figures A-1 and A-2 indicate a slight preference among listeners for a window length of 12.5 msec (100) over 37.5 ms (300) for female speech, but for male speech a window length of 37.5 ms (300) was clearly preferable to 12.5 ms (100). The combined

responses, however, confirm that the degradation in quality (pitch fracturing) associated with the shorter 12.5 ms (100) window length for male speech is worse than the degradation (slight reverberation) associated with the longer window length 37.5 ms (300) for female speech.

This preference is the result of pitch period differences among the two categories. A window length of 25.0 ms (200) provided comparable quality to the 37.5 ms (300) window for both male and female speakers. Due to the differing effects of window lengths on perceived quality it is better to use a longer window with slight reverberation than risk pitch fracturing using too short a window. Since using 256 sample points provides a convenient data block size for computation, a window length of 32 ms (256 sample points) is suggested as a reasonable compromise between 25 and 37.5 msec (200 and 300).

A.1.2 Listen Test Results for Speed-Up Using Various Window Lengths

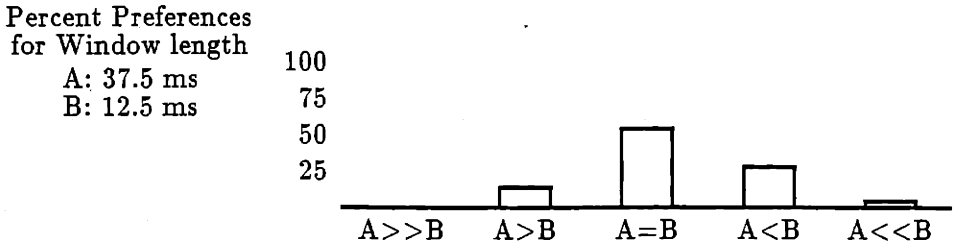


Figure A-1: 37.5 ms window (A) v.s. 12.5 ms window (B): Responses for female speech only.

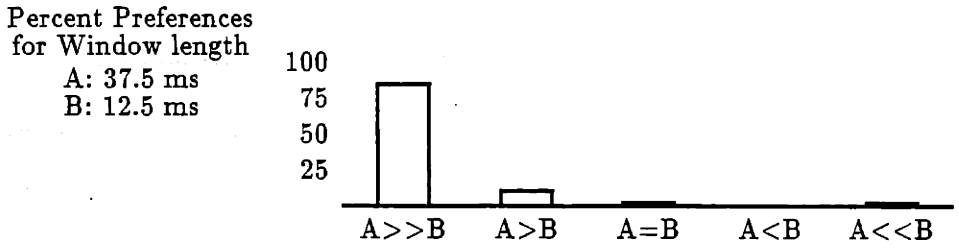


Figure A-2: 37.5 ms window (A) v.s. 12.5 ms window (B): responses for male speech only.

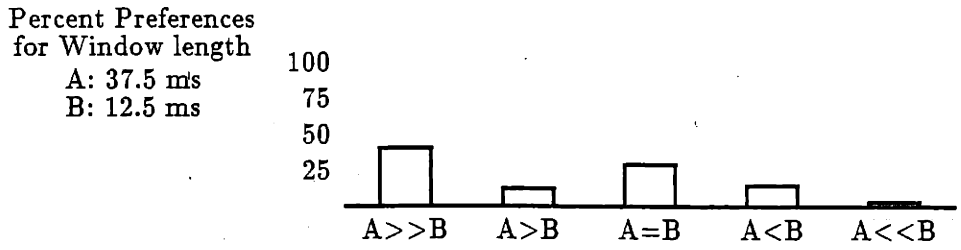


Figure A-3: 37.5 ms window(A) v.s. 12.5 ms window(B): Combined responses for male and female speech.

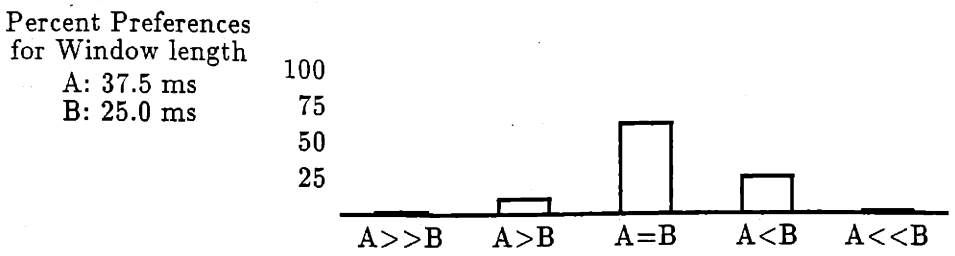


Figure A-4: 37.5 ms window (A) v.s. 25.0 ms window (B): Combined responses for male and female speech.

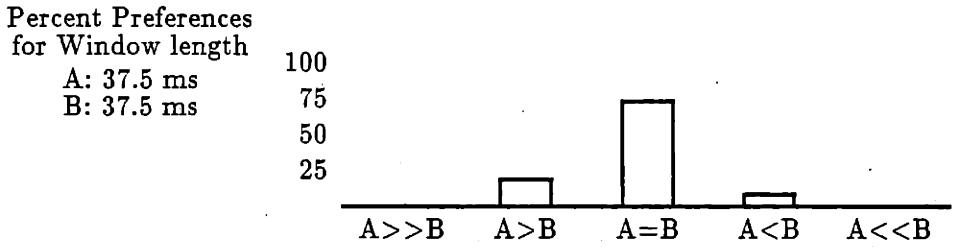


Figure A-5: 37.5 ms window (A) v.s. 37.5 ms window (B): Combined responses for male and female speech.

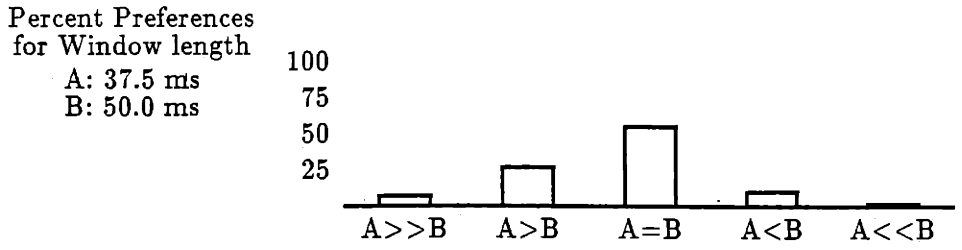


Figure A-6: 37.5 ms window (A) v.s. 50.0 ms window (B): Combined responses for male and female speech.

A.2 Listen Test II

The second listen test was performed to determine the point at which increases in the value of S_a resulted in speech of detectably lower quality. This test also allowed additional information on the effects of window length, analysis shift, and output signal averaging on the quality of time-scale expanded-by-two speech ($\alpha = 2$) to be examined. The following parameter sets were chosen for this test:

P-set #	α	*winlen*	S_a	S_s	K_{max}	AWO	SWO
220	2	200	20	40	100	10	5
320	2	300	20	40	100	15	7.5
380	2	300	80	160	100	03.75	1.875
312	2	300	125	250	100	02.4	1.2

To limit the number of test sentence pairs evaluated by the listeners, the following pairs were compared:

Sentence A	Sentence B	Number of trials
P-set #320	P-set #220	6
P-set #320	P-set #380	5
P-set #320	P-set #320	3
P-set #320	P-set #312	6

A.2.1 Averaging in Output Signal

The first comparison was used to gauge the effect of window length and output signal averaging on the perceived quality of time-scale expanded speech from several different speakers. The results indicate no perceptible difference between averaging indexes of AWO = 5 vs. AWO = 7.5 for a fixed analysis shift of 2.5 ms (20 samples).

A.2.2 Analysis Shift

The effect of increasing the analysis shift from 2.5 ms (20) to 10.0 ms (80) to 15.625 ms (125) were evaluated. The results indicate a distinct preference for the 2.5 ms (20) shift over the 10.0 ms (80) shift when a difference was detected. For much of the speech, however, no

obvious difference was detectable (see Figure A.8). The decrease in quality from increasing the shift to 15.625 msec (125) is readily apparent from the listeners' preferences (see Figure A.9). From the data it can be inferred that values for the analysis shift between 2.5 ms (20) and 6.25 ms (50) should provide speech of equal perceived quality.

A.2.4 Summary

This test was performed to estimate the points at which increases in the number of frames/unit time offered no perceptual improvement in quality. The critical point appears to fall just slightly below 10.0 ms for time-scale expansion by a factor of two using a window length of 37.5 msec and a shift search interval of 12.5 msec.

A.2.3 Listen Test Results for Slow-Down Using Several Analysis Resolutions

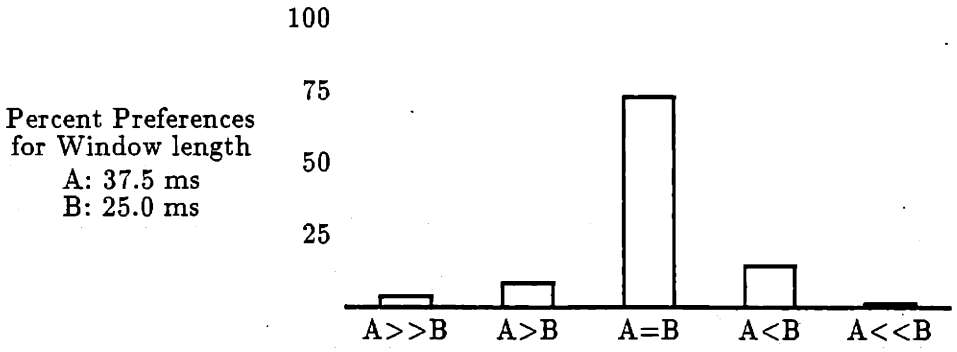


Figure A-7: Differing window lengths for a fixed Analysis Shift of 2.5 ms.

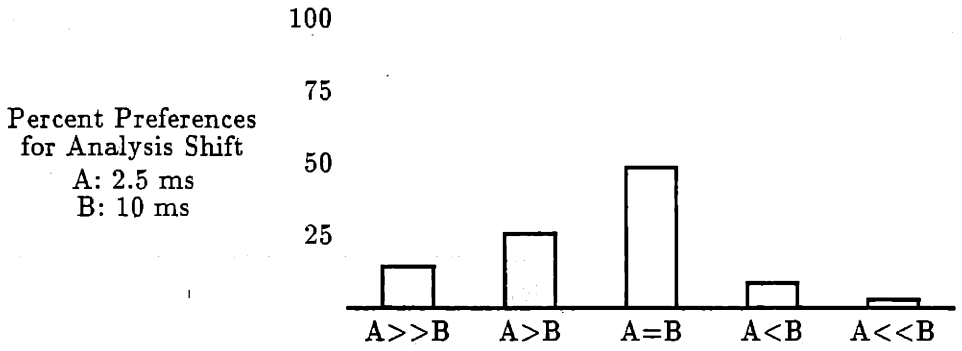


Figure A-8: Differing Analysis Shifts for a fixed Window Length of 37.5 ms: 2.5 ms (A) v.s. 10.0 ms (B).

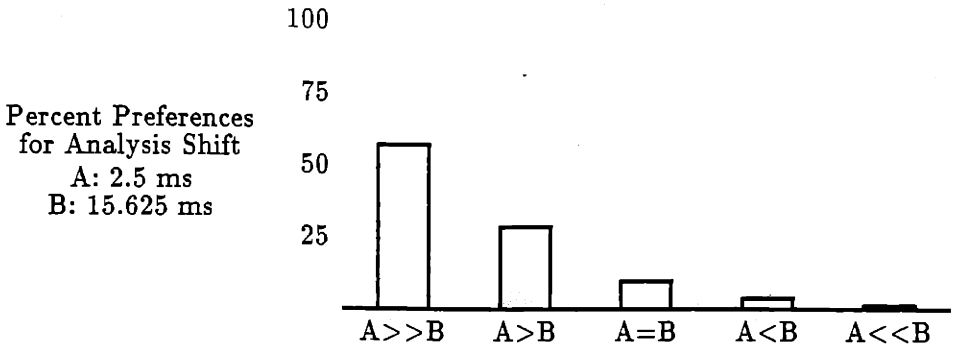


Figure A-9: Differing Analysis Shifts for a fixed Window Length of 37.5 ms: 2.5 ms (A) v.s. 15.625 ms (B).

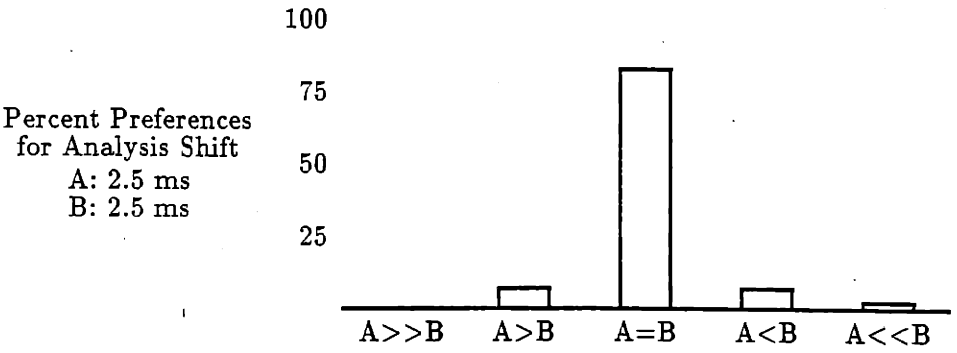


Figure A-10: Identical Analysis Shifts for fixed Window Length of 37.5 ms: 2.5 ms (A) v.s. 2.5 ms (B).

Bibliography

[1] 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.

[2] N. S. Jayant and P. Noll. *Digital Coding of Waveforms*, pages 658–665. Prentice-Hall, Englewood Cliffs, NJ, 1984.

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

[4] 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.

[5] M. R. Portnoff. *Time-Scale Modification of Speech Based on Short-Time Fourier Analysis.* PhD thesis, M.I.T., 1978.

[6] T. F. Quatrieri and R. J. McAulay. Speech transformations based on a sinusoidal representation. *IEEE Transactions on ASSP*, ASSP-34:1449–1464, December 1986.

[7] L. R. Rabiner. On the use of autocorrelation analysis for pitch detection. *IEEE Transactions on ASSP*, ASSP-25, No. 1:24–33, February 1977.

[8] L. R. Rabiner, M. J. Cheng, A. E. Rosenburg, and C. A. McGonegal. A comparitive performance study of several pitch detection algorithms. *IEEE Transactions on ASSP*, ASSP-24, No. 5:399–418, October 1976.

[9] L. R. Rabiner and R. W. Schafer. *Digital Processing of Speech Signals.* Prentice-Hall, Englewood Cliffs, NJ, 1978.

[10] M. J. Ross, H. L. Shaffer, A. Cohen, R. Freudberg, and H. J. Manley. Average magnitude difference function pitch extractor. *IEEE Transactions on ASSP*, ASSP-22, No. 2:353–362, October 1974.

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

[12] M. M. Sondhi. New methods of pitch extraction. *IEEE Transactions on Audio and Electroacoustics*, AU-16, No. 2:262–266, June 1968.

[13] J. Wayman, E. Reinke, and D. Wilson. High quality speech expansion, compression, and noise filtering using the sola method of time scale modification. Obtained from James L. Wayman, 1989.

[14] J. L. Wayman and 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*, 36, No. 1:139–140, January 1988.