APPENDIX 9

TIME MODIFICATION OF SPEECH. THEORY: SPEECH ANALYSIS, SEGMENTATION, AND LABELING

A9.1 INTRODUCTION

The time-modification system varies the duration of selected segments of the speech signal. The segment duration can be lengthened or shortened using the software toolbox described in Chapter 8. Possible segments include vowels, nasals, unvoiced fricatives, and so forth. Each segment duration is modified according to parameters specified by the user. These parameters apply to either all occurrences of a specific type of segment, or to a single occurrence of a segment. One example of a parameter that applies to a vowel is the vowel scale factor, SF_{vowel}. The vowel scale factor specifies the desired ratio of the duration of the vowel segment(s) in the time-modified word to the duration of the corresponding vowel segment(s) in the original, unmodified word.

Since the speech segments are the basis of the time-modification system, it is important that the segments are accurately detected and identified. (For brevity, the detection and identification processes will hereafter be called detection.) To accomplish this goal, the system designer is faced with one of two choices: manual detection (by hand) or automatic detection (software). While manual detection provides good results, it is extremely tedious and time consuming. This limits the usefulness of the overall time-modification system. Automatic detection is quick and relatively "painless," but is more prone to mistakes than the manual method and requires significant initial development time.

As a compromise, we use automatic detection of the speech segments with subsequent manual editing to correct errors that may occur in the detection process. Automatic detection consists of three main steps: (1) speech analysis; (2) segmentation of the word or sentence into segments of unknown type; and (3) appropriate labeling of these segments. The manual editing process allows the user to display and edit the automatic segmentation and labeling results using a set of software programs with a convenient, easy-to-use, graphical user interface (GUI). The GUI allows the user to insert silent segments, change segment boundaries, change segment labels, and merge adjacent segments that have been labeled as the same type. This is accomplished with the click of a workstation mouse. The software toolbox is described in Chapter 8.

This appendix describes both the automatic and manual algorithms used to analyze, segment, and label the speech segments. It begins with a discussion of the selection of speech segment categories. Next, a brief overview of the automatic segment detection process is presented with an example of automatic detection of the voiced/unvoiced/silent (V/U/S) parameter for the word sue. The example illustrates the methods associated with automatic detection of a single parameter, or "feature," of speech. The relevance and application of these methods to the general set of feature detection programs is described. Each of the automatic feature detection algorithms is then described in detail. The segmentation and labeling processes are also presented in detail. The mistakes produced by the automatic algorithms are discussed. The editing system and corresponding GUI are described in Chapter 8.

A9.1.1 Time Modification Methods of the Past

The methods used to accomplish time modification have evolved through several stages over the last 40 years. Although there are variations in the specific implementations, almost all of the methods used to date can be assigned to one of four main categories.

A9.1.2 Variable-Playback-Rate Method

The variable-playback-rate method is relatively simple, and accomplishes a rate change by playing back previously recorded speech at a rate different from the original recording rate. An example of this is playing an LP phonograph record at 45 RPM, instead of its intended rate, 33 1/3 RPM. A modern, digital signal processing (DSP) analogy of this is digital-to-analog conversion (with appropriate filtering) at a sampling rate different from the signal's original sampling rate, without interpolation or decimation. Note that for this technique, the rate change is always accompanied by a linear shift in the frequency content of the signal. For speech that has been slowed down or speeded up by a factor of about two or more, a frequency shift leads to undesirable perceptual effects that mask the identity of the speaker, and, in general, causes a decrease in intelligibility (Garvey, 1953a, 1953b; Tiffany and Bennett, 1961).

The variable-playback-rate method was never popular among researchers, mainly because of the detrimental perceptual effects attributed to the accompanying pitch shift. However, it was the only method available until about 1950, when the sampling method was introduced.

A9.1.3 Sampling Method

The general class of rate-change techniques known as the "sampling method" involves the periodic removal or duplication of small segments of recorded speech. The remaining segments are then spliced together to form the rate-altered speech. The main advantage of this method is that the frequency content of the resulting speech is not affected, and as a result, many of the speaker-dependent characteristics are preserved. It has also been shown that for a variety of different rates, the speech produced by this method is significantly more intelligible than speech produced by the variable-playback-rate method (Fletcher, 1929; Garvey, 1951; Lee, 1972).

One way of implementing the sampling method is by manually cutting and splicing magnetic recording tape (Garvey, 1949). A disadvantage of this method is the time required to perform the task. Another disadvantage is that it is impossible to guarantee waveform continuity across the splice boundary. This results in audible "pops" and "clicks," although these clicks can be reduced by cutting the tape at a 45-degree angle relative to the edge of the tape. This method does, however, have the advantage of allowing the user to (manually) select the locations and durations of the discarded segments. To mark the tape, the operator manually passes the tape back and forth across the playback head of a tape recorder and listens for the starting and stopping points of a syllable. Once these points are found, they are marked and labeled on the back of the tape with a grease pencil. This method is flexible, but extremely time consuming. As a result, it is often impractical for extensive use and is typically used only for "proof of concept."

Today, this "cut and splice" method is typically implemented on a digital computer using a video display. Although the physical inconvenience of the magnetic tape is no longer present, the problems of identifying the desired segment and maintaining waveform continuity across the splice boundary remain.

An automatic time-modification method exists that is based on a modified magnetic tape recorder (Fairbanks et al., 1954). It involves rotating the playback head assembly that contains four playback heads. As the head assembly rotates, each one of the four playback heads individually and sequentially contacts the moving magnetic tape. The outputs from the four heads are wired in parallel. Time compression results from the fact that there are gaps between each of the four heads. Therefore, as the head assembly rotates, every segment of the tape that these gaps contact (instead of a playback head) are not reproduced by the playback head. The spacings in the head assembly are calculated so that as the head assembly turns, one playback head is always beginning to make contact with the tape just as the previous playback head is loosing contact with the tape. The output of the rotating head

assembly is rerecorded onto a second, conventional, tape recorder. The second tape then contains the compressed speech.

The initial automatic time-modification device introduced by Fairbanks in 1954 has some major limitations. One problem is that the duration of the segments that are discarded or duplicated is fixed. This was changed in later adaptations and copies of the original machine (Lee, 1972; Neuburg, 1978). However, other problems ultimately limit the fidelity and usefulness of the machine. The first of these problems is the lack of repeatability of the process. In order to repeat the process of compressing a tape segment, the user has to know exactly the starting phase of the rotating head assembly with respect to the beginning of the tape. The second problem is the noise created by the rotating head assembly's slip rings and distortion due to the heads' misalignment with the moving tape. Despite the shortcomings of the method, the large majority of published research on the intelligibility and comprehensibility of compressed speech was done using the sampling method.

A9.1.4 Vocoder Methods

A third class of rate change techniques is accomplished by the use of vocoders (VOice CODERS). Vocoders were originally designed to reduce the bandwidth requirements for transmission of a normal voice signal. Their ability to modify the rate of speech is thought of as a secondary benefit. Of all of the vocoders, the phase vocoder is the best suited for rate modification (Flanagan and Golden, 1966; Rabiner and Schafer, 1978).

Vocoders implement an analysis–synthesis speech transmission scheme. In the analysis stage, natural speech is analyzed, typically by a bank of bandpass filters. The output of each bandpass filter in the bank is coded by one of a variety of different methods, and this coded information is transmitted across a channel. At the synthesis stage at the receiving end of the channel, the coded information is decoded, and is used to control a bank of tuned oscillators. The outputs of the oscillators are then summed to produce synthesized speech (Rabiner and Schafer, 1978).

Typically, the synthesis oscillators are tuned to the same frequencies as the bandpass filters in the analysis stage. However, this one-to-one match in tuning is not strictly required, and if the oscillator frequencies are tuned to multiples of the analysis stage's bandpass filters, it is possible to implement a modification of the synthesized speech. For example, the phase vocoder can be used to implement a rate change in the following two-stage manner. In the first stage, speech is analyzed by a bank of equally spaced bandpass filters with center frequencies at ω_i for $i = 1, \ldots, N$. The outputs of the bank of bandpass filters are then used to control a bank of oscillators tuned to center frequencies $\omega_i/2$. At this point, the rate of the synthetic speech is identical to that of the original speech, but the frequency spectrum of the synthetic speech is shifted down to one-half that of the original speech. The second stage of the process is to double the playback speed of the speech synthesized by the first stage. The resulting speech is twice the rate of the original speech, and has the same spectrum as the original speech.

While vocoders are able to modify the rate of speech, they suffer from the fact that their analysis–synthesis schemes create unwanted artifacts in the speech signal (Portnoff, 1981). The speech produced by vocoders is often described as sounding artificial or "buzzy." Another problem previously associated with the use of vocoders for research applications is that in the 1960s and 1970s, vocoders were relatively expensive and not a cost-effective option for many researchers.

A9.1.5 Recent Methods

Over the last 10 to 15 years, other methods for modifying the rate of speech have evolved. While the speech produced by these methods is seldom tested in formal intelligibility or comprehensibility tests, the methods are being studied due to their low cost, low computational requirement, and relative ease of implementation. Note that some of the newer methods are hybrids of older vocoder technology and recent waveform coding technology.

The simplest new method consists of "a pitch detector followed by an algorithm that discards (or repeats) pieces of speech equal in length to a pitch period" (Neuburg, 1978). This is a minor variation of the sampling method. The method does not operate pitch-synchronously, which means that the beginning of the segment that is either duplicated or discarded does not occur at the instant of

glottal closure. The method relies upon the fact that for the majority of time, the speech signal does not vary greatly across a single pitch period (about 10 msec for a male speaker). Therefore, as long as the duration of the discarded (or repeated) segment is exactly equal to the pitch period, the ear can not discern any significant distortion from the process. No formal listening tests have been conducted for this method.

Another speech rate modification method, by Malah (Malah, 1979), is similar in principle to the two-step process implemented by the phase vocoder. For both of these methods, if the speech rate is modified by an integer scale factor n, the first step shifts the frequency spectrum by a factor of n, and the second step plays the frequency-shifted speech at a speed of n times the original rate. Since the second step of this process is essentially trivial, the success of this method relies on the ability to shift the frequency spectrum of the speech. Malah developed numerically efficient algorithms that shift the frequency spectrum of speech (without changing the rate). These algorithms are known as time-domain harmonic scaling (TDHS) algorithms. In most cases, the algorithms require only one multiplication and two additions per output sample of speech. The primary problem in this implementation is that rate modification is only implemented in integer multiples (i.e., 2:1, 3:1). Thus, only a small, finite set of compression and expansion ratios can be implemented. Malah claims that for the two-step rate modification scheme, rates of greater than 2:1 are impractical, "due to perceptual limitations." Because of this limitation to integer multiples, the algorithm is not that applicable to speech research. Although no formal tests were conducted, the author states that "Simulation results with a scaling factor of two for different speakers and texts have been informally judged to be very good. . . ."

A recent and popular approach for time modification is based upon the short-term Fourier transform (STFT). The method is composed of three parts. The first part models the speech signal with the STFT. The second part modifies the STFT parameters to implement the rate change. The third part synthesizes the modified speech signal from the modified STFT parameters (Portnoff, 1981). While no formal listening tests were performed, the author claims that the system "is capable of producing high quality rate-changed speech . . . for compression ratios as high as 3:1 and expansion ratios as high as 4:1." For ratios outside this range, the method introduces reverberation for expanded speech, and exhibits a "rough" quality for compressed speech.

Another recent approach models the speech signal by a set of time-varying sine waves (Quatieri and McAulay, 1992). In terms of the general procedure, this method is very similar to the STFT approach. The speech is modeled by a set of time-varying parameters, namely sine wave amplitudes and phases. The algorithm adjusts the speech parameters, and then resynthesizes the modified speech by controlling a set of sine wave generators. Again, no formal listening tests were conducted, and the authors state that for time-modified speech, "the synthesized speech was generally natural sounding and free of artifacts." Perhaps the most interesting point in the Quatieri and McAulay study is that they experiment with what they call "speech-adaptive time-scale modification." In essence, they implement rate change by modifying only the voiced portions of speech. In addition, they measure the "degree" of voicing, and concentrate their time-base modification on the frames that exhibit the highest degree of voicing. Note that their measurement of the degree of voicing is based upon how little the harmonic structure varies across multiple frames, that is, the less the harmonics vary, the higher the "degree of voicing." However, no formal listening tests were conducted.

A9.1.6 Phonological and Psychological Testing

Time modification of speech is a research methodology that is used to study certain aspects of speech. In general, psychological testing is concerned with such issues as (1) determination of the highest rate at which speech can be presented to a listener and still be understood; (2) an explanation of why our perceptual process fails at higher speaking rates; and (3) the role of short-term and long-term memory in speech perception. In contrast, phonological testing is often conducted by speech pathologists. The goal is to model the human perception of phonemes (or similar segments of speech). The results are discussed in terms of measurable acoustic features and how their presence (or absence) affects perception. The literature in these areas is reviewed in White (1995). Many of the studies in the literature use a computerized cut and paste method. However, such methods rely on the user to identify various speech segments within a word. In effect, previous systems were manually operated waveform editors.

A9.2 SELECTION OF SPEECH SEGMENT CATEGORIES

The complexity of the algorithms for segmentation and labeling in any speech analysis task depends upon the degree of recognition that must be achieved (Davis and Mermelstein, 1980). For a given speech sound, algorithms that determine only the phoneme category will be less complicated than algorithms that determine not only the category, but the identity of the phoneme as well. Likewise, for a given speech sound, algorithms that determine the allophonic variation of a particular phoneme can be expected to be complicated, due to the number of variations of the pronunciation of a single phoneme that can occur during conversational speech (Klatt and Stevens, 1973). Thus, the complexity of the segmentation and labeling task is dependent upon both the number and choice of categories used to subdivide the speech.

There are several possibilities for selecting speech segment categories. In a top-down paradigm, the simplest choice is to classify speech as voiced, unvoiced, or silent (V/U/S). The next, more complex choice is to classify each speech sound as a member of one of the basic phoneme types. Although the exact description and number of different phoneme categories vary slightly depending upon the school of thought, these categories usually include vowels, nasals, semivowels, voiced fricatives, voiced stops, unvoiced fricatives, unvoiced stops, and silence. An even more complex categorization requires identification of the exact phoneme. This requires matching the segment under consideration with one of the 41 or so phonemes in English.

For the system described here, speech is divided into eight segment categories: vowels, semivowels, nasals, voiced fricatives, voice bars, unvoiced stops, unvoiced fricatives, and silence. Overall, this choice is a compromise between the complexity of the segment recognition algorithms and the resolution of the resulting speech segments. We assume that recognition of individual phonemes is not required. This greatly reduces the complexity of the segment recognition algorithms, since the choices in the matching process are reduced from 41 to eight. In addition, it is easier to recognize the typically large differences between phoneme categories than it is to recognize smaller differences between various phonemes of the same category (Schwartz and Makhoul, 1975).

A9.3 OVERVIEW OF AUTOMATIC SEGMENT DETECTION

Automatic detection of the speech segments is accomplished by a series of software programs that sequentially analyze the speech signal. These programs are grouped according to the task they perform in the detection process. The three main tasks are shown in Figure A9.1 as: (1) speech analysis; (2) segmentation of the word or sentence into segments of unknown type; and (3) labeling of these segments with the most appropriate segment label. A brief overview of each of these tasks follows, with a more detailed discussion provided later.

Speech analysis is the most complicated of the three tasks, and is divided into several steps. A block diagram of the speech analysis task is shown in Figure A9.2. The initial analysis and decomposition of the speech waveform is derived from a two-pass method developed by Hu (1993) (see also Childers and Hu, 1994) and is essentially the same method as that used for the voice conversion toolbox. In the first pass, the sampled waveform is divided asynchronously into 5 msec frames. A 13th-order, linear predictive coding (LPC) analysis is performed for each frame, and the residue is processed to determine the glottal closure points (Hu, 1993). Only the glottal closure indices (GCI) are retained for use in the second pass of the algorithm. In the second pass, the sampled waveform is again divided into frames. The frames are chosen pitch asynchronously for unvoiced speech and silence, and pitch synchronously for voiced speech, using the glottal closure indices as a reference. A 13th-order, LPC analysis is performed for each frame, and the residue, LPC coefficients, and power are saved for later modification and synthesis. Next, a set of feature detection algorithms analyzes each frame individually. Each algorithm in the set detects a different acoustic feature. For example, one of the algorithms detects the presence or absence of nasals, while another algorithm detects the presence or absence of semivowels. Each feature detection algorithm uses a combination of fixed thresholds, median filtering, and empirical rules to calculate the final result, or "feature score."

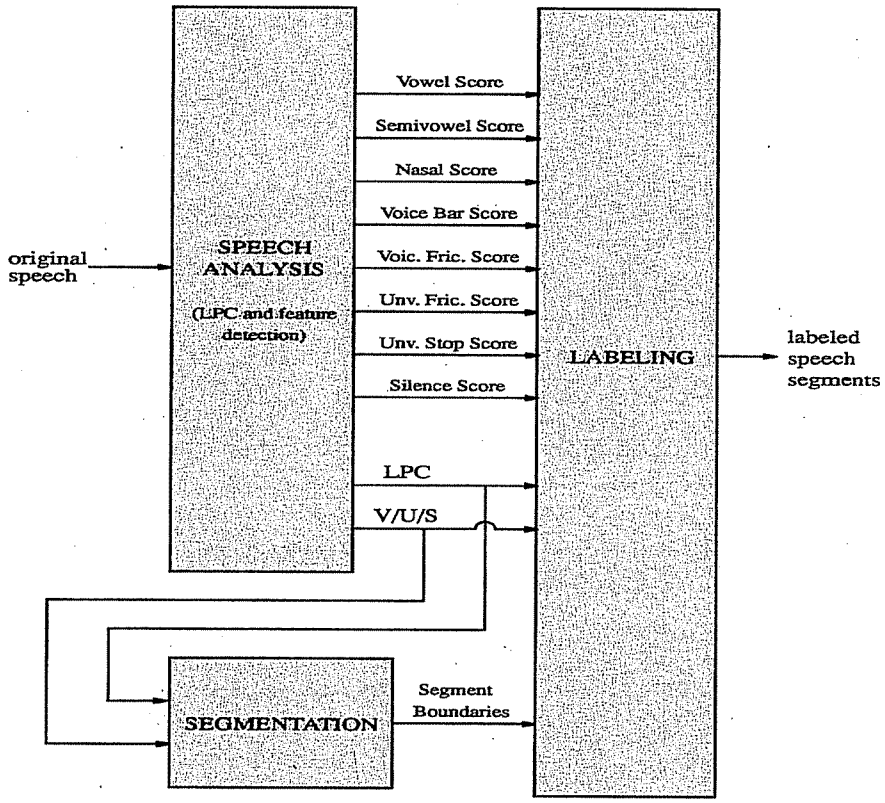

FIGURE A9.1 Block diagram of automatic speech detection.

The second automatic detection task shown in Figure A9.1 is determination of the time-domain boundaries that separate the segments of the speech signal. This process is known as segmentation. The boundaries are chosen such that each segment has relatively stable acoustic properties for the duration of the segment. Segmentation is accomplished by combining the results of two different algorithms. The first algorithm determines the changes in the "trend" of the short-term frequency spectra, and the second uses the results of the voiced/unvoiced/silent (V/U/S) feature detector.

The third task in Figure A9.1 is labeling of the segments using the results obtained from the feature detection algorithms. Examples of labels are vowel, semivowel, and unvoiced fricative. Labeling is done in two steps. First, the average feature detection scores are calculated. Next, empirical rules are applied to the average scores to determine the most appropriate label for each segment.

A9.4 FEATURE DETECTION ALGORITHMS—GENERAL DEVELOPMENT

Acoustic feature detection is the search for different (acoustic) features. Examples of acoustic features include voicing, nasality, and sonorance. While acoustic features are used to help differentiate between the various segment categories, it is important to realize that individual acoustic features may not be unique to one particular segment category. For example, nasality may indicate the presence of a nasal, or it may indicate the presence of a nasalized vowel. Thus, in this example, one acoustic feature is common to two different segment categories. This lack of one-to-one correspondence between acoustic features and segment categories requires that multiple acoustic features be evaluated and weighed when attempting to match an unknown speech segment with the most appropriate segment label.

Although it is logical to define the term "segment detector" as an algorithm that detects one of the eight segment types listed in Section A9.2, this term is misleading. It can be confused with the previously defined definition of segmentation, which is the task of dividing the speech signal into segments of unknown type. Therefore, the term "feature detector" is used in a broad sense, and implies both an algorithm that detects a single acoustic feature, as well as an algorithm that detects multiple acoustic features in order to detect one of the eight segment types listed in Section A9.2.

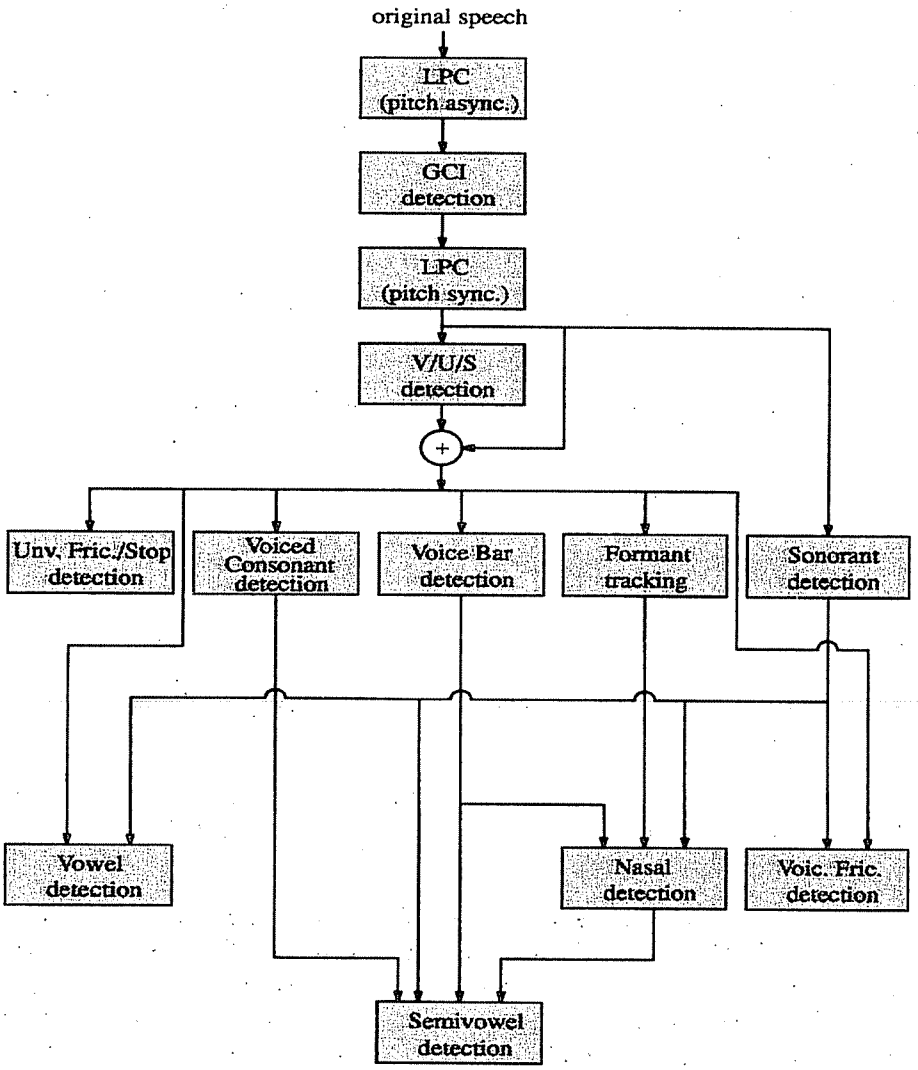

FIGURE A9.2 Block diagram of speech analysis.

Feature detection is achieved by algorithms that examine the short-term frequency spectra of the speech signal. The spectra are calculated from the LPC coefficients that are, in turn, calculated for each frame of the signal during the initial analysis stage. It has been shown that the short-term frequency spectra method is a reliable technique that is used in a wide variety of recognition systems (Bush et al., 1983; Glass and Zue, 1986; Glass and Zue, 1988; Klatt, 1977; Leung et al., 1993; McCandless, 1974; Meng and Zue, 1991; Mermelstein, 1977; Weinstein et al., 1975; Zue et al., 1989).

Each feature detection algorithm utilizes a sequence of processing stages to calculate the resulting feature score. In many instances, the structure of each of the feature detection algorithms is similar, although the exact numerical values may differ.

The detection of acoustic features from the speech signal is the most complicated portion of the analysis, segmentation, and labeling process. Because of the complexity of the feature detection algorithms, the explanation of the algorithms is broken down into two sections. In this section, a simple example is given to explain one feature detection algorithm and its development and implementation. Although the example is for a single feature detector, it illustrates the general structure of the majority of the feature detectors. The example also discusses the problems and considerations associated with the set of feature detection algorithms as a whole. In the next section, the algorithms are detailed individually, examining the specific equations of the algorithms.

The feature detection algorithms use a combination of methods to produce the final results. These methods include bandpass filters, fixed thresholds, median filter smoothing, and empirical pattern recognition rules. These methods are used in a similar manner in each of the algorithms. The example that follows illustrates how these methods work in a feature detection algorithm that detects the voiced/unvoiced/silence (V/U/S) feature of speech.

A9.4.1 Input Data and V/U/S Pre-Processing

All of the feature detection algorithms require the LPC results as input data. Most of the feature detectors also require the results from other feature detectors (specifically the V/U/S results), as shown in Figure A9.2.

V/U/S classification is different from the other feature detection algorithms in that a portion of the algorithm is accomplished during the initial LPC analysis algorithm. During the first pass of the LPC algorithm, the first reflection coefficient is calculated for each pitch-asynchronous frame. The frame is classified as voiced (V) if the reflection coefficient is greater than 0.2, and is classified as unvoiced (U) if the reflection coefficient is less than or equal to 0.2. This threshold was determined empirically by Hu (1993). In addition, Hu makes no distinction between unvoiced and silent frames. Therefore, all silent frames are classified as unvoiced. During the second pass of the LPC algorithm, certain frames are labeled as transitional frames (T). The first voiced frame in an unvoiced-voiced sequence, and the last voiced frame in a voiced-unvoiced sequence are changed to transitional frames. Hu's explanation for this is that mistakes may be made in the simple V/U decision process, so the frames at the transition regions are marked, since this is typically where the mistakes are made. Since it has been observed that the transition frames are always voiced, all transition frames are converted to voiced frames in this system.

A9.4.2 Volume Function

A volume function, V(i), similar to one presented by Weinstein et al. (1975) is calculated for each frame to determine a quantity analogous to the loudness, or acoustic volume, of the signal at the output of a hypothetical bandpass filter. This is the first processing step in the majority of feature detectors. The volume function is normalized by the number of samples in the frame, and is given as

$$V(i) = \frac{1}{N_i}\sqrt{\sum_{m=A}^{B}\left|H_i\left(e^{j\pi\frac{m}{256}}\right)\right|^2} \qquad \text{(A9.4.2.1)}$$

where i is the current frame index, N_i is the number of samples in frame i, A is the index of the low cutoff frequency of the bandpass filter, B is the index of the high cutoff frequency of the bandpass filter, and $H_i(e^{j\pi\frac{m}{256}})$ is the complex, single-sided, frequency response of the IIR filter, $H_i(z)$, produced by the LPC coefficients and evaluated at the points $\exp(j\pi m/256)$, for $0 \le m \le 255$. $H_i(z)$ is given as

$$H_i(z) = \frac{G(i)}{a_0 + a_1 z^{-1} + a_2 z^{-2} + \cdots + a_N z^{-N}} \qquad \text{(A9.4.2.2)}$$

where $N = 13$, $a_0 = 1$, and G(i) is given as

$$G(i) = \sqrt{\sum_{n=s}^{t} r^2(n)} \qquad \text{(A9.4.2.3)}$$

where r(n) is the value of the LPC residue at sample n, i is the current frame index, s is the beginning sample number of the current frame, and t is the ending sample number of the current frame.

The volume function of Equation (A9.4.2.1) is used extensively in the software, although the frequency range of the bandpass filter varies depending upon the specific detector. In addition, many of the feature detection algorithms calculate the ratio of two volume functions, each with its own frequency range. This compares the energy in one frequency band to the energy in a second frequency band.

In the majority of feature detectors, median filtering is done to smooth any large, short-term, fluctuations in the volume function. The fluctuations are caused by a variety of sources including incorrect GCI determination, incorrect V/U/S classification, and recording artifacts such as background noise. Although the majority of the feature detectors use a 5th-order median filter for smoothing, the exact order is given in the detailed description of each detector. The filter order is determined empirically in each case.

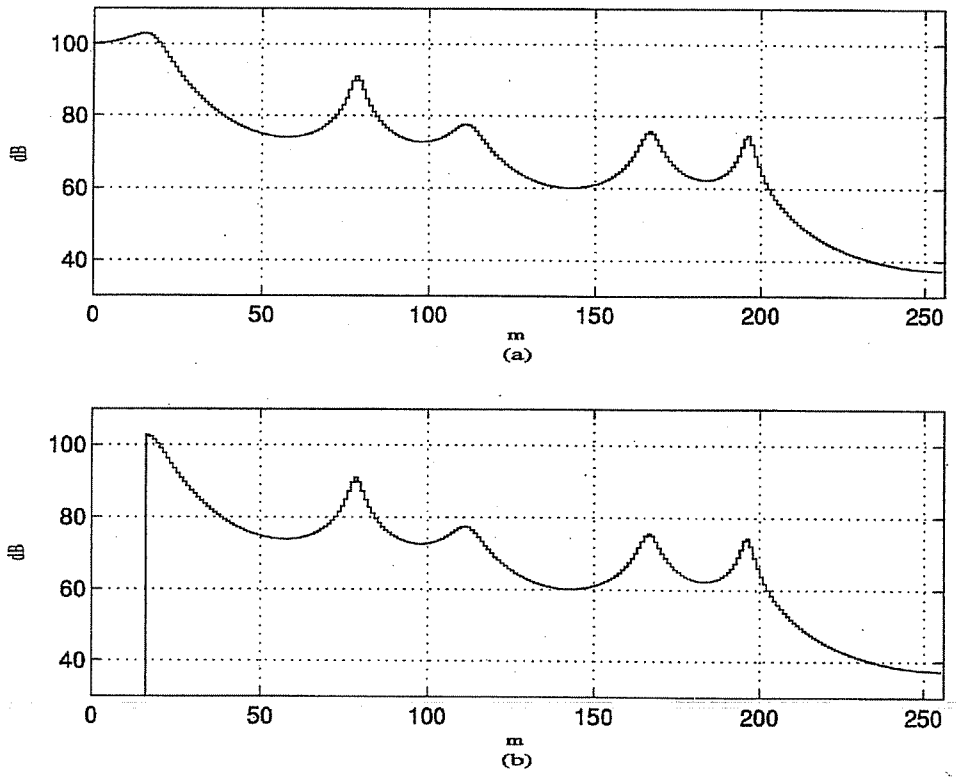

FIGURE A9.3 $H_i(z)$ used to calculate the V/U/S volume function for one pitch period of the vowel portion of the word sue. (*a*) $H_i(z)$ before filtering. (*b*) $H_i(z)$ after filtering.

The V/U/S detector uses a single volume function of Equation (A9.4.2.1) with the values A = 17 and B = 255. The lower limit of A = 17 serves to highpass (HP) filter the frequency response with a cutoff frequency of 312 Hz (the upper limit B = 255 corresponds to one-half the sampling rate, thus a highpass instead of a bandpass filter). Weinstein claims that the HP filter is needed to reduce the sensitivity to voiced stops, but experiments show that its primary effect is to reduce low-frequency artifacts such as wind noise and other pop-like sounds caused by non-optimum microphone placement during the recording process. In general, the volume function is used in the V/U/S detector as a relatively wide-band integrator that calculates the approximate energy in each frame. The role of this integrator is discussed in the next section.

A graph of the frequency response of $H_i(z)$ before and after the hypothetical highpass filter for one pitch period of the vowel portion of the word "sue" spoken by a male speaker is shown in Figure A9.3. As described in Equation (A9.4.2.1), the single-sided frequency response of $H_i(z)$ is evaluated at 256 equally spaced points around the upper half of the unit circle in the z-plane. Although it is not shown on the graph, the value of V(i) for the pitch period analyzed in Figure A9.3(*b*) is 70.62 dB. It is seen from the graph that this is approximately the average level of the frequency response.

Note also that unlike the other feature detectors, median filtering is not performed on the V/U/S volume function. This is to ensure that any short-term energy fluctuations, such as those produced by stops, are not inadvertently smoothed.

A9.4.3 Fixed Thresholds and Feature Scores

Each feature detection algorithm calculates a feature score to indicate the presence of the corresponding acoustic feature in a given frame of speech. The feature score is typically continuous over the range [0, 1], although there are exceptions (several of the feature scores are discrete, either binary or ternary). In general, the feature score is calculated by comparing the value of the volume function (or the ratio of two volume functions) with one or more fixed thresholds. The values of the thresholds are determined empirically by trial and error during the analysis of approximately 100 words of the Diagnostic Rhyme Test (DRT) spoken by two male and one female speakers (Voiers, 1983). In some

cases, initial estimates for the thresholds are taken from literature (sources are listed in discussion of the specific algorithms), and the thresholds are then "fine tuned" using the DRT speech data. The empirical determination of these thresholds constitutes a type of "learning" phase in the algorithm development. This contrasts with one of the initial goals of the automatic segmentation and labeling process, which is to not require "training" of the algorithms. However, given the nature and variability of the speech signal, in hind sight it now appears that to create a set of reliable segmentation and labeling algorithms based upon frequency distributions (i.e., volume functions) that some type of training, or parameter "tuning," is required.

The advantages and disadvantages of training are obvious. If the training data does not accurately represent the set of intended users, the algorithms will not function as expected in practice. If the training data completely represents the set of intended users, the algorithms will work efficiently and accurately. Since the topic of training of speech recognition algorithms is beyond the scope of this software toolbox, it will be accepted that training is mandatory, regardless of the particular algorithm.

In general, if two thresholds are used, the feature score for each frame is determined by

$$\text{Feature\_Score(i)} = \begin{cases} 1, & \text{if Vol\_Fcn(i)} \geq T_{upper} \\ 0, & \text{if Vol\_Fcn(i)} < T_{lower} \\ \dfrac{\text{Vol\_Fcn(i)} - T_{lower}}{T_{upper} - T_{lower}}, & \text{if } T_{lower} \leq \text{Vol\_Fcn(i)} < T_{upper} \end{cases} \qquad \text{(A9.4.3.1)}$$

for a feature score that increases as the volume function increases. If the feature score decreases as the volume function increases, the feature score is given by

$$\text{Feature\_Score(i)} = \begin{cases} 0, & \text{if Vol\_Fcn(i)} \geq T_{upper} \\ 1, & \text{if Vol\_Fcn(i)} < T_{lower} \\ \dfrac{T_{upper} - \text{Vol\_Fcn(i)}}{T_{upper} - T_{lower}}, & \text{if } T_{lower} \leq \text{Vol\_Fcn(i)} < T_{upper} \end{cases} \qquad \text{(A9.4.3.2)}$$

For both equations, i is the current frame index, T_{lower} is the fixed lower threshold, T_{upper} is the fixed upper threshold, and Vol_Fcn(i) is the volume function (or the ratio of two volume functions) for the current frame. Both Equations (A9.4.3.1) and (A9.4.3.2) are used in practice. If only one threshold is used to calculate a binary feature score (zero or one), then either Equation (A9.4.3.1) or (A9.4.3.2) is used with $T_{lower} = T_{upper}$.

The original LPC analysis described in Section A9.4.1 distinguishes only between voiced and non-voiced frames. The non-voiced frames denoted by Hu (1993) as "unvoiced" are either unvoiced or silent. The procedure used in the V/U/S detector to classify non-voiced frames as either unvoiced or silent is based upon a single volume function, the background noise power in the speech signal, and Equation (A9.4.3.1). First, the mean and the standard deviation of the background noise power, BNP, are calculated as

$$\text{BNP}_{mean} = \frac{1}{20}\sum_{n=1}^{20} p(n) \qquad \text{(A9.4.3.3)}$$

$$\text{BNP}_{std\,dev} = \sqrt{\frac{1}{20}\sum_{n=1}^{20}(p(n) - \text{BNP}_{mean})^2} \qquad \text{(A9.4.3.4)}$$

where p(n) is the frame power in decibels (dB). It is assumed that the first 100 msec (20 frames) of the speech signal are silence.

The V/U/S volume function for each non-voiced frame is then compared to a constant threshold, $T_{U/S}$, using Equation (A9.4.3.1) with $T_{lower} = T_{upper}$ and $T_{upper} = T_{U/S}$. The V/U/S feature score for each non-voiced frame is given as

$$\text{VUS\_Score(i)} = \begin{cases} 1, & \text{if } 20\ \log_{10}(V(i)) \geq T_{U/S} \\ 0, & \text{if } 20\ \log_{10}(V(i)) < T_{U/S} \end{cases} \qquad \text{(A9.4.3.5)}$$

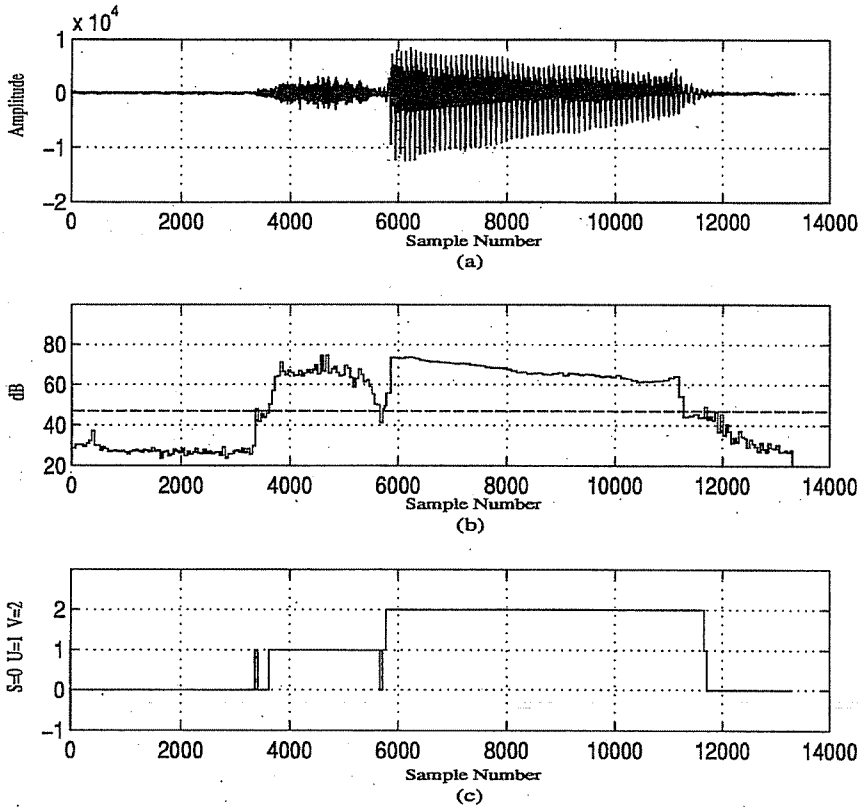

FIGURE A9.4 Voiced/unvoiced/silent classification using the volume function and a single fixed threshold for the non-voiced portion of the word sue. (*a*) Time-domain waveform. (*b*) Volume function and $T_{U/S}$ threshold (dashed line). (*c*) V/U/S score.

where V(i) is calculated from Equation (A9.4.2.1) with A = 17 and B = 255, and i is the index of the current frame. If VUS_Score(i) = 1, the frame is classified as unvoiced, and if VUS_Score(i) = 0, the frame is classified as silent. Note that this method only separates unvoiced from silent frames. The value of VUS_Score(i) is arbitrarily set equal to two for all voiced frames. As a result, the V/U/S feature score is different from many of the other feature scores in two ways: First, it spans the range [0, 2] instead of [0, 1]. Second, it can have only one of three discrete values, while most of the other feature scores are continuous.

It is found (empirically) that the best results are obtained when

$$T_{U/S} = BNP_{mean} + (k)(BNP_{std\,dev}) \qquad \text{(A9.4.3.6)}$$

where k = 2.0. Obviously, the value used for k is dependent upon the statistical properties of the background noise in the speech signal. However, the absolute level of the background noise is compensated for automatically since the value of BNP_{mean} is calculated before analysis of each word.

Figure A9.4 shows the V/U/S classification for the word "sue" spoken by a male speaker. Part (*a*) shows the time-domain speech waveform, part (*b*) shows the volume function (in dB) from (A9.4.2.1) and the fixed threshold, $T_{U/S}$, and part (*c*) shows the resulting V/U/S score.

A9.4.4 Automatic Correction Rules

The feature scores produced by Equation (A9.4.3.1) [or Equation (A9.4.3.2)] sometimes require additional processing to help eliminate false detection of features. The processing is accomplished by the application of pattern recognition rules. These rules are used sparingly, and counteract specific, regularly occurring, incorrect classifications of feature types. They are developed empirically on an algorithm-by-algorithm basis.

In the case of the V/U/S detector, the unprocessed, or "raw," V/U/S score produced by Equation (A9.4.3.5) has the undesirable effect of sometimes oscillating between states during word onsets

TABLE A9.1 Rules to Modify Initial Voiced/Unvoiced/Silent (V/U/S) Results[a]

Rule number	Initial pattern	Requirements for modification	Final pattern
1	VUS	length V > 100.0 msec, length U < 25.1 msec	VVS
2	xSy	length S < 10.1 msec	xxy
3	SUV	length U < 7.5 msec	SSV
4	xUy (except SUV)	length U < 10.0 msec	xyy (if x = S), else xxy

[a] The symbols x and y denote an arbitrary segment type.

and offsets, as well as during transitions between unvoiced and voiced regions. Since this does not accurately model the human speech process, rules are applied to smooth the V/U/S score, or "track." Table A9.1 lists the rules.

The first rule eliminates an incorrect unvoiced classification at the end of a long voiced segment. Since the energy often decreases quickly during the last few glottal cycles of a voiced-silent transition, the classification algorithm sometimes (incorrectly) labels these pitch periods as unvoiced. The second rule smoothes out momentary "drop outs" that occur when the signal level drops below the $T_{U/S}$ threshold for a brief period of time. The third rule is similar to the first rule, except that it smoothes out the beginning of the segment, instead of the end. It reclassifies the very short, low energy frame at the beginning of a silence-unvoiced-voiced transition from unvoiced to silent. The fourth rule smoothes out momentary unvoiced segments of very short duration. This rule does not eliminate the short noise bursts exhibited by plosives, since it only acts if the unvoiced segment is less than 10 msec, which is far shorter than the average duration of the plosive burst (Klatt, 1979; Umeda, 1977).

Figure A9.5 shows the V/U/S score for the word "sue" spoken by a male speaker before and after the application of the pattern recognition rules. The rules reduce the number of non-silent segments from five to two.

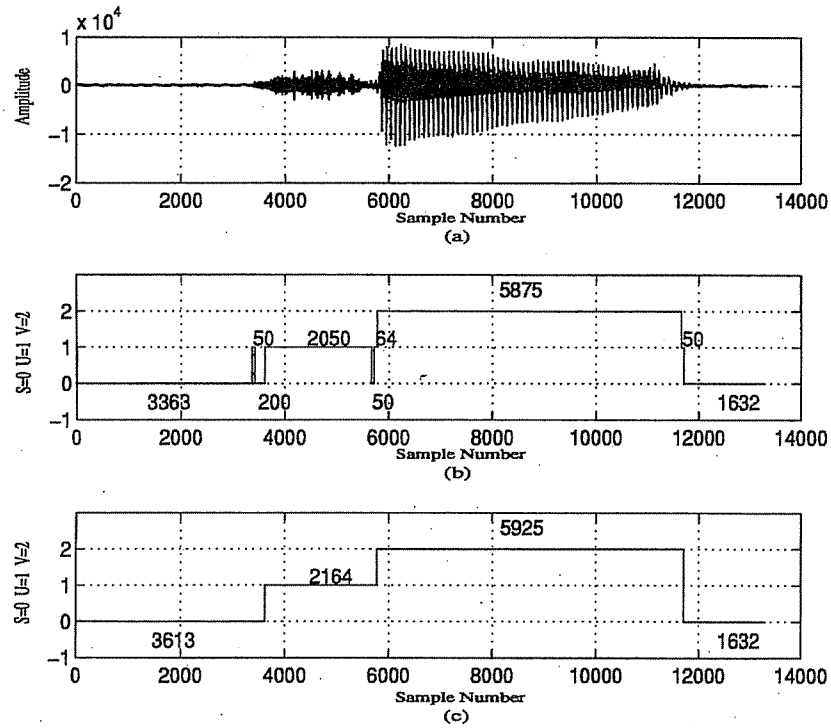

FIGURE A9.5 Voiced/unvoiced/silent (V/U/S) classification for the word sue. Segment durations are indicated in number of samples. (*a*) Time-domain waveform. (*b*) Before application of pattern recognition rules. (*c*) After application of pattern recognition rules.

A9.4.5 Summary

Each feature detection algorithm consists of a similar sequence of processing stages. In general, the first stage calculates one or more volume functions. The volume function (or ratio of volume functions) is smoothed by a median filter to remove any short-term fluctuations. The second stage calculates a feature score, typically over the range [0, 1], by comparing the volume function with one or two fixed thresholds. The third stage applies pattern recognition rules to correct for any known deficiencies in the algorithms.

The following section details the individual feature detection algorithms. Each of the detectors in Figure A9.2 is described, and any differences from the V/U/S detector are discussed.

A9.5 FEATURE DETECTION ALGORITHMS—DETAILED DESCRIPTIONS

This section gives the details of the feature detection algorithms. The algorithms are, for the most part, similar in form to the V/U/S feature detection algorithms described in Section A9.4.

A9.5.1 Sonorant Detection

To be classified as sonorant, a frame must be voiced and must also have a high ratio of low-frequency to high-frequency energy. The group of sonorants typically include vowels, voice bars, nasals, and semivowels. The non-sonorants include unvoiced fricatives, unvoiced stops, and strong, voiced fricatives. Weak voiced fricatives are classified as sonorant if they have a relatively large proportion of low-frequency energy (Weinstein et al., 1975).

The volume function from Equation (A9.4.2.1) is calculated for each frame with G = 1, A = 5, and B = 46. This is termed the low-frequency volume function, or LFV. The LFV is equivalent to a bandpass filter from 98 Hz to 898 Hz. A second volume function from Equation (A9.4.2.1) is calculated for each frame with G = 1, A = 189, and B = 255. This is termed the high-frequency volume function, or HFV. The HFV is equivalent to a bandpass filter from 3691 Hz to 5000 Hz. The sonorant ratio, R(i), is calculated for each frame as

$$R(i) = \frac{LFV(i)}{HFV(i)} \tag{A9.5.1.1}$$

where i is the index of the current frame.

The sonorant ratio is then smoothed by a fifth-order median filter. The smoothed sonorant ratio is compared to a threshold, T_{son}, and a binary (zero or one) sonorant score, SS(i), is calculated for each frame as

$$SS(i) = \begin{cases} 0, & \text{if } R(i) < T_{son} \\ 1, & \text{if } R(i) \geq T_{son} \end{cases} \tag{A9.5.1.2}$$

where i is the frame index and $T_{son} = 10$. The threshold T_{son} is determined empirically.

Figure A9.6 shows the sonorant detection results for the word "sue" spoken by a male speaker. The threshold $T_{son} = 10$ is shown as a dashed line in Figure A9.6(*b*). Note that the sonorant ratio is nearly zero for the entire duration of the /s/.

A9.5.2 Vowel Detection

Vowel detection is accomplished in a manner similar to that for sonorant detection. A LFV function from Equation (A9.4.2.1) is calculated with G = 1, A = 1, and B = 51. The LFV is equivalent to a bandpass filter from 20 Hz to 996 Hz. A HFV function from Equation (A9.4.2.1) is calculated for G = 1, A = 52, and B = 255. The HFV is equivalent to a bandpass filter from 1016 Hz to 5000 Hz.

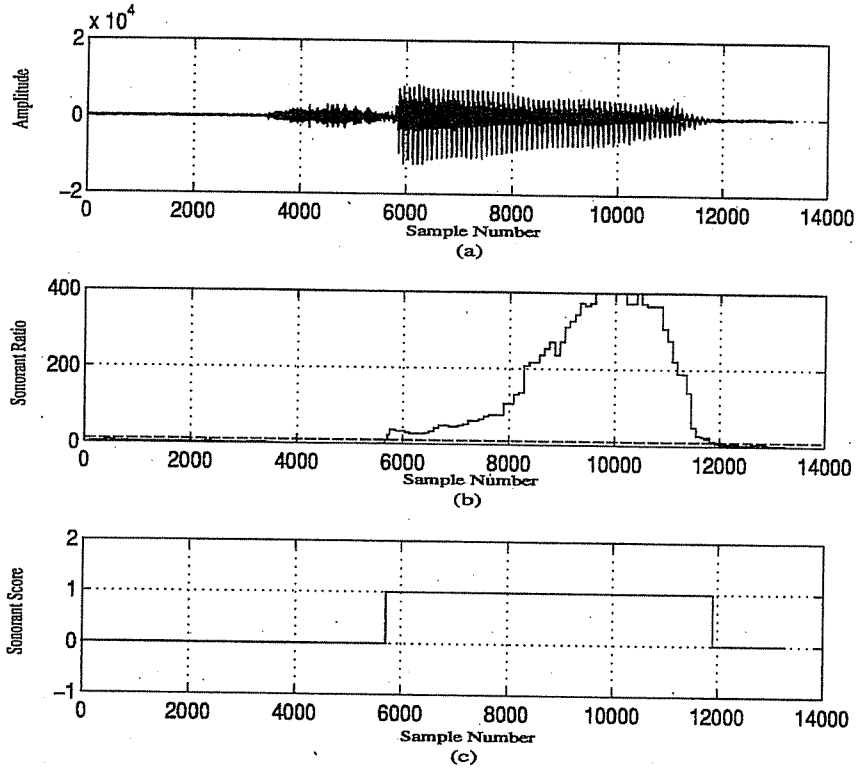

FIGURE A9.6 Sonorant detection for the word sue. (*a*) Time-domain waveform. (*b*) Sonorant ratio and threshold. (*c*) Sonorant score.

A vowel ratio, VWL(i), is calculated for each frame by

$$\text{VWL(i)} = \frac{\text{LFV(i)}}{\text{HFV(i)}} \tag{A9.5.2.1}$$

where i is the frame index. The vowel ratio is then smoothed with a fifth-order median filter. A vowel score, VWLS(i), within the continuous range [0, 1] is calculated for each frame by comparing the smoothed vowel ratio with two thresholds. The score is given by

$$\text{VWLS(i)} = \begin{cases} 0, & \text{if VWL(i)} \geq \text{T}_{\text{upper}} \\ 1, & \text{if VWL(i)} < \text{T}_{\text{lower}} \\ \dfrac{\text{T}_{\text{upper}} - \text{VWL(i)}}{\text{T}_{\text{upper}} - \text{T}_{\text{lower}}}, & \text{if T}_{\text{lower}} \leq \text{VWL(i)} < \text{T}_{\text{upper}} \end{cases} \tag{A9.5.2.2}$$

where $\text{T}_{\text{upper}} = 18$ and $\text{T}_{\text{lower}} = 8$. The two thresholds are determined empirically.

In a final processing stage, the vowel score is automatically set to zero for all frames in any vowel segment that is 150 samples (15 msec) or less in length. This helps to reduce false vowel detection.

Figure A9.7 shows the vowel detection results for the word "said" spoken by a male speaker. The two thresholds are shown as dashed lines in Figure A9.7(*b*). Note that the voiced /d/ exhibits a high vowel score for a short duration. Since there is no voiced stop segment category in this system, voiced stops are typically classified as either a vowel-unvoiced stop sequence, or a vowel-unvoiced fricative sequence.

A9.5.3 Voiced Consonant Detection

Voiced consonant detection is accomplished in a manner almost identical to vowel detection. A LFV function from Equation (A9.4.2.1) is calculated with G = 1, A = 1, and B = 51. The LFV is equivalent to a bandpass filter from 20 Hz to 996 Hz. A HFV function from Equation (A9.4.2.1) is

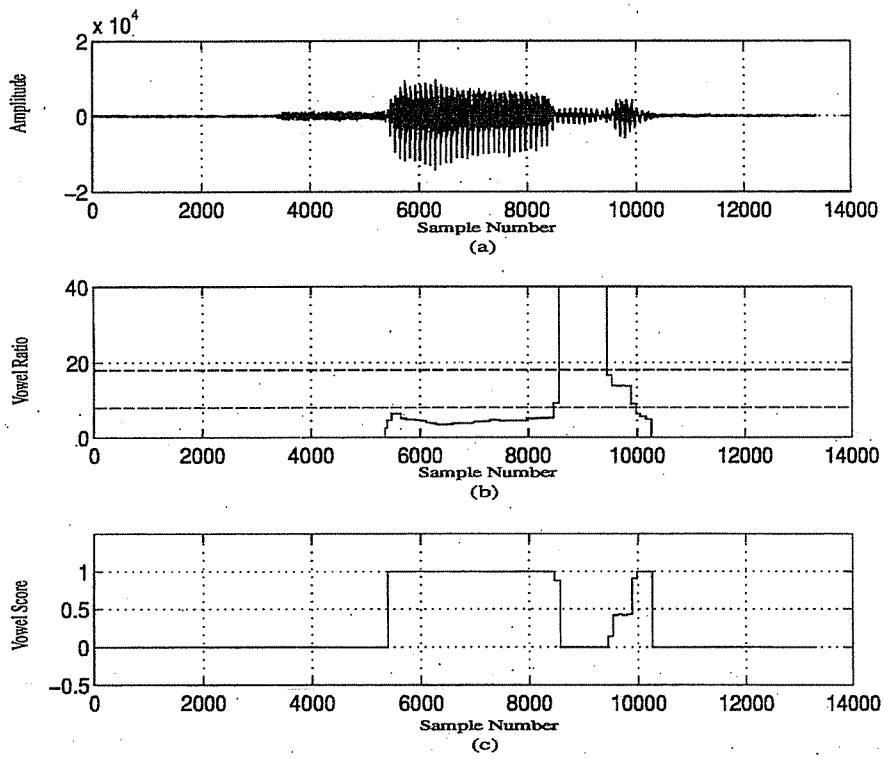

FIGURE A9.7 Vowel detection for the word said. (*a*) Time-domain waveform. (*b*) Vowel ratio and thresholds. (*c*) Vowel score.

calculated for G = 1, A = 52, and B = 255. The HFV is equivalent to a bandpass filter from 1016 Hz to 5000 Hz. These filter values are the same as those used for vowel detection. A voiced consonant ratio, VC(i), is calculated for each frame as

$$\mathrm{VC(i)} = \frac{\mathrm{LVF(i)}}{\mathrm{HVF(i)}} \tag{A9.5.3.1}$$

where i is the frame index. The voiced consonant ratio is then smoothed with a fifth-order median filter. A voiced consonant score, VCS(i), within the continuous range [0, 1] is calculated for each frame by comparing the smoothed voiced consonant ratio with two thresholds. The score is given by

$$\mathrm{VCS(i)} = \begin{cases} 1, & \text{if } \mathrm{VC(i)} \geq \mathrm{T_{upper}} \\ 0, & \text{if } \mathrm{VC(i)} < \mathrm{T_{lower}} \\ \dfrac{\mathrm{VC(i)} - \mathrm{T_{lower}}}{\mathrm{T_{upper}} - \mathrm{T_{lower}}}, & \text{if } \mathrm{T_{lower}} \leq \mathrm{VC(i)} < \mathrm{T_{upper}} \end{cases} \tag{A9.5.3.2}$$

where $\mathrm{T_{upper}} = 18$ and $\mathrm{T_{lower}} = 8$. The thresholds are determined empirically.

Note that VCS can be calculated during the VWLS calculation for each frame, since VCS = 1 − VWLS, provided that the value of VWLS is used before the short segment (<15 msec) vowel detection and elimination of Section A9.5.2 is done. This is because the same filters and thresholds are used for both vowel and voiced consonant detection.

However, calculating VCS directly from VWLS eliminates the possibility of future experiments with the filter characteristics and thresholds for voiced consonant detection independent of the vowel detection algorithm.

Figure A9.8 shows the voiced consonant detection results for the word "said" spoken by a male speaker. The two thresholds are shown as dashed lines in Figure A9.8(*b*). The voice bar that occurs before the release of the /d/ is clearly classified as a voiced consonant. However, the algorithm assigns a score of slightly greater than 0.5 to the initial portion of the release of the /d/. This shows the difficulty of detecting voiced stops using a single acoustic feature.

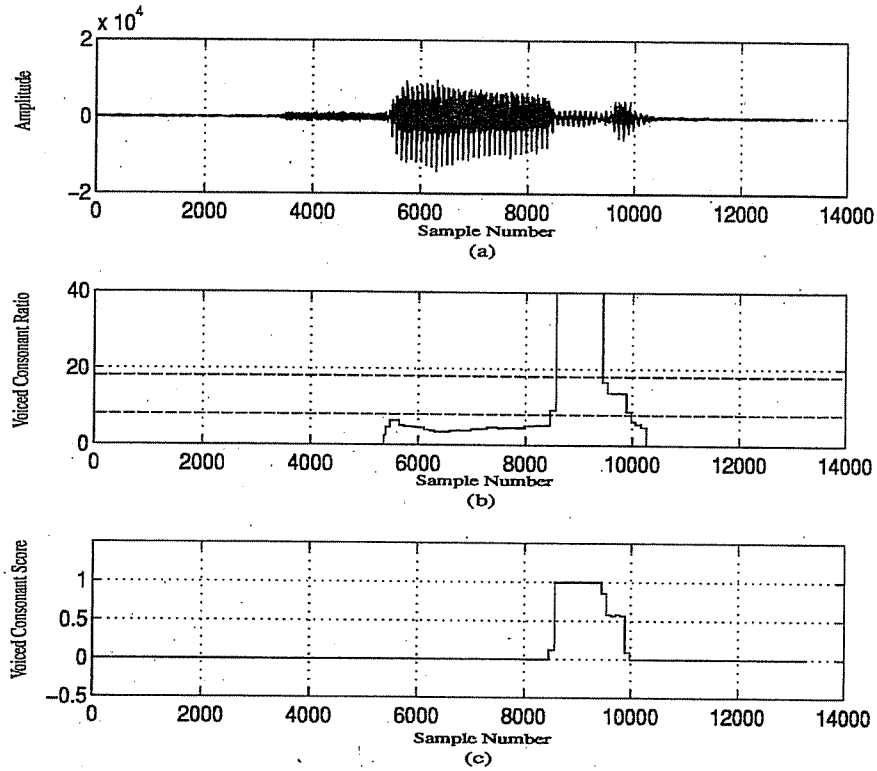

FIGURE A9.8 Voiced consonant detection for the word said. (*a*) Time-domain waveform. (*b*) Voiced consonant ratio and thresholds. (*c*) Voiced consonant score.

A9.5.4 Voice Bar Detection

Voice bar detection is accomplished in a manner similar to both vowel and voiced consonant detection. A LFV function from Equation (A9.4.2.1) is calculated with G = 1, A = 1, and B = 33. The LFV is equivalent to a bandpass filter from 20 Hz to 645 Hz. A HFV function from Equation (A9.4.2.1) is calculated for G = 1, A = 34, and B = 255. The HFV is equivalent to a bandpass filter from 664 Hz to 5000 Hz. A voice bar ratio, VB(i), is calculated for each frame as

$$\mathrm{VB(i)} = \frac{\mathrm{LFV(i)}}{\mathrm{HFV(i)}} \tag{A9.5.4.1}$$

where i is the frame index. The voice bar ratio is then smoothed with a fifth-order median filter. A voice bar score, VBS(i), within the continuous range [0, 1] is calculated for each frame by comparing the smoothed voice bar ratio with two thresholds. The score is given by

$$\mathrm{VBS(i)} = \begin{cases} 1, & \text{if } \mathrm{VB(i)} \geq \mathrm{T_{upper}} \\ 0, & \text{if } \mathrm{VB(i)} < \mathrm{T_{lower}} \\ \dfrac{\mathrm{VB(i)} - \mathrm{T_{lower}}}{\mathrm{T_{upper}} - \mathrm{T_{lower}}}, & \text{if } \mathrm{T_{lower}} \leq \mathrm{VB(i)} < \mathrm{T_{upper}} \end{cases} \tag{A9.5.4.2}$$

where $\mathrm{T_{upper}} = 30$ and $\mathrm{T_{lower}} = 10$. The thresholds are determined empirically.

In a final processing stage, the voice bar score is automatically set to zero for all frames in any voice bar segment that is 300 samples (30 msec) or less in length. This helps to reduce false voice bar detection.

Figures A9.9 and A9.10 show voice bar detection for the words "said" and "bond," respectively, spoken by a male speaker. The two thresholds are shown as dashed lines in Figures A9.9(*b*) and A9.10(*b*). In Figure A9.9, the voice bar before the release of the /d/ is clearly detected. Figure A9.10 shows both the voice bar of the initial /b/, and the voice bar associated with the final /d/, for the word

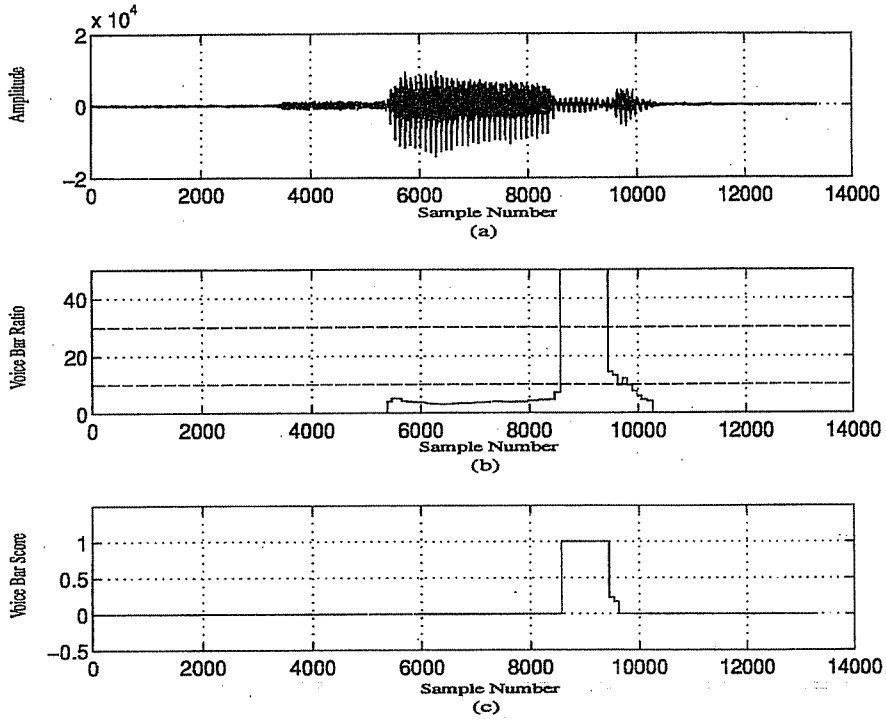

FIGURE A9.9 Voice bar detection for the word said. (*a*) Time-domain waveform. (*b*) Voice bar ratio and thresholds. (*c*) Voice bar score.

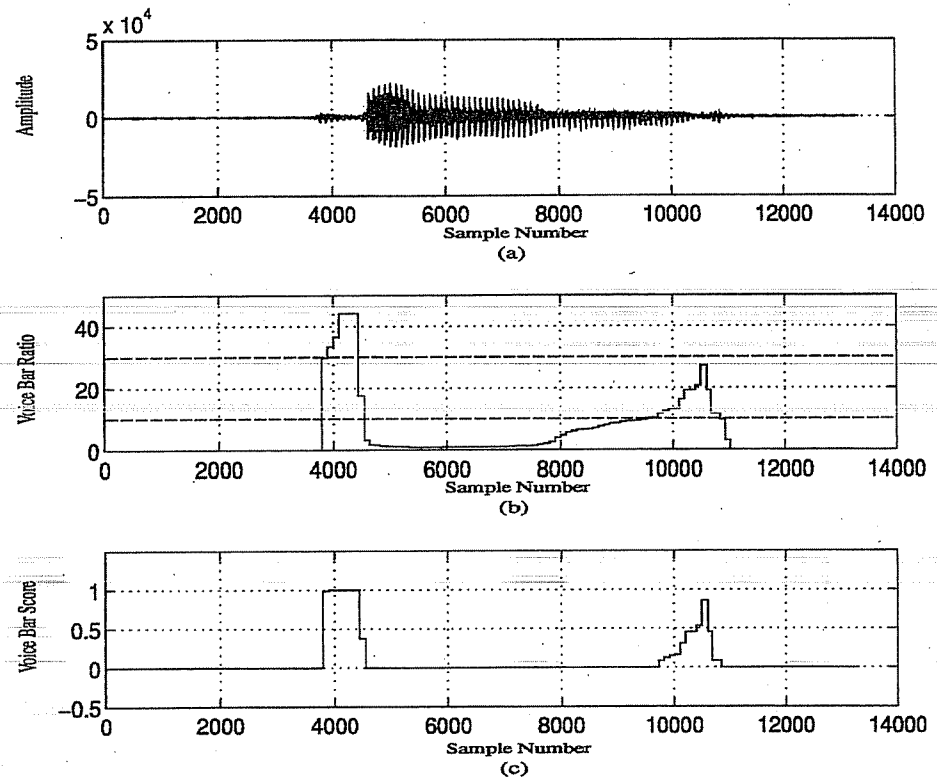

FIGURE A9.10 Voice bar detection for the word bond. (*a*) Time-domain waveform. (*b*) Voice bar ratio and thresholds. (*c*) Voice bar score.

"bond." Also note in Figures A9.9 and A9.10 that the voice bar associated with the final /d/ is much shorter in the word "bond" than in the word "said" due to the preceding nasal in "bond."

A9.5.5 Formant Tracking

Formant tracking is accomplished in a manner completely different from the typical detection process. The output of the algorithm is also different from the other feature detector outputs. Actually, the formant tracking algorithm is a "front-end" processor for the nasal detection algorithm, since the nasal detector does not use volume functions, but rather a ratio of the amplitudes of the first two formant frequencies to determine the nasal feature score.

Formant tracking is accomplished by an algorithm developed by McCandless (1974). Only a brief description is given, since the algorithm is documented elsewhere. Only the voiced portions of the speech signal are analyzed to estimate the formant tracks.

The algorithm attempts to find a best match between the peaks of the frequency response obtained from the filter produced from the LPC coefficients, and estimates for the first four formant frequencies. The amplitudes of the first four formant peaks are also estimated. Initially, the estimates for the four formant frequencies are set to values that are typical for the male voice (or the female voice, if female speech is being analyzed). For the male voice, the initial estimates are $F_1 = 320$ Hz, $F_2 = 1440$ Hz, $F_3 = 2760$ Hz, and $F_4 = 3200$ Hz. For the female voice, the initial estimates are $F_1 = 480$ Hz, $F_2 = 1760$ Hz, $F_3 = 3200$ Hz, and $F_4 = 3520$ Hz. The algorithm matches each peak of the frequency response of the LPC filter with the closest formant frequency estimate. The estimates for the formant frequencies are updated after each frame of speech is processed, provided that a match has been made.

In any given frame, if there is no match between the LPC filter peaks and the formant frequency estimates, the algorithm attempts to increase spectral resolution by iteratively evaluating the "frequency response" of the LPC filter on a circle in the z-plane with a radius less than one. This is done by evaluating the z-transform of the LPC filter with $z = re^{j\theta}$, where r denotes the radius of the circle. The initial radius value is unity, and is decreased by 0.004 during each iteration until a match is obtained, or until the radius is less than 0.88. This procedure is able to resolve two closely spaced poles if they are relatively close to the unit circle. If the radius is reduced to less than 0.88 and a match is still not found for all of the formant frequencies in the frame, the algorithm re-evaluates the matches it has made for the frame. During the re-evaluation, the algorithm either changes the matches it has made, or assigns a zero value to the formant frequency in question for that particular frame.

The final results are smoothed by checking each of the first three formant frequency tracks individually for any zero values (this is not a part of the McCandless algorithm). If the formant frequency is zero for either one or two (consecutive) frames, the frequency and amplitude are linearly interpolated and the zero values are removed.

The first three estimated formant frequency and estimated formant amplitude tracks for the voiced portion of the word "meat" spoken by a male speaker are shown in Figure A9.11. Although only the first two formant tracks are used in this system, the first three formants are retained for future expansion of the system.

A9.5.6 Nasal Detection

Nasal detection is accomplished by comparing the estimated amplitudes of the first two formants obtained in the McCandless formant tracker. A nasal ratio, N(i), is calculated for each frame as

$$N(i) = \frac{A2(i)}{A1(i)} \qquad \text{(A9.5.6.1)}$$

where A1(i) is the estimated first formant amplitude, A2(i) is the estimated second formant amplitude, and i is the current frame index (Mermelstein, 1977). The nasal ratio is then smoothed by a fifth-order median filter. A nasal score, NS(i), within the continuous range [0, 1] is calculated for each frame by

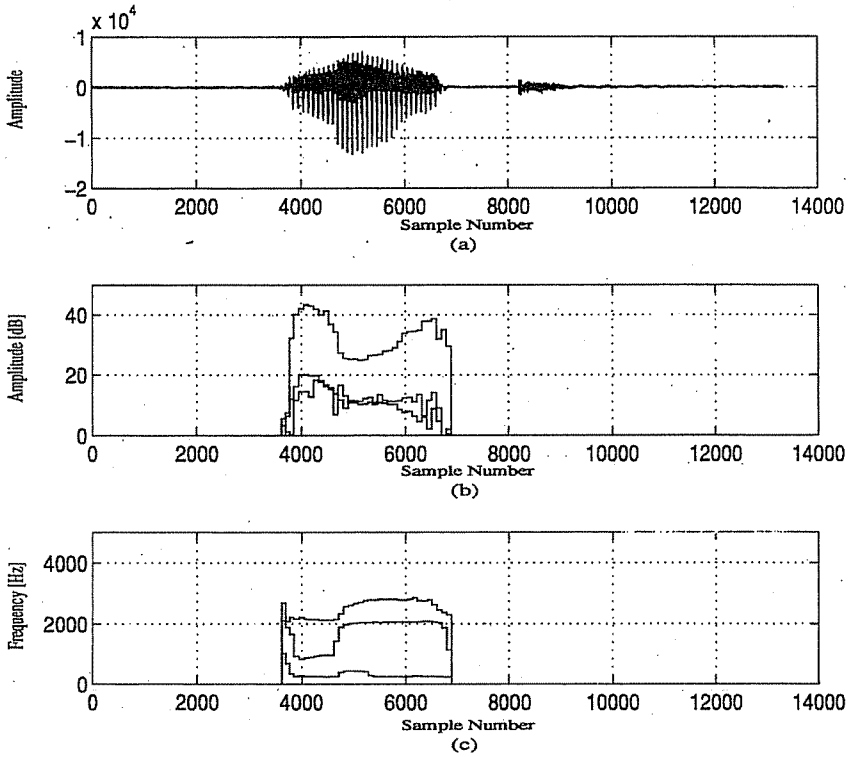

FIGURE A9.11 McCandless formant tracker for the word meat. (*a*) Time-domain waveform. (*b*) Amplitudes of the first three formants. (*c*) Center frequencies of the first three formants.

comparing the smoothed nasal ratio with two thresholds. The score is given by

$$NS(i) = \begin{cases} 0, & \text{if } N(i) \geq T_{upper} \\ 1, & \text{if } N(i) < T_{lower} \\ \dfrac{T_{upper} - N(i)}{T_{upper} - T_{lower}}, & \text{if } T_{lower} \leq N(i) < T_{upper} \end{cases} \qquad \text{(A9.5.6.2)}$$

where $T_{upper} = 0.20$ and $T_{lower} = 0.05$. The thresholds are determined empirically.

Additional processing is achieved by applying pattern recognition rules to distinguish nasals from other segment types. First, if a frame has a voice bar score greater than 0.75, the nasal score for that frame is set to zero. This is because strong voice bars typically exhibit strong nasal scores. The opposite, however, is not true. The nasal score is then set to zero if the frame has a zero sonorant score. This is done to prevent non-sonorant frames from being classified as nasal. Finally, the nasal score for all frames in any continuous nasal segment that is less than 25 msec in length are set to zero.

Figure A9.12 shows nasal detection for the word "bond" spoken by a male speaker. The two thresholds are shown as dashed lines in Figure A9.12(*b*). Note that the nasal score rises slowly after the transition from the vowel to the nasal. This shows that the formant-amplitude-based algorithm is not as accurate as the other feature detection algorithms that are based upon the short-term frequency response. Still, the algorithm is able to correctly identify the nasal region.

A9.5.7 Semivowel Detection

Semivowel detection is based on a method developed by Espy-Wilson (1986). The algorithm deviates slightly from the standard detector, although it uses the volume functions from Equation (A9.4.2.1). A LFV function from Equation (A9.4.2.1) is calculated with G = 1, A = 1, and B = 20. The LFV is equivalent to a bandpass filter from 20 Hz to 391 Hz. A HFV function from Equation (A9.4.2.1) is calculated for G = 1, A = 21, and B = 50. The HFV is equivalent to a bandpass filter from 410 Hz

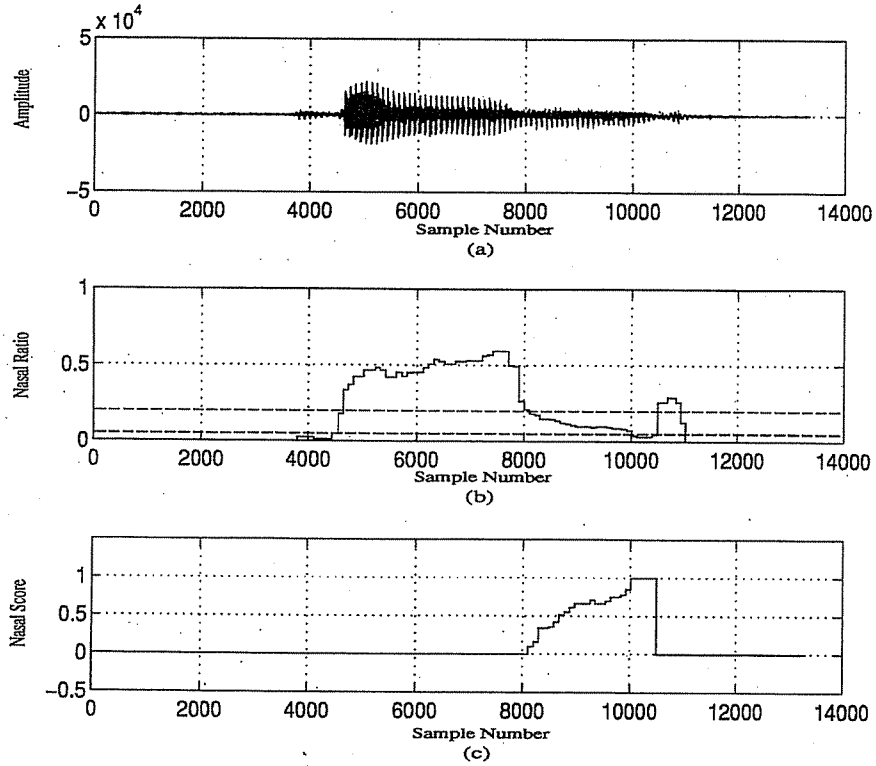

FIGURE A9.12 Nasal detection for the word bond. (*a*) Time-domain waveform. (*b*) Nasal ratio and thresholds. (*c*) Nasal score.

to 977 Hz. A murmur ratio, MUR(i), is calculated for each frame as

$$\mathrm{MUR(i)} = \frac{\mathrm{LFV(i)}}{\mathrm{HVF(i)}} \tag{A9.5.7.1}$$

The murmur ratio is then smoothed with a fifth-order median filter. A murmur score, MS(i), within the continuous range [0, 1] is calculated for each frame by comparing the smoothed murmur ratio with two thresholds. The score is given by

$$\mathrm{MS(i)} = \begin{cases} 1, & \text{if } \mathrm{MUR(i)} \geq \mathrm{T_{upper}} \\ 0, & \text{if } \mathrm{MUR(i)} < \mathrm{T_{lower}} \\ \dfrac{\mathrm{MUR(i)} - \mathrm{T_{lower}}}{\mathrm{T_{upper}} - \mathrm{T_{lower}}}, & \text{if } \mathrm{T_{lower}} \leq \mathrm{MUR(i)} < \mathrm{T_{upper}} \end{cases} \tag{A9.5.7.2}$$

where $\mathrm{T_{upper}} = 12$ and $\mathrm{T_{lower}} = 4$. The thresholds are determined empirically. The semivowel score, SVS(i), is then calculated for each frame as

$$\mathrm{SVS(i)} = (1 - \mathrm{MS(i)})(1 - \mathrm{VBS(i)})\mathrm{VCS(i)} \tag{A9.5.7.3}$$

where i is the frame index, VBS(i) is the voice bar score from Section A9.5.4, and VCS(i) is the voiced consonant score from Section A9.5.3. The value of SVS is limited to the range [0, 1]. If SVS is greater than one, it is set to unity for the frame. Equation (A9.5.7.3) shows that if a frame has a high voiced consonant score, a low murmur score, and a low voice bar score, it will have a high semivowel score.

Additional processing is done to smooth SVS. If the frame has a nasal score greater than 0.5, SVS is set to zero for that frame. This is because some strong nasals get labeled as semivowels. In addition, the semivowel scores for all of the frames in any continuous semivowel segment less than 30 msec in duration are set to zero. This eliminates mislabeling of short segments that are not semivowels.

Figure A9.13 shows semivowel detection for the word "wield" spoken by a male speaker. The algorithm correctly detects the leading /w/ as well as the /l/ near the end of the word immediately

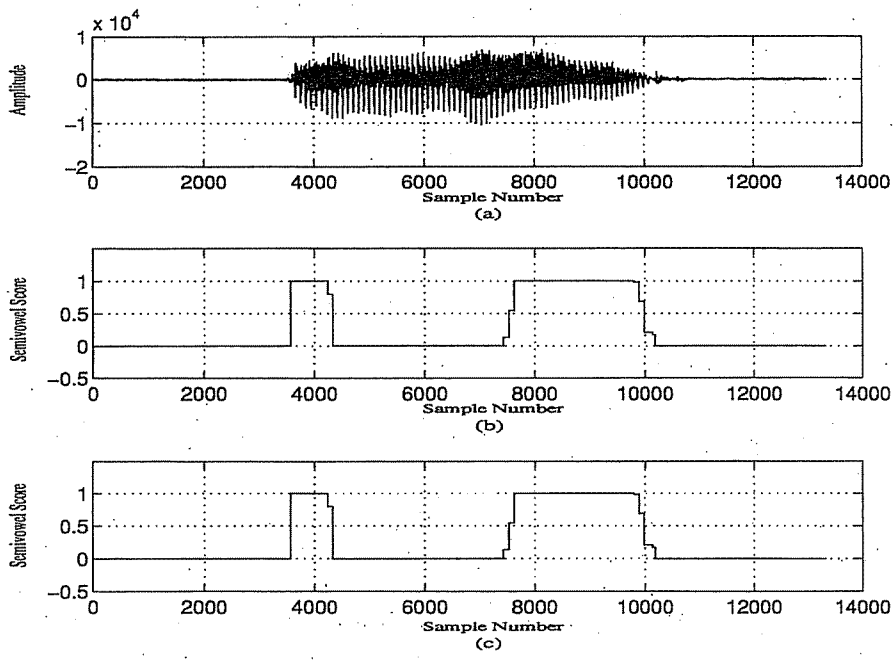

FIGURE A9.13 Semivowel detection for the word wield. (*a*) Time-domain waveform. (*b*) Semivowel score before post-processing. (*c*) Semivowel score after post-processing.

preceding the release of the plosive /d/. Note that listening reveals that the actual release in this example more closely resembles an unvoiced /t/ rather than a voiced /d/. Also note that the post-processing does not detect any nasal segments nor any semivowel segments less than 30 msec long. Therefore, the results for both before and after post-processing are the same.

A9.5.8 Voiced Fricative Detection

The voiced fricative detection algorithm deviates from the standard detector, although it does calculate feature scores from fixed thresholds. The first step in voiced fricative detection is to add preemphasis to the frequency response of the filter produced by the LPC coefficients. In other studies, the typical preemphasis method is to calculate the first difference of the sampled data waveform before calculating the LPC coefficients. In this system the first difference is not calculated before the LPC analysis, so the first difference function is approximated by a weighting function, W, in the frequency domain given by

$$W\left(e^{j\pi\frac{m}{256}}\right) = \frac{m}{256}, \quad \text{for } 0 \le m \le 255 \tag{A9.5.8.1}$$

The magnitude of the weighting function's frequency response is within 3 dB of the magnitude of the frequency response of a first-order differentiator for all frequencies in the filter passband. The preemphasized frequency response for frame i, $\hat{H}_i$, is

$$\hat{H}_i\left(e^{j\pi\frac{m}{256}}\right) = W\left(e^{j\pi\frac{m}{256}}\right) H_i\left(e^{j\pi\frac{m}{256}}\right), \quad \text{for } 0 \le m \le 255 \tag{A9.5.8.2}$$

where H_i is calculated from Equation (A9.4.2.2) for frame i with G = 1. The mean frequency of the preemphasized frequency response, MF(i), is then found for each frame as

$$MF(i) = \frac{1}{H_{total(i)}} \sum_{m=0}^{255} \left(\frac{m}{256} \frac{F_s}{2} \left| \hat{H}_i\left(e^{j\pi\frac{m}{256}}\right) \right| \right) \tag{A9.5.8.3}$$

where F_s = 10 kHz, and i is the frame index. $H_{total(i)}$ is given for frame i as

$$H_{total(i)} = \sum_{m=0}^{255} \left| \hat{H}_i\left(e^{j\pi\frac{m}{256}}\right) \right| \tag{A9.5.8.4}$$

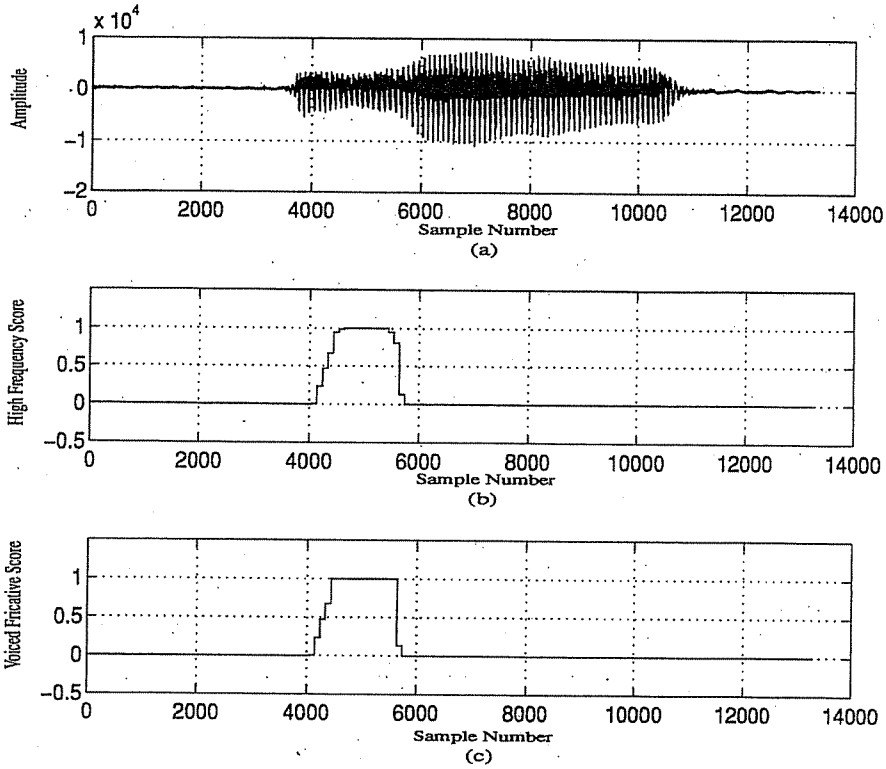

FIGURE A9.14 Voiced fricative detection for the word zoo. (*a*) Time-domain waveform. (*b*) High frequency score. (*c*) Voiced fricative score.

MF(i) is then smoothed by a third-order median filter. A high frequency score, HFS(i), is calculated for each frame as

$$\mathrm{HFS(i)} = \begin{cases} 1, & \text{if } \mathrm{MF(i)} \geq \mathrm{T_{upper}} \\ 0, & \text{if } \mathrm{MF(i)} < \mathrm{T_{lower}} \\ \dfrac{\mathrm{MF(i)} - \mathrm{T_{lower}}}{\mathrm{T_{upper}} - \mathrm{T_{lower}}}, & \text{if } \mathrm{T_{lower}} \leq \mathrm{MF(i)} < \mathrm{T_{upper}} \end{cases} \tag{A9.5.8.5}$$

where $\mathrm{T_{upper}} = 3200$ and $\mathrm{T_{lower}} = 2400$. The thresholds are determined empirically.

The voiced fricative score, VFS(i), is then calculated using HFS(i). If the frame is voiced and the sonorant score is zero, then VFS(i) = 1 for the frame. This is done because the frame is voiced and also has a relatively large amount of high-frequency energy (i.e., the frame is non-sonorant). If the frame is voiced and the sonorant score is 1, then VFS(i) = HFS(i) for the frame. In this case, the voiced fricative score depends solely upon the frame's high-frequency energy distribution. The final step is to set VFS(i) to zero for all frames in any voiced fricative segment less than 15 msec in duration. This is done to eliminate false detection of short segments.

Figure A9.14 shows the results of voiced fricative detection for the word "zoo" spoken by a male speaker. Examination of the spectrogram (not shown) reveals that there is little high-frequency energy at the beginning of the /z/. This explains why the algorithm does not classify the beginning of the /z/ as a voiced fricative.

A9.5.9 Unvoiced Stop and Fricative Detection

If a frame is classified as unvoiced, it is either an unvoiced stop or an unvoiced fricative. The algorithm used to distinguish between the two segment types differs from the standard feature detector, and uses both time-based and frequency-based parameters. First, the mean frequency is calculated for each frame from Equations (A9.5.8.1) through (A9.5.8.4). The mean frequency track is smoothed by a third-order median filter. The high frequency score is then calculated for each frame from

Equation (A9.5.8.5) with $T_{upper} = 3800$ and $T_{lower} = 2400$. The base-ten logarithm of the power, $P_{log\,10}$, is calculated from the initial LPC analysis results of each frame. Next, all adjacent unvoiced frames are grouped into segments. For example, the word "sit" has two unvoiced segments, /s/ and /t/, and each of these unvoiced segments is comprised of multiple, adjacent, unvoiced frames.

The slope of $P_{log\,10}$ of the initial twelve frames (60 msec) of each unvoiced segment is examined. If the segment is shorter than twelve frames, all of the frames are used. A first-order approximation (i.e., a straight line), $M_{seg}(j)$, of the slope of $P_{log\,10}$ is calculated for each segment j using the MATLAB function "polyfit." This is a least-squares fit. A segment slope score, $MS_{seg}(j)$, is computed for each segment from $M_{seg}(j)$ by

$$M_{seg}(j) = \begin{cases} 0, & \text{if } M_{seg}(j) \geq T_{upper} \\ 1, & \text{if } M_{seg}(j) < T_{lower} \\ \dfrac{T_{upper} - M_{seg}(j)}{T_{upper} - T_{lower}}, & \text{if } T_{lower} \leq M_{seg}(j) < T_{upper} \end{cases} \qquad \text{(A9.5.9.1)}$$

where $T_{upper} = 1.0$, $T_{lower} = -1.0$, and j is the index of the current segment. The "seg" subscript is included to draw attention to the fact that the slope score is calculated as a single value for the entire unvoiced segment. All of the frames in a given unvoiced segment are assigned the same MS_{seg} value. The frame slope score is denoted as MS(i). Thus, $MS(i) = MS_{seg}(j)$ for each frame i in segment j.

Calculation of the unvoiced stop score, USS(i), takes advantage of the fact that unvoiced stops are inherently shorter in duration than unvoiced fricatives (Cole and Cooper, 1975; Klatt, 1979; Umeda, 1977). The unvoiced stop score, USS(i), is given for each frame by

$$USS(i) = K_s MS(i) \qquad \text{(A9.5.9.2)}$$

where i is the current frame index, and K_s is given as

$$K_s = \begin{cases} G_s\left(1 + \dfrac{T_{stop} - L_j}{T_{stop}}\right), & L_j < T_{stop} \\ 1, & T_{stop} \leq L_j \leq T_{fric} \\ \left(1 + \dfrac{L_j - T_{fric}}{T_{fric}}\right)^{-1}, & L_j > T_{fric} \end{cases} \qquad \text{(A9.5.9.3)}$$

where $G_s = 8.0$, $T_{stop} = 50$ msec, $T_{fric} = 80$ msec, and L_j denotes the length of the unvoiced segment j (in milliseconds). The two thresholds and the gain, G_s, are all determined empirically. The term K_s acts as a duration-dependent scale factor that greatly amplifies the stop score for unvoiced segments less than 50 msec long. To a lesser degree, K_s also attenuates the stop score for unvoiced segments greater than 80 msec long. The unvoiced stop score, USS(i), is then limited to the range [0, 1]. If USS(i) is greater than one for a given frame, it is set to unity for the frame.

The final unvoiced fricative score, UFS(i), is calculated for each frame by

$$UFS(i) = HFS(i) \qquad \text{(A9.5.9.4)}$$

which is simply the high frequency score for the frame.

Figure A9.15 shows both the unvoiced stop and the unvoiced fricative scores for the word "pest" spoken by a male speaker. The /p/ is correctly detected as an unvoiced stop, the /s/ is correctly detected as an unvoiced fricative, and the /t/ has both a high unvoiced stop score and a high unvoiced fricative score. This is because of the high mean frequency of the /t/. Note in Figure A9.15(*b*), that the /t/ is incorrectly split into two different segments, which is an error caused by the V/U/S algorithm. Still, despite the V/U/S error, the /t/ has a greater unvoiced stop score than unvoiced fricative score, which is desirable.

A9.6 SPEECH SEGMENTATION

The algorithms described in the previous section focus primarily on the acoustic features associated with individual frames. However, in order to segment and label the speech into the segment categories defined in Section A9.2, the boundaries between the phoneme segments must be determined. This is

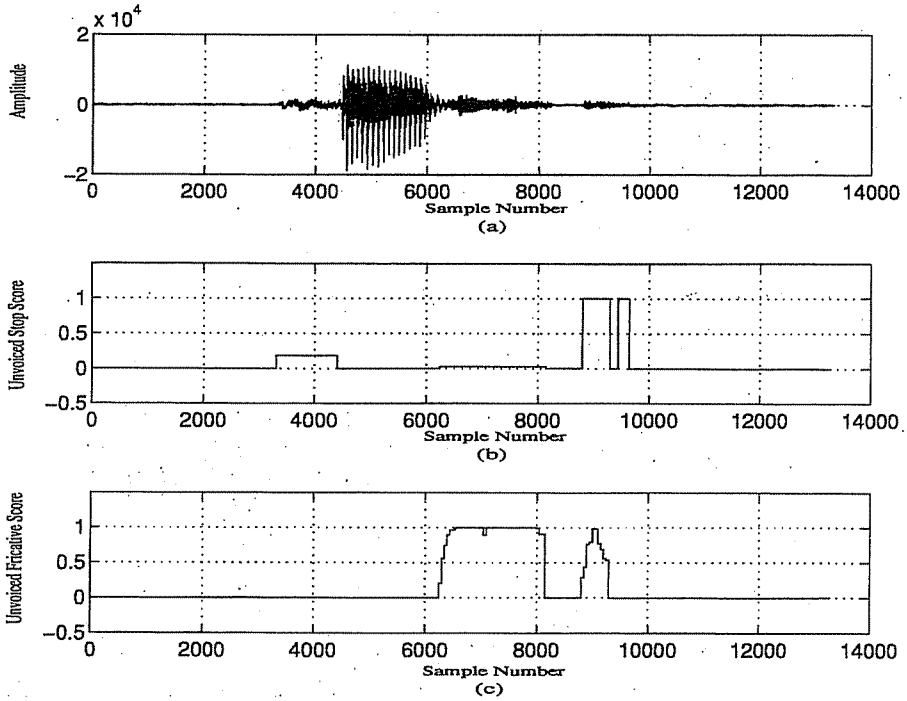

FIGURE A9.15 Unvoiced fricative and stop detection for the word pest. (*a*) Time-domain waveform. (*b*) Unvoiced stop score. (*c*) Unvoiced fricative score.

achieved with two algorithms that are described in the following subsections. The results from the two algorithms are then combined to determine the final segment boundaries and durations.

A9.6.1 Spectral-Based Boundary Detection and Segmentation

The first segmentation algorithm is based upon changes in the short-term frequency spectra of the speech signal. It uses an algorithm developed by Glass and Zue (1986) that measures the similarity between a current frame and its neighbors. To do so, the absolute value of the frequency response of the filter produced by the LPC coefficients from Equations (A9.4.2.2) and (A9.4.2.3) is calculated for each frame. A Euclidian distance measure, D(x, y), is defined as

$$D(x, y) = \sum_{m=0}^{255} \left\| H_x\left(e^{j\pi \frac{m}{256}}\right) | - | H_y\left(e^{j\pi \frac{m}{256}}\right) \right\| \qquad \text{(A9.6.1.1)}$$

where x is the current frame index, y is a past or future frame index, and $H_x(e^{j\pi \frac{m}{256}})$ is the single-sided frequency response evaluated for frame x at the points $\exp(j\pi m/256)$ for $0 \leq m \leq 255$. From Glass and Zue (1986), the decision strategy is to associate the current frame x with past frames if

$$\max(D(x, y)) < \min(D(x, v)) \quad x - 4 \leq y \leq x - 2, x + 2 \leq v \leq x + 4 \qquad \text{(A9.6.1.2)}$$

and to associate the current frame x with future frames if

$$\min(D(x, y)) > \max(D(x, v)) \quad x - 4 \leq y \leq x - 2, x + 2 \leq v \leq x + 4 \qquad \text{(A9.6.1.3)}$$

No association (a "don't care" state) is made if neither of these conditions is met. After each frame is associated with one of the three states, a segment boundary is determined to occur whenever the current frame's association changes from the past to the future. The location of the boundary is at the first sample of the frame where the transition to the future occurs. Post processing is also done to remove any boundaries that occur in the middle of silent segments.

Figure A9.16 shows the spectral-based boundary detection results for the word "wield" spoken by a male speaker. The algorithm marks boundaries at the beginning of the /w/, at the beginning of the relatively stationary portion of the vowel /i/, at the end of the transition from the vowel to the liquid /l/, and at the beginning of the release of the /d/. While all of these points are clearly seen from

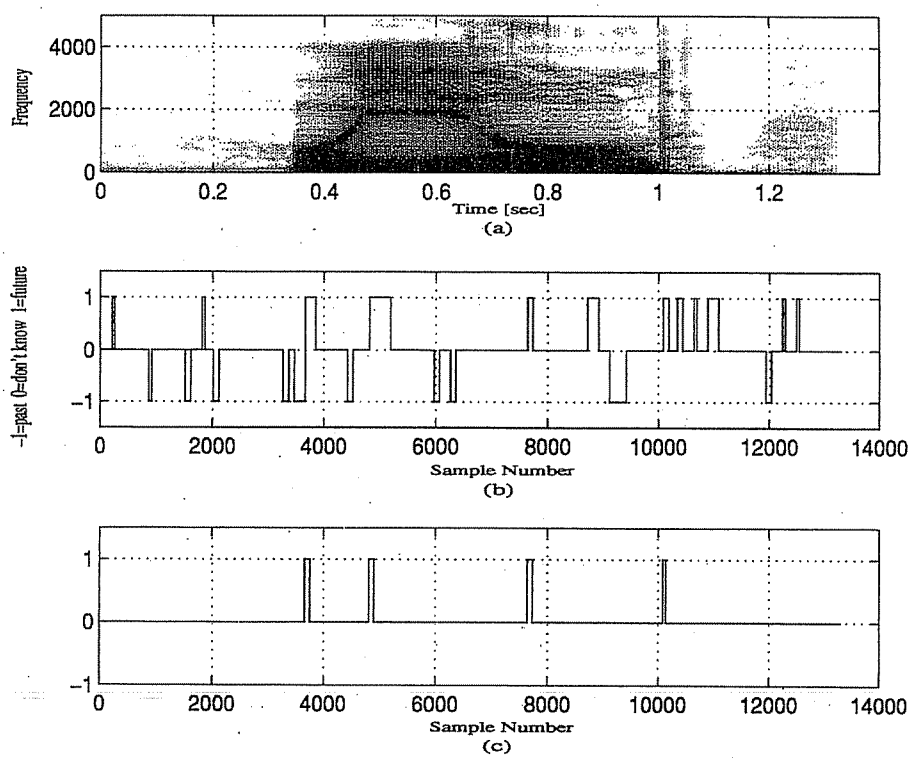

FIGURE A9.16 Spectral-based boundary detection for the word wield. (*a*) Spectrogram. (*b*) Frame association. (*c*) Spectral boundaries.

Figure A9.16(*a*) as dividing lines between different parts of the word, the locations may not always agree with the results obtained by manual parsing. For example, in the transition from the /i/ to the /l/, manual parsing might put the location of the transition at the middle of the second formant transition region, instead of at the end of the transition. However, this lack of agreement between automatic and manual segmentation results can often be attributed to the lack of a universally accepted method to manually specify the "correct" transition point between two phonemes.

A9.6.2 V/U/S Boundary Detection

The second segmentation algorithm is based upon the voiced/unvoiced/silent (V/U/S) feature detection algorithm results. The raw V/U/S results are processed using the pattern recognition rules listed in Table A9.1. Boundaries are determined to occur wherever transitions in the V/U/S track occur. The boundary is marked at the first sample of the first frame of the new segment.

Figure A9.17 shows the V/U/S boundary detection results for the word "wield" spoken by a male speaker. Note that as discussed in Section A9.5.7, the release of the /d/ more closely resembles a /t/, and is classified as unvoiced.

A9.6.3 Final Segmentation

Results from both the spectral segmentation and the V/U/S segmentation algorithms are used in the final segmentation process. All boundaries from the V/U/S algorithm are marked as boundaries in the final result. Any boundary from the spectral-based boundary detection algorithm that occurs in the middle of a voiced segment (as determined from the V/U/S results) is also marked as a boundary in the final result, provided that the boundary occurs at a frame that is located greater than two frames away from any V-U, U-V, V-S, or S-V boundary. This "two-frame rule" keeps the two algorithms from marking the same phoneme boundary as two separate, but closely spaced boundaries.

Figure A9.18 shows the final segmentation results for the word "wield" as spoken by a male speaker. Note that the boundaries in both Figure A9.18(*b*) and A9.18(*c*) are added together to create the final segmentation results that are shown in a later figure. The two spectral boundaries from

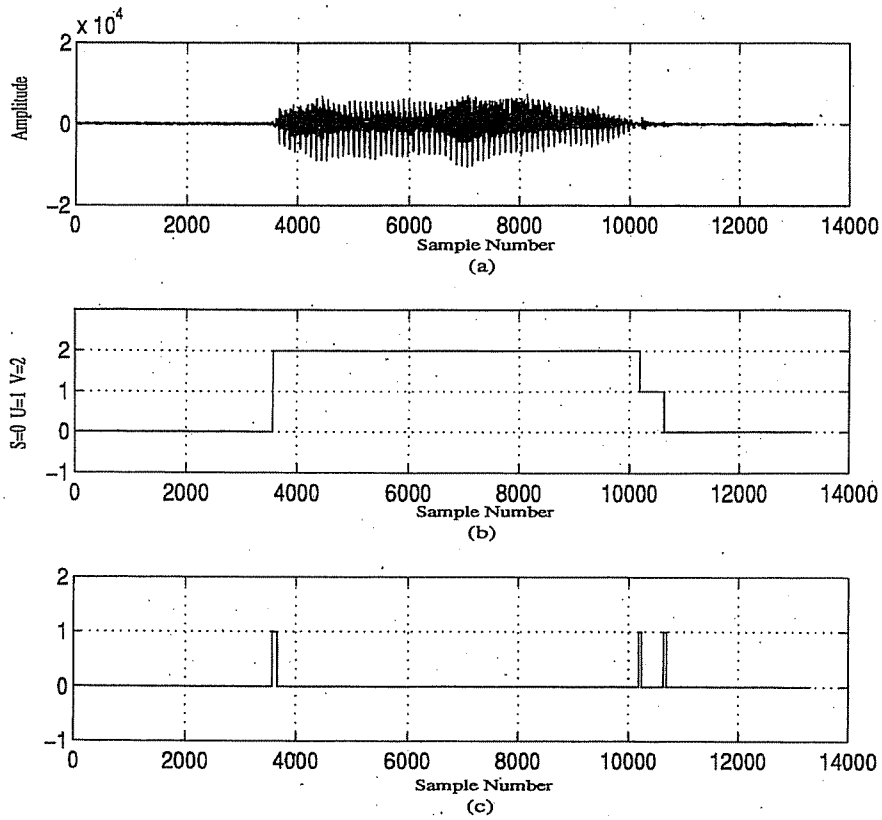

FIGURE A9.17 V/U/S boundary detection for the word wield. (*a*) Time-domain waveform. (*b*) V/U/S classification. (*c*) V/U/S boundaries.

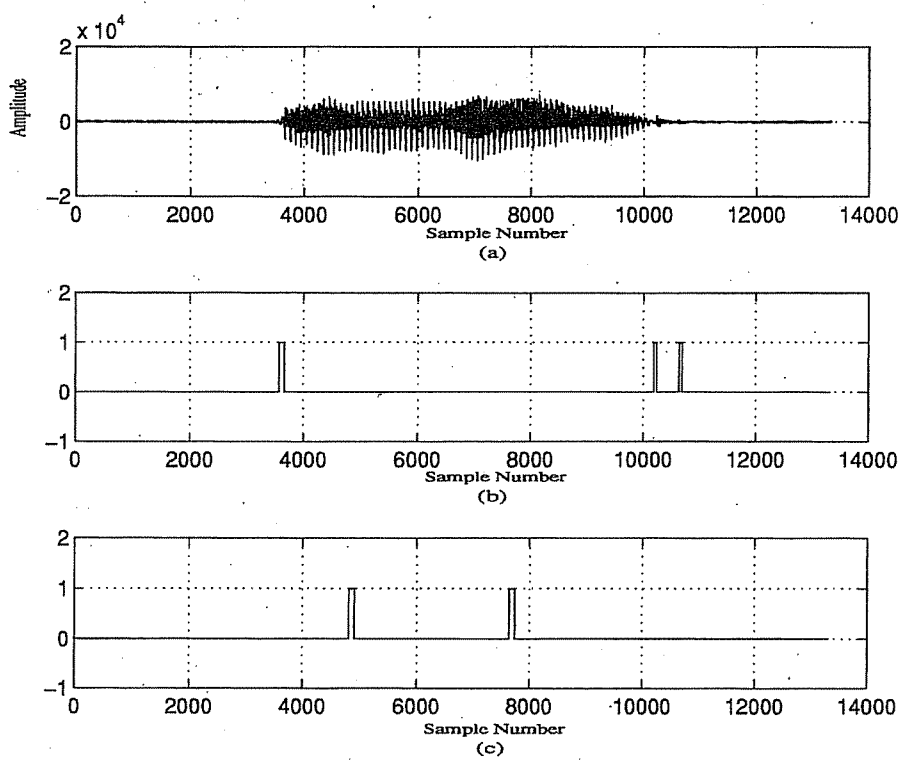

FIGURE A9.18 Final boundary detection for the word wield. (*a*) Time-domain waveform. (*b*) V/U/S boundaries. (*c*) Spectral boundaries.

Figure A9.16(*c*) at (approximately) sample numbers 3700 and 10,100 have been discarded in Figure A9.18(*c*) since they coincide with the V/U/S boundaries and are eliminated by the two-frame rule.

The inclusion of the spectral segment boundaries creates a type of "sub-segmentation" that further divides voiced segments into regions of smaller durations. Examples of this are the semivowel-vowel and the vowel-semivowel transitions shown in Figure A9.18. Here, the boundaries between the /w/ and the /i/, and the /i/ and the /l/, divide the voiced region into three distinct parts.

Spectral boundaries that occur in the middle of unvoiced segments are ignored. This is done to lessen mistakes in subsequent labeling, since the specific token words that are analyzed with this system do not contain double consonant patterns. Note, however, that double consonant patterns regularly exist in the English language. Therefore, in future work, the spectral boundaries that occur in unvoiced segments should be included in the final segmentation results.

A9.7 SEGMENT LABELING

Segment labeling is defined as the task of correctly assigning a label from one of the eight speech segment categories of Section A9.2 to each (unknown type) segment produced by the final segmentation algorithm of Section A9.6.3.

The labeling algorithm first examines the V/U/S results for each segment. If the segment is voiced, it can be labeled as a vowel, semivowel, nasal, voice bar, or voiced fricative. If the segment is unvoiced, it can be labeled as either an unvoiced stop or an unvoiced fricative. If the segment is silent, it can only be labeled as silent. For each segment, each of the possible feature scores is averaged over the duration of the segment. For example, if the unknown segment is unvoiced, then USS and UFS will both be averaged across the frames that comprise the segment. Since the segment in this example is unvoiced, the average scores for VWLS, NS, SVS, VBS, and VFS need not be calculated for the segment. Likewise, if the segment is voiced, the average scores for VWLS, NS, SVS, VBS, and VFS are all calculated, and the averages for USS and UFS are not. The average unvoiced stop score, $\mathrm{USS}_{\mathrm{mean}}$, is given by

$$\mathrm{USS}_{\mathrm{mean}}(j) = \frac{1}{b-a+1}\sum_{i=a}^{b}\mathrm{USS}(i) \tag{A9.7.1}$$

where a is the index of the starting frame in segment j, and b is the index of the final frame in segment j. The averages for all of the other feature scores are calculated in the same manner.

Once the average scores are calculated for all of the unknown segments, a first choice label, L1(j), and a second choice label, L2(j), are selected for each segment j. The first choice label for each segment is the feature with the highest mean score. The second choice label for each segment is the feature with the second highest mean score. Reliability scores are also calculated for L1(j) and L2(j). The reliability score, R1(j), for L1(j) is defined as the mean score for the first choice label divided by the sum of all of the mean scores for that segment. For example, if a segment is voiced and L1(j) is nasal, then R1(j) is given by

$$\mathrm{R1}(j) = \frac{\mathrm{NS}_{\mathrm{mean}}(j)}{\mathrm{VWLS}_{\mathrm{mean}}(j) + \mathrm{SVS}_{\mathrm{mean}}(j) + \mathrm{NS}_{\mathrm{mean}}(j) + \mathrm{VBS}_{\mathrm{mean}}(j) + \mathrm{VFS}_{\mathrm{mean}}(j)} \tag{A9.7.2}$$

Likewise, if the segment is unvoiced, and L1(j) is an unvoiced fricative, then R1(j) is given by

$$\mathrm{R1}(j) = \frac{\mathrm{UFS}_{\mathrm{mean}}(j)}{\mathrm{UFS}_{\mathrm{mean}}(j) + \mathrm{USS}_{\mathrm{mean}}(j)} \tag{A9.7.3}$$

The reliability score, R2(j), for L2(j) is defined as the mean score for the second choice label divided by the sum of all the mean scores for that segment.

There are two different cases where R1(j) and R2(j) can be used to override L1(j). In the first case, if for a given segment j, (1) the preceding segment is a vowel; and (2) the current segment is not a vowel; and (3) the current segment has a mean vowel score, $\mathrm{VWLS}_{\mathrm{mean}}$, greater than 0.45; and (4) the second choice for the segment, L2(j), is a vowel; and (5) the reliability of the second

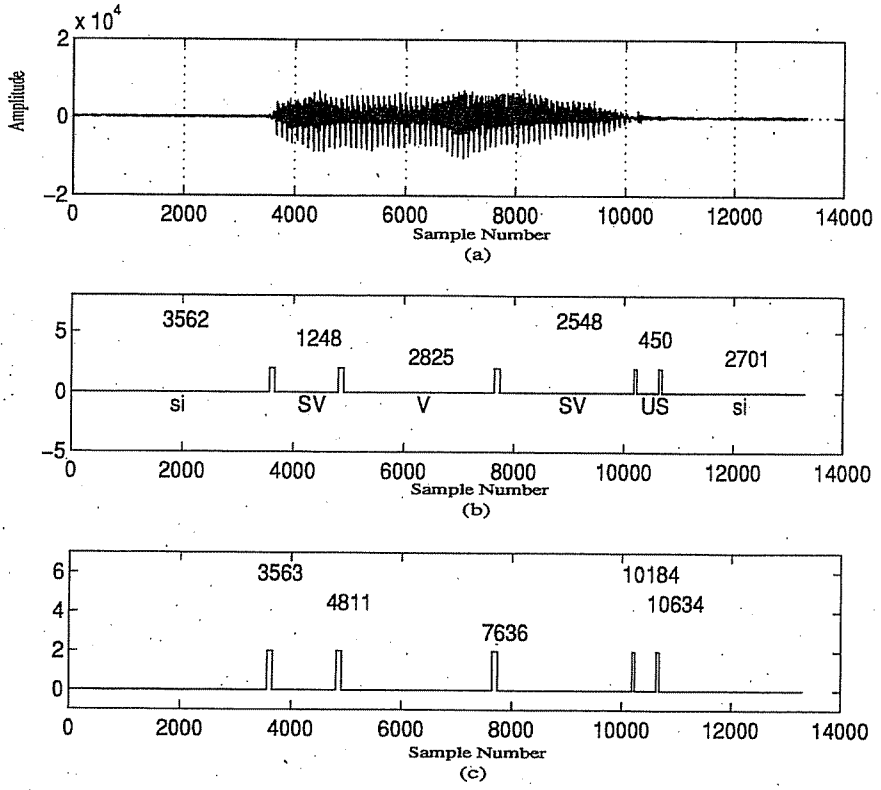

FIGURE A9.19 Final segmentation and labeling for the word wield. (*a*) Time-domain waveform. (*b*) Segment labels and segment durations (in number of samples). (*c*) Segment boundary points (boundary sample number).

choice, R2(j), is greater than 0.35, then L1(j) is changed to a vowel. This is done because the spectral sub-segmentation algorithm of Section A9.6.1 occasionally divides a single vowel into two or more parts. Often this is because the latter part of the vowel is actually starting a transition to a following non-vowel and is exhibiting coarticulation effects. The coarticulation effects may be strong enough to cause the final portion of the vowel to be classified as a non-vowel. Therefore, if the final vowel segment meets the conditions listed above, it can be assumed that coarticulation is taking place, and that the segment is actually a vowel.

The second case where R1(j) and R2(j) are used to possibly override L1(j) is as follows. First, this rule is invoked only if the preceding rule was not invoked for the current frame. Then, if (1) the current segment is a vowel; and (2) the current segment has a mean vowel score, $VWLS_{mean}$, less than 0.50; and (3) the reliability of the second choice, R2(j), is greater than 0.10, then L1(j) is changed to the second choice, L2(j). This is done because the average vowel score is less than 0.5, which implies from Section A9.5.3 that the average voiced consonant score (which is not calculated since it is not a segment category) is greater than 0.5. This indicates that the segment is actually some type of voiced consonant.

Figure A9.19 shows the final segmentation and labeling results for the word "wield" spoken by a male speaker. The symbol "si" denotes silence, the symbol "SV" denotes semivowel, the symbol "V" denotes vowel, and the symbol "US" denotes unvoiced stop. The durations of the individual segments are shown as the number of samples in A9.19*b*. The starting sample number of each new segment is shown in Figure A9.19(*c*).

A9.8 MANUAL MODIFICATION OF AUTOMATIC SEGMENTATION AND LABELING RESULTS

All speech recognition algorithms make mistakes. The number and nature of the mistakes depend upon many factors including the variability of human speakers, the choice of recognition categories,

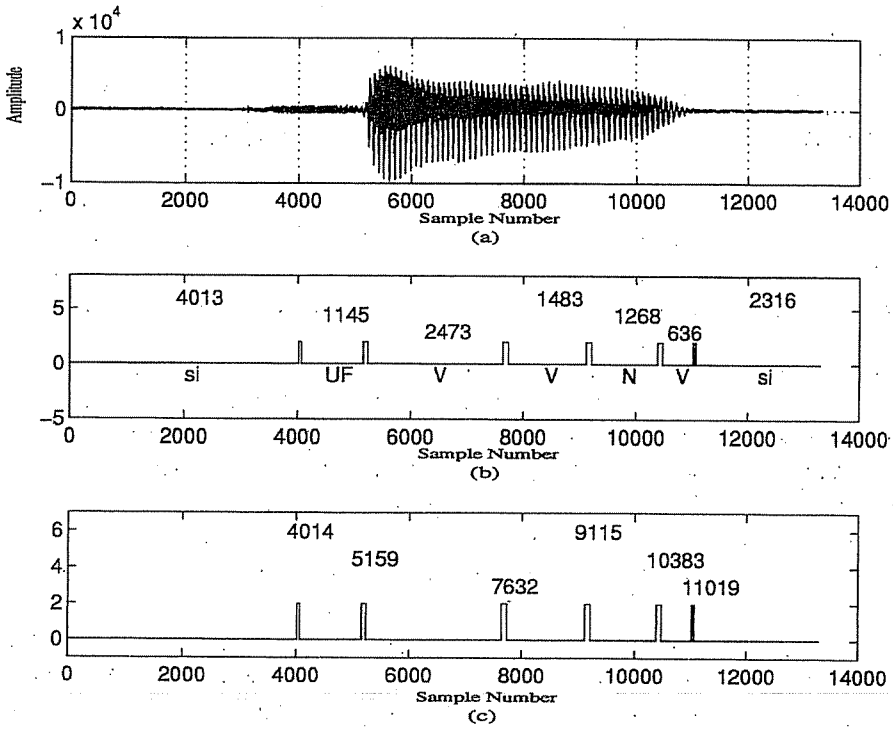

FIGURE A9.20 Final segmentation and labeling for the word foo. (*a*) Time-domain waveform. (*b*) Segment labels and segment duration (in number of samples). (*c*) Segment boundary points (boundary sample number).

and the recognition algorithms themselves. Since our primary goal is to modify the time sequence of speech, and not to create a totally new speech recognition scheme, the errors that occur in automatic segmentation and labeling are repaired manually before the time-modification algorithms are invoked.

This section describes the nature of the errors. The set of software programs along with a graphical user interface (GUI) created to help the user manually edit and correct the automatic segmentation and labeling results are described in Chapter 8.

A9.8.1 Description of Errors

A variety of errors can occur. Segmentation errors result when the algorithms pick either an incorrect number of segments, or incorrect locations for the segment boundaries. A labeling error results when the algorithms pick the wrong label for a segment. Figure A9.20 shows examples of these types of errors.

In Figure A9.20(*c*), the beginning of the unvoiced fricative is incorrectly detected as starting at sample number 4014. Examination of the time waveform shows that the actual beginning of the unvoiced fricative is closer to sample number 3300. This type of error typically occurs with weak, unvoiced fricatives such as /f/, since the energy level of the /f/ is not much greater than the energy level of the background noise. Strong, unvoiced fricatives such as /s/ and /s/ do not exhibit this problem.

A second type of error is seen in Figure A9.20(*b*). The vowel is divided into four parts by the spectral segmentation algorithm. While this is not a problem in itself, it creates the possibility for labeling errors by requiring that the four parts of the same vowel be labeled separately.

A third type of error is also seen in Figure A9.20(*b*). An error occurs in the labeling stage for the third part of the vowel. The third part is labeled as a nasal (N), instead of a vowel (V). This occurs most often for long, voiced segments that are comprised of nasalized vowels and/or word-final vowels.

Figure A9.21 shows a different type of error caused by the spectral segmentation algorithm for the word "veal" spoken by a male speaker. In this case, the boundary between the /v/ and the /i/ is not detected. As a result, the combined /v/-/i/ segment is classified as a vowel, since the vowel score has the

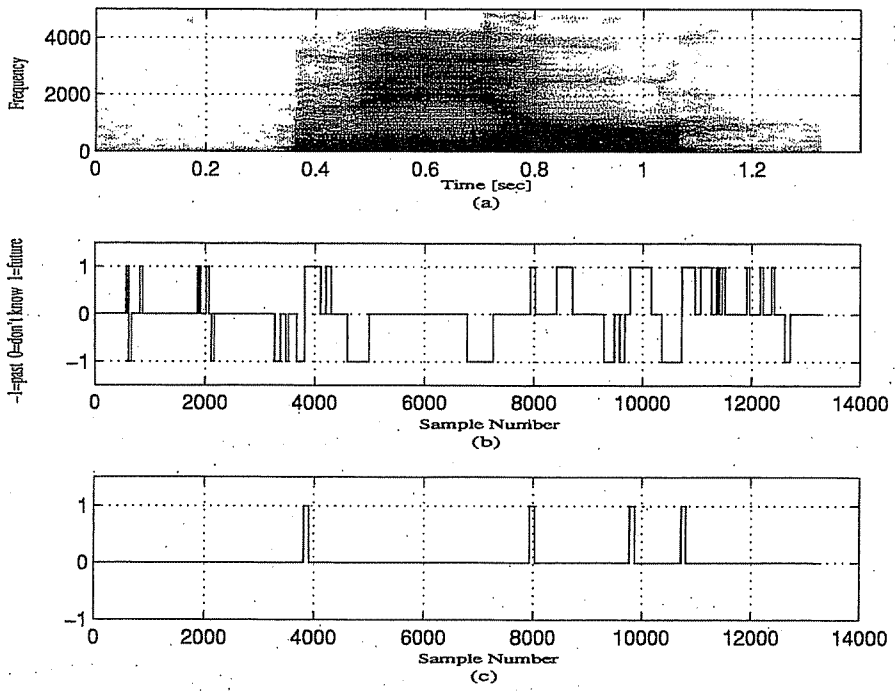

FIGURE A9.21 Spectral segmentation for the word veal. (*a*) Spectrogram. (*b*) Frame association. (*c*) Spectral boundaries.

greatest mean value for the segment. This typically occurs for the weak voiced fricatives, namely /v/ and /th/.

Figure A9.22 also shows the segmentation and labeling results for the word "veal" spoken by a male speaker. Note that the final portion of the word in Figure A9.22(*b*) is classified as an unvoiced stop. Examination of the spectrogram in Figure A9.21(*a*) shows that there is significant, unvoiced, low-frequency noise present at the end of the word. Listening reveals that the noise is caused by the speaker exhaling after completion of the word. Therefore, while unexpected, an unvoiced segment does exist at the end of the word, and it is correctly detected.

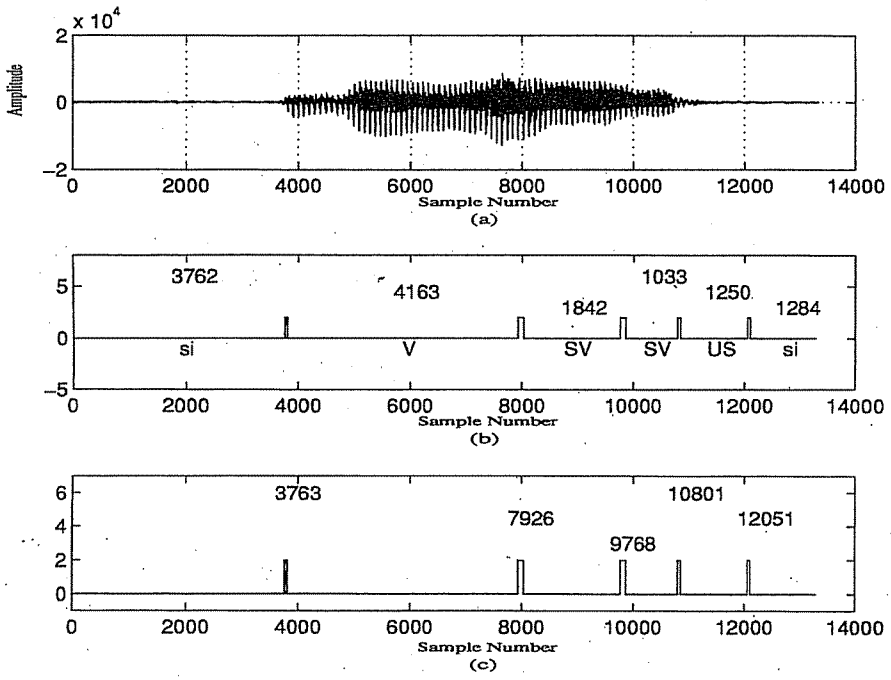

FIGURE A9.22 Final segmentation and labeling for the word veal. (*a*) Time-domain waveform. (*b*) Segment labels and segment durations (in number of samples). (*c*) Segment boundary points (boundary sample number).

A9.9 THE LINEAR PREDICTION CODING (LPC) SPEECH SYNTHESIZER

The all-pole synthesizer filter is expressed in the z-domain as

$$H_i(z) = \frac{1}{A_i(z)} \tag{A9.9.1}$$

where $A_i(z)$ for frame i of the speech signal is

$$A_i(z) = a_{0_i} + a_{1_i}z^{-1} + a_{2_i}z^{-2} + \cdots + a_{N_i}z^{-N} \tag{A9.9.2}$$

The vector $A_i(z)$ is calculated and stored for each analysis frame. The value $N = 13$ adequately models the human speech production system (Hu, 1993).

The error signal obtained during the LPC analysis is the residue signal, r(n), which is used to excite the all-pole filter during synthesis. Let $R_i(n)$ be defined as the portion of the residue signal obtained during the analysis of frame i. For example, if frame 2 begins at sample number 101 and ends at sample number 150, then $R_2(n)$ is the 1 by 50 vector given by

$$R_2(n) = [r(101), r(102), r(103), \ldots, r(149), r(150)] \tag{A9.9.3}$$

The input to the LPC speech synthesizer for frame i can be described by the ordered pair (A_i, R_i), where A_i is a 1 by N vector of LPC coefficients, R_i is a 1 by M_i vector of the residue signal, and M_i is the length (in number of samples) of frame i. Note that M_i is not constant for a pitch-synchronous synthesizer, since the frame size usually changes from pitch period to pitch period.

A9.10 TIME MODIFICATION BASICS— FRAME SKIPPING AND FRAME DOUBLING

The basic time-modification method used here involves either elimination (for compression) or doubling (for expansion) of the (A_i, R_i) ordered pairs prior to being used by the LPC synthesizer. This is best illustrated by the following examples. To synthesize the original speech token without time modification, the ordered pairs (A_i, R_i) are sent to the LPC speech synthesizer for $i = (1, 2, 3, \ldots, L-2, L-1, L)$, where L is the total number of frames in the original speech signal. As a result, each frame is synthesized once. This is depicted in Figure A9.23(*a*). To synthesize the token at approximately twice the original speaking rate (one-half the original duration) the ordered pairs (A_i, R_i) are sent to the synthesizer for $i = (1, 3, 5, \ldots, L-4, L-2, L)$. This method skips every other frame during the synthesis process. This is depicted in Figure A9.23(*b*). Note that we said that the token is synthesized at approximately twice the original speaking rate because the pitch period (and therefore the duration) of each frame is not constant. To synthesize the token at one-half the original speaking rate (twice the original duration) the ordered pairs (A_i, R_i), where $i = (1, 1, 2, 2, 3, 3, \ldots, L-2, L-2, L-1, L-1, L, L)$, are sent to the synthesizer. This method synthesizes every frame twice, and is depicted in Figure A9.23(*c*). The three synthesized speech tokens created by these methods for the word "meat" are shown in Figure A9.24. Note that for each of these examples, the silent segment preceding the unmodified word (from sample number 0 to sample number 3300) is not modified. This is done only for demonstration purposes to preserve the alignment of the beginnings of the three synthesized speech tokens in the three graphs.

Although these examples are simple, they demonstrate the basic method of time modification of speech used in this system. This method involves the manipulation of the sequence of (A_i, R_i) ordered pairs used as inputs by the LPC speech synthesizer. Note however, that these examples do not demonstrate selective time modification, since they control the information that is removed or doubled in a simple manner. Selective time modification is accomplished by exercising greater control (than the previous examples) over the ordered pairs used for synthesis. For this system, the selection of the ordered pairs is based on multiple user-specified parameters that are based on the phonemic content of the speech token.

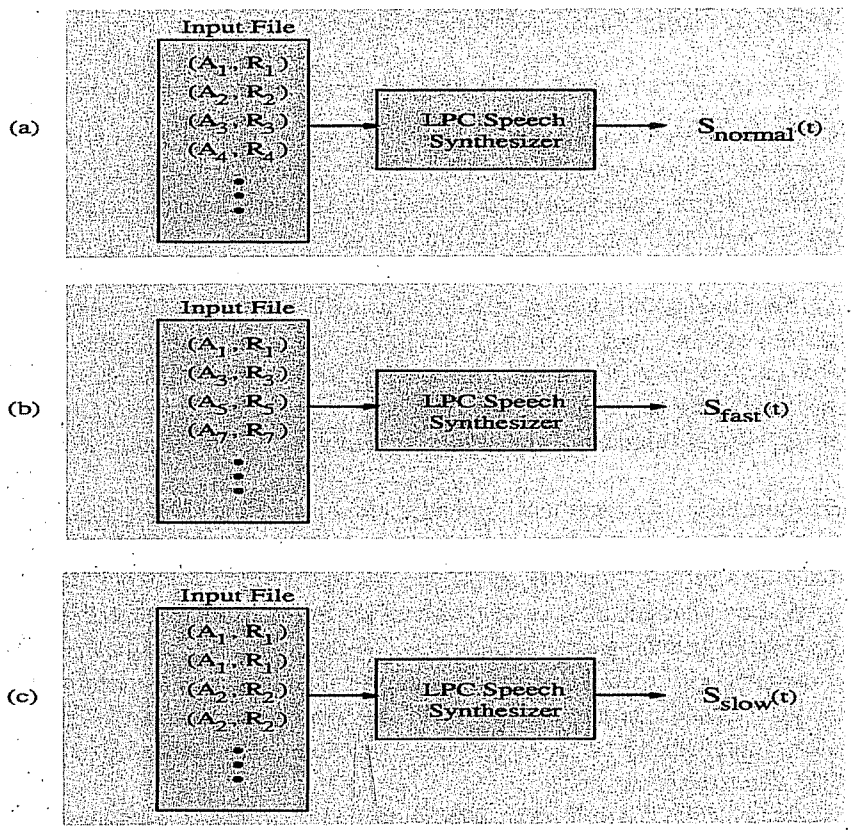

FIGURE A9.23 Examples of time modification using an LPC speech synthesizer. (*a*) Normal rate with no time modification. (*b*) Fast rate with approximately one-half the original duration. (*c*) Slow rate with twice the original duration.

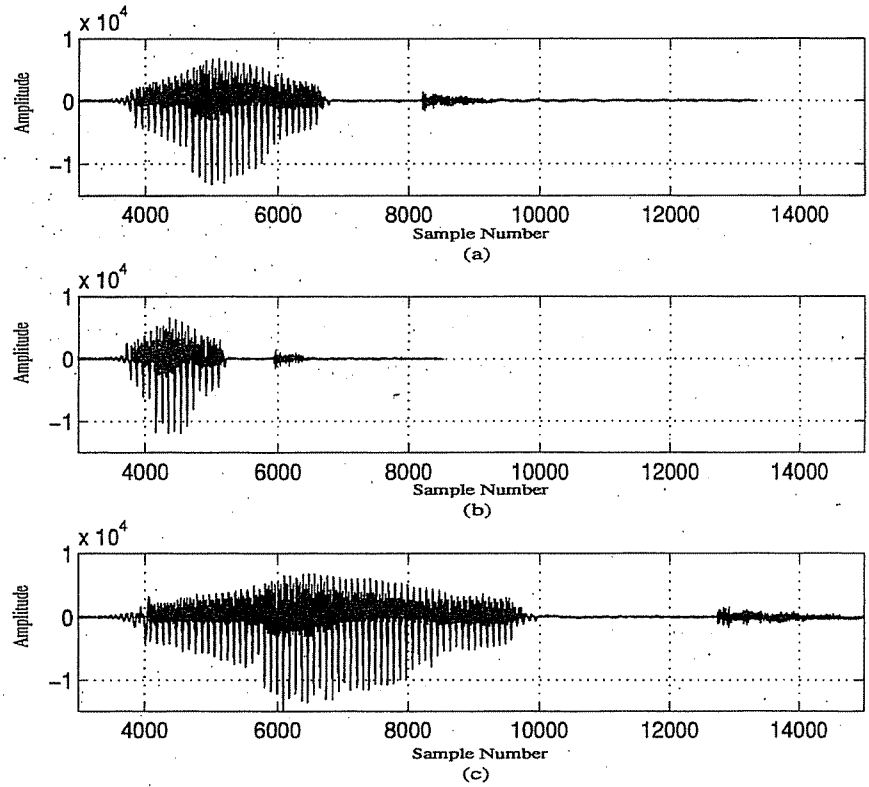

FIGURE A9.24 Examples of time-modified speech. (*a*) Normal rate with no time modification. (*b*) Fast rate with approximately one-half the original duration. (*c*) Slow rate with twice the original duration.

A9.11 USER-SPECIFIED TIME-MODIFICATION PARAMETERS

The user-specified time-modification parameters are based on the eight segment types defined previously. Each segment type (vowel, nasal, unvoiced fricative, etc.) has two, global, user-specified parameters. The first parameter is the duration scale factor (SF). It is expressed as a real number with a resolution of 0.01. There are a total of eight SF parameters, with one per segment type, that is, SF_{vowel}, SF_{nasal}, and so on. The SF parameter specifies the desired ratio of the final segment duration to the original segment duration. For example, if $SF_{vowel} = 0.33$, the duration of the vowel segment(s) in the time-modified word are approximately 33% of the duration of the corresponding vowel segment(s) in the original word. Again we say approximately, since the resolution of the algorithm is controlled by the frame size. The frame is the smallest unit that can be added or removed in the algorithm, and the duration of a discrete number of frames may not exactly equal 33% of the original duration.

The second global parameter is the minimum segment duration (MD), which specifies the minimum duration of the time-modified segment. It is expressed in milliseconds. There are a total of eight MD parameters, one per segment type; that is, MD_{vowel}, MD_{nasal}, and so on. Note that this parameter can override the desired final duration calculated from the segment's SF parameter. For example, if $SF_{vowel} = 0.00$, then the desired final duration of each vowel segment in the time-modified word is, initially, zero msec. However, if MD_{vowel} is greater than zero, the algorithm automatically adjusts SF_{vowel} so that the final duration of each vowel segment is as close as possible to MD_{vowel}. Note that the final duration may not be equal to MD_{vowel} since the resolution is equal to the duration of a single frame, as discussed previously.

The manual scale factor (MSF) is a third, user-specified parameter. It is not global in scope, and is not associated with one particular type of segment. It is expressed as a real number, with a resolution of 0.01. The MSF parameter is used to override the SF parameter for a given segment. A separate MSF value is specified for each segment in the speech token. For example, in the word "man," the initial /m/, the /a/, and the final /n/ each have a unique MSF parameter. The default MSF value is unity, and the MSF parameter must also be "activated" for each segment if it is to be used. The default state is "inactive." In addition, the MD parameter can override the desired final duration calculated from the MSF parameter in the same manner that the MD parameter can override the desired final duration calculated from the SF parameter.

The MSF parameter allows a single occurrence of a particular type of segment to be modified independently in words that have multiple occurrences of the segment type. An example of how the MSF parameter is used is as follows. Suppose, from the previous example, that the word "man" is being modified. If $SF_{nasal} = 0.50$, then both the initial and final nasals are modified during synthesis. If the user only wants the initial nasal to be modified, he or she activates the MSF parameter for segment 3 (the final /n/), and sets its value to 1.00. Since the activated MSF parameter for segment 3 overrides any global SF parameter, the final nasal /n/ has a scale factor of 1.00, which results in no time modification of the segment.

A9.12 MAPPING

Once the desired final durations are specified, the algorithm must determine the frames of a given segment that will be removed or doubled. In the simple examples of Section A9.10, every other frame was removed to achieve compression, and every frame was doubled to achieve expansion. However, these techniques do not offer the flexibility of modifying specific portions of a given segment (i.e., selective modification).

This algorithm uses a method that allows the user to assign a weighting function, or "map," to each segment. Each frame in the segment is assigned a weight between zero and one. During synthesis, the frames with the lowest weights are eliminated (if SF < 1.00), and the frames with the highest weights are doubled (if SF > 1.00). Obviously, if SF (or MSF) for the segment is 1.00, the weight is trivial, since frames are neither eliminated nor doubled.

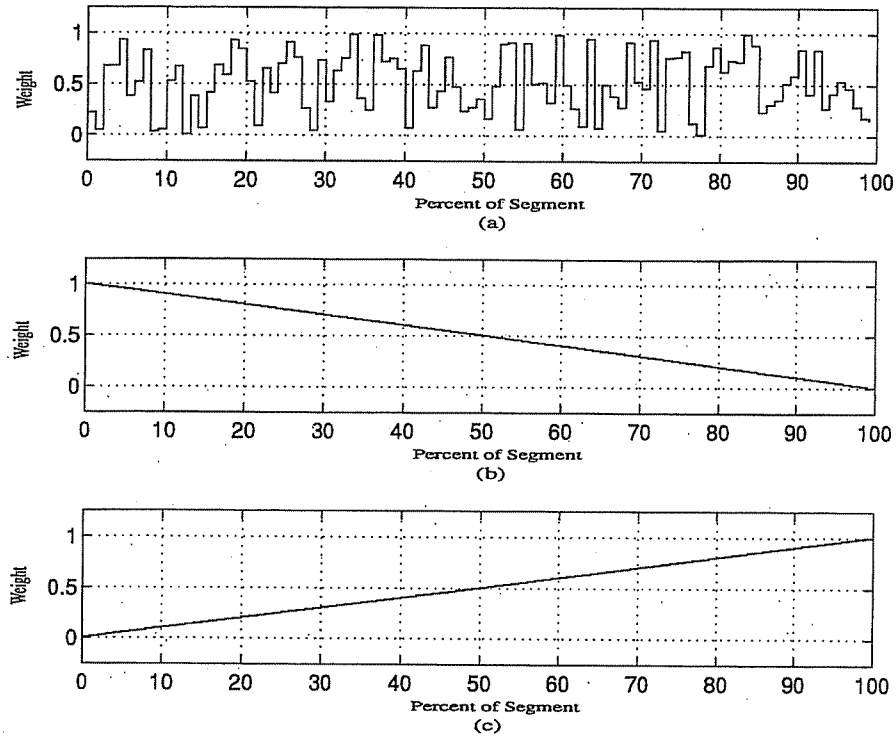

FIGURE A9.25 User selectable mapping functions. (*a*) random. (*b*) fixed_1. (*c*) fixed_2.

The user selects one of eight weighting maps for each segment. Five of the maps are fixed (the "fixed maps"), and the other three can be edited by the user (the "user maps"). The maps are shown in Figures A9.25 and A9.26. The random (fixed) map shown in Figure A9.25(*a*) arbitrarily assigns a random weight to each frame in the segment. The fixed_1 map in Figure A9.25(*b*) emphasizes the beginning of the segment. The fixed_2 map in Figure A9.25(*c*) emphasizes the end of the segment. The fixed_3 map in Figure A9.26(*a*) emphasizes the end points of the segment. The fixed_4 map shown in Figure A9.26(*b*) emphasizes the middle of the segment. One of the three user maps

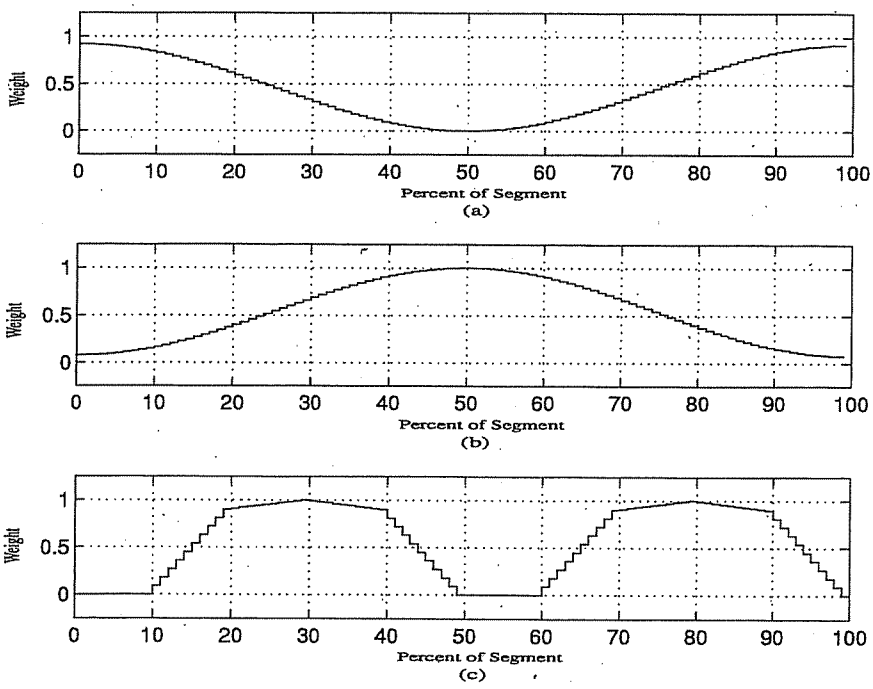

FIGURE A9.26 User selectable mapping functions. (*a*) fixed_3. (*b*) fixed_4. (*c*) user_1.

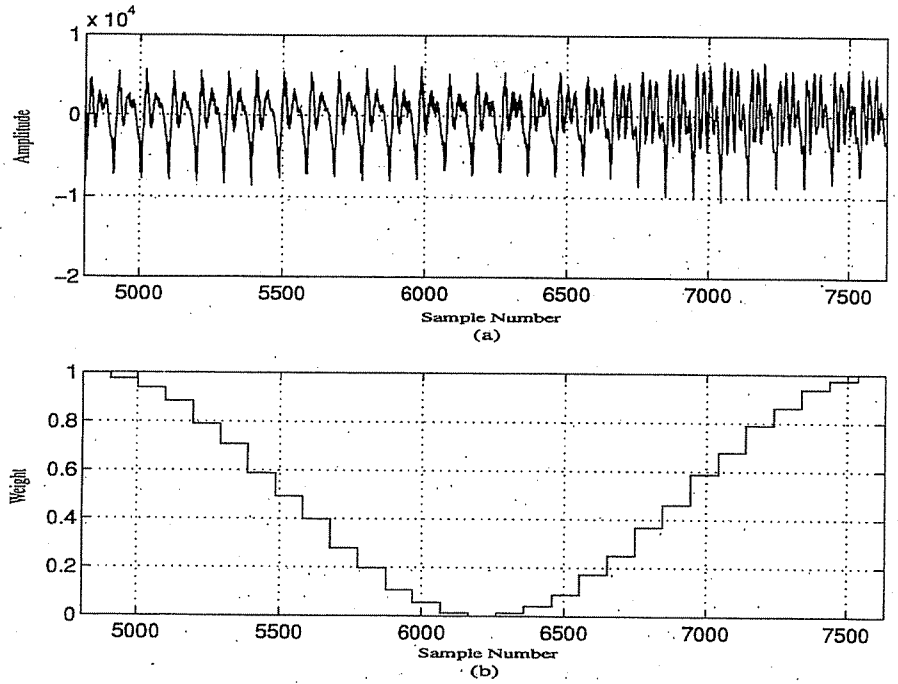

FIGURE A9.27 Interpolated map for the vowel segment of the word "wield." (*a*) Time-domain waveform. (*b*) Interpolated weighting map (fixed_3 map).

(the user_1 map) is shown in Figure A9.26(*c*). The weighting function shown is arbitrary and can be edited by the user. The three user maps are identical in function. The difference between them is that they can each contain a different weighting curve. As a result, the remaining two user maps (user_2 and user_3) are not shown. The three user maps are saved for each speech token. As a result, each token has its own, unique, mapping functions. For example, the user_1 map for the word "meat" is not necessarily the same as the user_1 map for the word "sue."

Each map is stored as a 1 by 100 vector. Since few segments have exactly 100 frames, an interpolation process determines the actual weight for each frame in the segment. The process first maps the weighting map onto the segment of interest. This is done by creating a temporary map (of length 1 by N samples) by

$$\text{temp\_map(i)} = \text{map(j)}; \quad 1 \le \text{i} \le \text{N}, \quad \text{j} = \text{ceil}\left\{\frac{100\text{i}}{\text{N}}\right\} \tag{A9.12.1}$$

where map(j) is the original 1 by 100 weighting map, N is the length of the segment in samples, and ceil is a MATLAB function that rounds the argument up to the closest integer. Once the temporary map is calculated, the sample number that resides at the center of each frame is then determined. The weight for each frame of the interpolated map is the value of temp_map(i) for the center sample of each frame.

An example of map interpolation is shown in Figure A9.27 for the vowel segment of the word "wield" spoken by a male speaker. The fixed_3 map is shown after interpolation in Figure A9.27(*b*). Note that each pitch period (frame) in Figure A9.27(*a*) has exactly one corresponding weight in Figure A9.27(*b*).

A9.13 TIME MODIFICATION AND SYNTHESIS

The time modification and synthesis processes are outlined in Figure A9.28. The algorithm begins by creating a 1 by M vector, F_{save}, of frame indices where

$$F_{save} = [1, 2, 3, \ldots, M-2, M-1, M] \tag{A9.13.1}$$

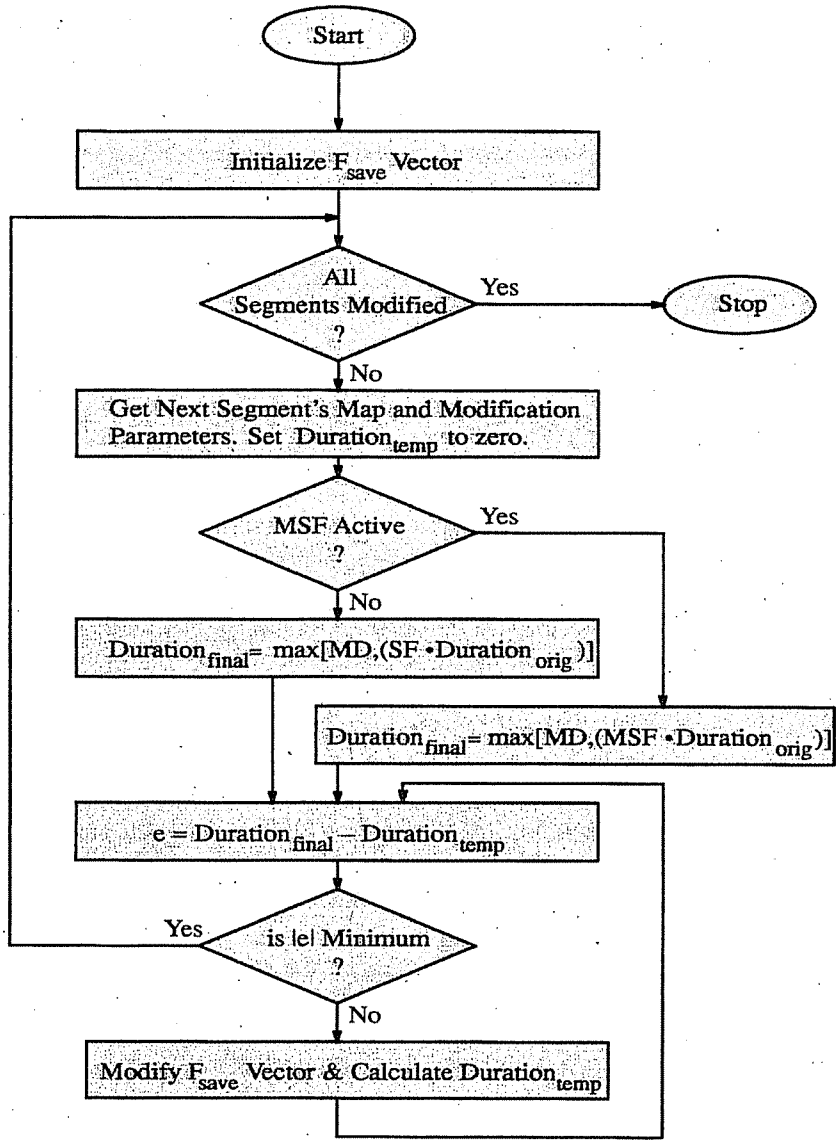

FIGURE A9.28 Block diagram of the time-modification algorithm.

and M is the total number of frames in the original, unmodified word. Once the scale factors and minimum durations are specified, the algorithm calculates the desired final duration for each segment. The process then enters an iterative loop for each segment. If frames are to be cut in a particular segment, then the algorithm examines the interpolated map for the segment, and removes the frame index from F_{save} that corresponds to the frame in the segment with the lowest weight. As a result, F_{save} becomes a 1 by M − 1 vector. If frames are to be added (instead of cut), the frame index with the highest weight is duplicated, and F_{save}, becomes a 1 by M + 1 vector. The duration of the total (new) number of frames in the segment is then calculated (this is denoted as $Duration_{temp}$ in Figure A9.28), and compared to the desired final duration. The algorithm continues in the loop, and removes (or adds) one frame index from F_{save} during each pass. Once the desired final duration is reached for the segment, the process exits the loop. The algorithm repeats the loop for each segment that is modified. The final result is F_{save}, which contains only the indices of the frames that are used to synthesize the time-modified speech.

Note that the algorithm cuts (or adds) the number of frames that causes the actual final duration for the segment to be as close as possible to the desired final duration. However, these two durations are not always the same. Tests show that the actual scale factor (calculated as the ratio of the duration of the modified segment to the duration of the corresponding unmodified segment) usually differs only by one or two percentage points from the desired scale factor. The tests also show that the actual

segment duration is within 2.5 msec of the desired segment duration for unvoiced segments, and 5.0 msec of the desired segment duration for voiced segments.

During synthesis, F_{save} controls the frames to be synthesized. Before synthesis, F_{save} is numerically sorted to guarantee that the elements occur in ascending order. The synthesizer then reads one frame index from F_{save} for each frame of speech that is synthesized. It uses the corresponding (A_i, R_i) ordered pair to obtain the excitation signal and filter coefficients.

The synthesizer architecture is a single, 13th-order, Direct-I type filter. The typical cascade of multiple, second-order sections is not used. Since all of the calculations are done in MATLAB, the filter architecture does not exhibit significant numerical precision problems (i.e., instability due to truncated coefficients).

A9.14 GLITCH PREVENTION

It is important to ensure that the synthesizer does not create artificial discontinuities, or "glitches," in the output signal. These glitches can occur at frame boundaries as the result of discontinuities in the F_{save} vector when frames are removed. The discontinuities can also result from frames being doubled, although the effects are not as severe as those caused by frame elimination. Frame removal can result in discontinuities in the final excitation (residue) signal, as well as large, abrupt changes in the frequency response of the LPC filter. In general, in order to prevent glitches, the contents of the 13 filter taps are analyzed at the frame boundaries. The contents of the filter taps are either preserved or modified, depending upon the specific case.

In the first case, if the indices of the frames on either side of the boundary are sequential with no missing frames, the filter contents are not modified. Thus, the final conditions of the filter after the last sample of the previous frame are the initial conditions of the filter for the first sample of the current frame. For example, if $F_{save} = [1, 2]$, then the final conditions for frame 1 are the initial conditions for frame 2.

In the second case, if the indices of the frames on either side of the boundary are not sequential and have doubled frames, the filter contents are not modified. Thus, the final conditions of the filter after the last sample of the previous frame are the initial conditions of the filter for the first sample of the current frame. For example, this applies if $F_{save} = [1, 1, 1]$. Note, however, that the excitation signal is modified according to a method developed by Hu (1993). This modification is done in two stages. In both stages, it is assumed that the previous frame index is the same as the current frame index. In the first stage, the amplitude of the excitation signal associated with the current frame is multiplied by a scale factor. This is done to simulate shimmer. The scale factor is constant over the duration of the frame, but changes randomly from frame to frame. The scale factor has a uniform distribution over the range [0.975, 1.025]. In the second stage, if the doubled frame is unvoiced, the excitation signal for the current frame is replaced by a zero-mean, white noise sequence that has the same length as the excitation signal. The amplitude of the white-noise excitation signal is scaled to have the same root-mean-square (RMS) value as the original excitation signal for the frame. This is done by Hu (1993) to prevent audible artifacts, such as "warble" or "phasing" effects that occur during time expansion of unvoiced segments. In theory, this is not required, since the excitation (i.e., the LPC residue) for unvoiced frames is a white-noise sequence. Therefore, the act of repeating the sequence should not create the artifacts described. However, due to the relatively short duration of an unvoiced frame (50 samples), the residue for a single frame typically does not exhibit a flat frequency spectrum. Therefore, the act of repeating the "non-white" sequence may indeed cause artifacts to be perceived. This is due to the possibility of periodicity existing in the resulting duplicated excitation sequences.

In the third case, if the indices of the frames on either side of the boundary are not sequential and one or more frames are missing, then a special rule is invoked. The filter calculates the initial conditions that would exist if the single, final missing frame was present, and uses these filter states as the initial conditions for the synthesized frame immediately following the missing (unsynthesized) frames. For example, if $F_{save} = [1, 2, 9]$, the filter calculates the final conditions that would be present if frame 8 were synthesized after frame 2, and uses these as the initial conditions for frame 9. The

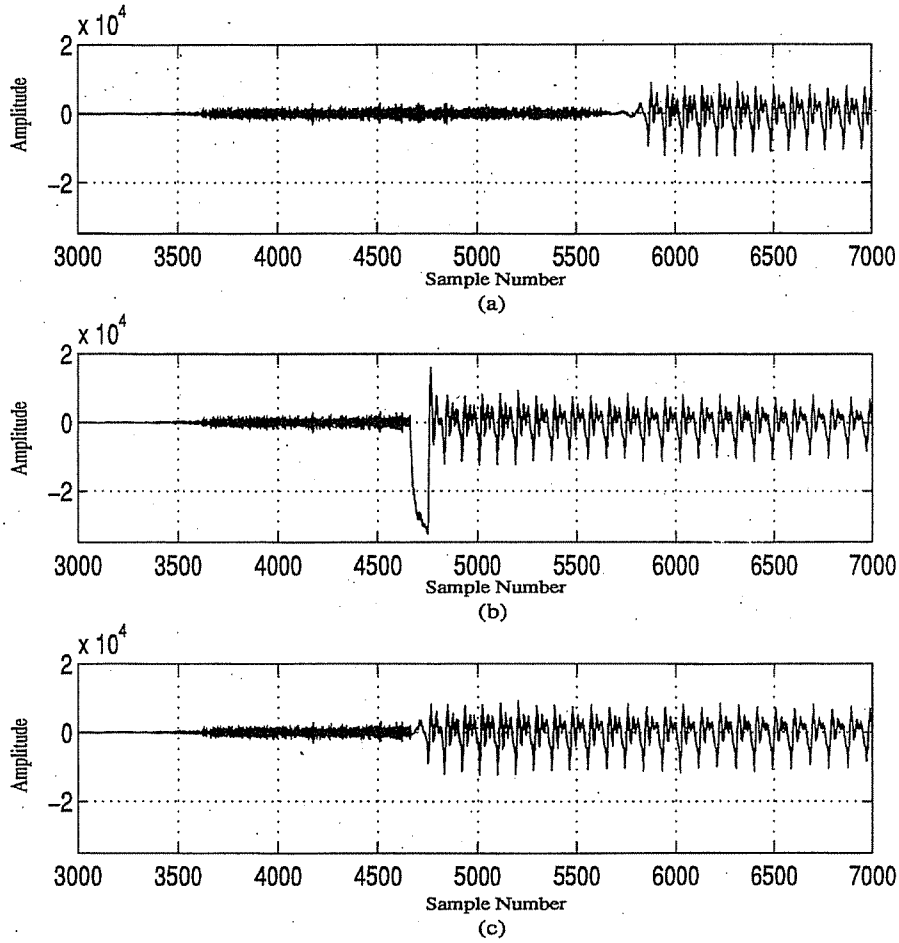

FIGURE A9.29 Glitch prevention for the unvoiced-voiced transition in the word sue. (*a*) Original unmodified waveform. (*b*) Synthesized with final 50% of /s/ removed and glitch prevention off. (*c*) synthesized with final 50% of /s/ removed and glitch prevention on.

motivation behind this rule is: The discontinuities are almost always a result of the energy that is stored in the digital filter, which can be greatly amplified by the new LPC coefficients immediately after the coefficients are updated. This occurs most often when frames are removed from an unvoiced-voiced transition region. Since the gain of the LPC filter is not normalized during the initial analysis to have a constant, average frequency response from frame to frame, the average gain of the LPC filter for voiced regions is much greater than the average gain of the filter during unvoiced regions. Thus, if no glitch prevention is performed, the stored energy in the digital filter during synthesis of an unvoiced frame is greatly amplified when the LPC coefficients are abruptly updated to those of a voiced frame.

An example of this is shown in Figure A9.29 for the transition from the /s/ to the /u/ in the word "sue" spoken by a male speaker. Figure A9.29(*a*) shows the original, unmodified word. Figure A9.29(*b*) shows the synthesized word where the final 50% of the /s/ is removed with no glitch prevention. Note that the filter creates a large glitch that dominates the output for several glottal cycles. Figure A9.29(*c*) shows the synthesized word where the final 50% of the /s/ is removed using the glitch prevention rule described above. There is no noticeable discontinuity at the transition, and the transition region closely resembles that of the original word (ignoring the position of the /s/ that is removed).

The theory described in this appendix has been implemented in the algorithms of the time modification toolbox in Chapter 8.