VARIABILITY ANALYSIS OF DISCRETE COSINE TRANSFORM COEFFICIENT (DCTC) FEATURES FOR SPEECH PROCESSING

by

Bingjun Dai

A Thesis Submitted to the Faculty of
Old Dominion University in Partial Fulfillment of the
Requirement for the Degree of

MASTER OF SCIENCE

ELECTRICAL ENGINEERING

OLD DOMINION UNIVERSITY
December 1998

ABSTRACT

VARIABILITY ANALYSIS OF DISCRETE COSINE TRANSFORM COEFFICIENT (DCTC) FEATURES FOR SPEECH PROCESSING

Bingjun Dai
Old Dominion University, 1998
Director: Dr. Stephen A. Zahorian

In this research, the variability of Discrete Cosine Transform Coefficient (DCTC) features was investigated. Additionally, a new pitch-synchronous processing method was explored to increase the stability of features and to reduce window effects when compared to the regular method. The noise sources that lead to feature variability were analyzed, and different smoothing methods were tested. It was found that longer frames, frequency warping, time smoothing of the log spectrum, and DCS level time smoothing, all help reduce DCTC variability and increase classification performance. The pitch-synchronous method was implemented with Matlab. Important processing methods, including pitch period estimation, time-domain resampling, and pitch-dependent scaled frequency range, were investigated. Twenty-four training speakers and eight test speakers were used for the classification tests. The final results showed similar performance as compared to the conventional method, which implies that window effects are a minor factor in speech recognition. However, processing using a pitch-dependent spectrum led to better performance than the conventional pitch-independent method. In addition, the new

method appears to reduce DCTC feature variability by a small amount.

ACKNOWLEGMENTS

I would like to express my gratitude to my advisor, Dr. Stephen A. Zahorian. His expertise and insight in the field of Speech Recognition has been a great help in this research. This work is impossible without his invaluable guidance, constant support and patience.

I would also give my thanks to the additional members of the advisory committee, Dr. Stoughton and Dr. Gray, for their precious time in reviewing this thesis and constructive suggestions.

Special thanks to my co-workers at ODU speech lab, Matthew Zimmer and Montri Karnjanadecha, for their suggestions and assistance.

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

FIGURE Page

CHAPTER I

INTRODUCTION

1.1 Background

In the field of speech recognition and classification, the proper definition and extraction of acoustically invariant characteristics (features) is an objective that has been sought for decades. Among the various types of features investigated over the past 40 years are two popular ones: spectral formants and features that reflect global spectral shape, such as cepstral coefficients.

Spectral formants represent the natural resonance of humans' vocal tracts in the frequency domain (Kelkar, 1992). On a spectrum plot, formants are the peaks of the spectral envelope, which are named as F1, F2, F3, etc., in the order of increasing frequency. In a classic paper, Peterson and Barney (1952) claimed that the first three formants(F1, F2 and F3) could be regarded as the primary source of spectral information. Ever since this paper, scientists and researchers have studied formant-based vowel perception extensively. Many successful models were developed to implement vowel perception using spectral formants as the fundamental feature set. For example, Miller (1989) developed the auditory-perceptual theory, which was based on formant-ratio theory, and

The reference model followed is *The Journal of the Acoustical Society of America*.

demonstrated that the preliminary target zones in formant space could be used to classify a proprietary database of American English vowels with up to 93% accuracy. Although the formant representation of speech spectra enjoys widespread use, particularly for the perception of vowels, some challenging problems remain. Bladon (1982) made arguments against the formant representation of speech primarily with regard to three aspects: first, formant representation is an incomplete spectral description; second, there is great difficulty in locating the formants in many cases; third, a formant representation does not provide a good prediction when the spectral peaks are widely spaced (Zahorian and Jagharghi, 1993). Furthermore, as stated in Kelkar’s thesis (1992), formants may be located in overlapped frequency regions; spurious peaks due to a single formant’s split make it difficult to identify the actual formant; the frequencies of adjacent formants may be too close to resolve. All these issues associated with spectral formants have caused interest in finding better feature representation methods. Global spectral shape factors have proved to be a good alternative.

As early as the late 60’s, a group of researchers had completed a series of experiments using spectral-shape representation of vowel spectra (Plomp et al., 1967; Pols et al., 1969; Klein et al., 1970). They defined a principal-components spectral-shape representation of vowel spectra and demonstrated that vowels could be classified automatically as accurately from a principal-components representation as from a formant representation (Zahorian and Jagharghi, 1993). The Discrete Cosine Transform Coefficients (DCTCs) of the log magnitude spectrum are one type of global spectral shape features. Zahorian and Gordy (1983) showed that a series cosine basis vector representation is very similar to a principal-components representation. Zahorian and Jagharghi (1993) also experimentally compared vowel classification test results between formants and DCTCs and found that the DCTC results were significantly higher (3.5%) than for the formant case, provided enough DCTCs (ten or more) were used.

The Visual Speech Display (VSD) system developed in Old Dominion University’s Speech Lab uses DCT basis vectors as the feature set to implement vowel classification. In the process of improving system performance, we have found instability of DCTC features in the real-time display. The major interest of this research is to explore the instability issue and develop approaches to address it.

1.2 Objective

The Visual Speech Display (VSD) system was originally developed on a DOS based personal computer. All signal processing work was implemented on a dedicated DSP board, with TMS320C25 being the core DSP chip. Although this hardware based system worked well in terms of recognition performance, it was expensive (due to dedicated hardware), and difficult to maintain and update (since the DSP chip requires assembly language programming). Because of all these drawbacks and thanks to the dramatic increase in CPU speed, a Windows NT/95 based system has been implemented. A standard multimedia sound card fulfills I/O operation. All signal processing and neural network classification work is done by software. This offers greater cost-effectiveness and flexibility than the old system.

However, an unexplained and unwanted phenomenon was observed in the process of system testing and refinement. Both the Bargraph Display and the Ellipse Display (these two display types are explained in Chapter II) showed “jitter” on the amplitude of the “bar” or the screen position of the “ball” even when the speaker pronounced and steadily held a vowel sound. In the meantime, a real-time plot of DCTC features showed similar variations. Intuitively, for steady-state vowels, the display should be steady, too. An “unstable” display is an indication of “unstable” neural network output, which determines the recognition performance. We suspect that this variability, to some extent, limits the classification performance of the new system. As a matter of fact, the previous hardware based system did give a more stable display when a vowel was steadily uttered. It was obvious that the migration from the previous system to the new system introduced some problems. These problems could have been the result of coding errors, background noise and noise of PC components (including the sound card), or due to changes in the methods used to compute the DCTCs.

The main objective of this research was to track the reasons of the DCTC feature variability and to investigate various approaches to eliminate this unwanted effect. First, the DCTC variability was recorded and the reasons for it were analyzed. Then, different approaches were compared and adopted to reduce the DCTC variability. Finally, a new pitch based DCTC feature computation approach was explored with the expectation of reducing variability from a different perspective. This last point is the most significant contribution of this thesis.

1.3 Overview Of Following Chapters

In chapter II, a brief description of the Visual Speech Display (VSD) software is given. The observed DCTC feature variability is described and analyzed. The efforts to eliminate or reduce the variability are also discussed.

Chapter III explores, theoretically, a pitch-synchronous approach in an attempt to enhance the front-end DCTC feature analysis. Important processing steps, such as pitch estimation, time-domain resampling, non-linear frequency scaling and pitch-dependent spectra, are explained in detail.

Chapter IV presents the experimental verification of the approach described in Chapter II using Matlab programs. System modules and routines implementing the pitch-synchronous front-end processing are listed and explained. The classification results of an Artificial Neural Network (ANN) are shown to test the performance of the pitch-synchronous front-end processor.

Chapter V summarizes the achievements attained in this research and discusses future work and possible improvements.

CHAPTER II

DCTC FEATURE VARIABILITY

2.1 Overview

The focus of this chapter is the DCTC feature variability observed on the Windows NT/95 based Visual Speech Display (VSD) system. In 2.2, a brief description is given on the structure and functionality of the VSD system. We describe the DCTC feature variability in 2.3. The possible reasons that cause this variability are explored and analyzed in the same section. Based on the analysis made in 2.3, three possible "noise" sources causing the variability are further verified, theoretically and experimentally, in 2.4 through 2.6. Conclusions are summarized in 2.7.

2.2 Visual Speech Display (VSD) Software

The Windows NT/95 based Visual Speech Display software is the result of the combined efforts of a group of people who worked or are currently working in the Speech Communications Lab. It is designed to generate visual feedback for vowel sounds uttered by speakers. An A/D converter, a front-end processor, an artificial neural network classifier and a Windows Graphical Interface are the main functional blocks in the system. Figure 1 shows the overall framework.

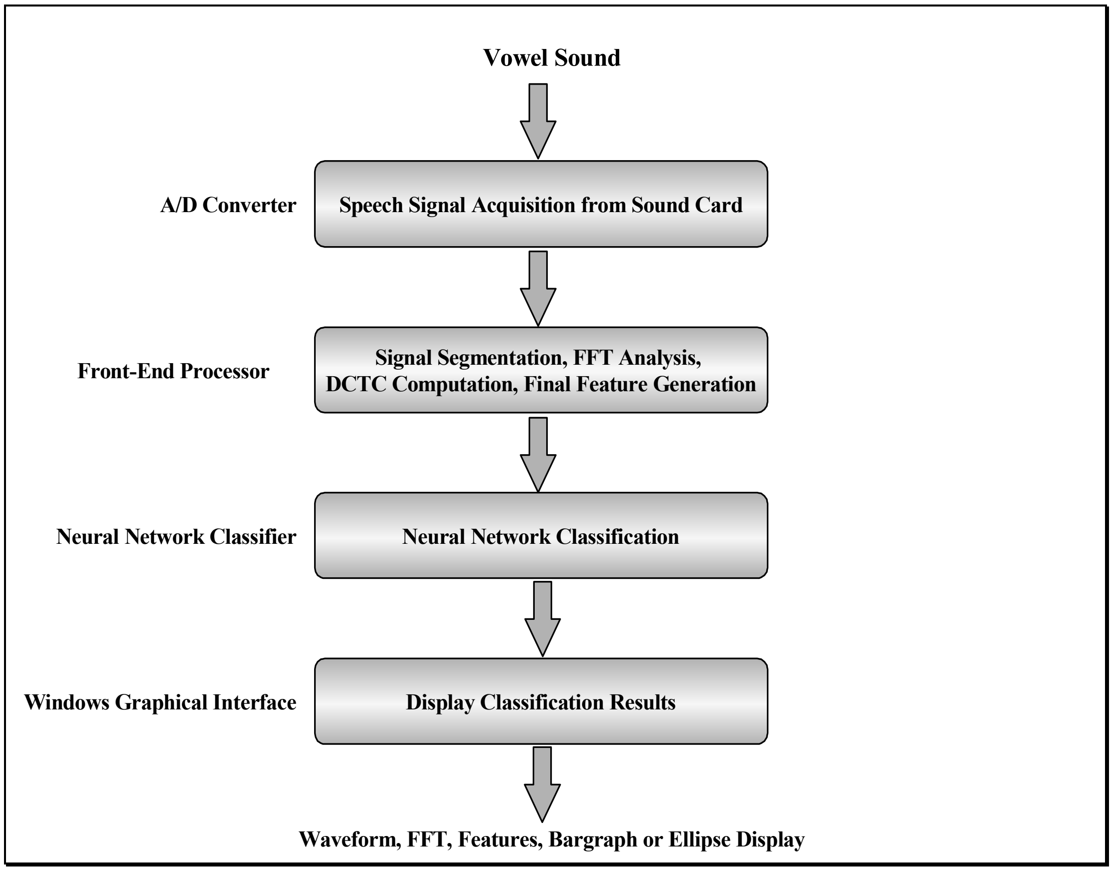

Vowel Sound
A/D Converter
Speech Signal Acquisition from Sound Card
Front-End Processor
Signal Segmentation, FFT Analysis, DCTC Computation, Final Feature Generation
Neural Network Classifier
Neural Network Classification
Windows Graphical Interface
Display Classification Results
Waveform, FFT, Features, Bargraph or Ellipse Display

Figure 1. Framework of the Visual Speech Display software

A/D Converter: the A/D converter utilizes the Windows Application Programming Interface (API) to work with a multimedia sound card to transform the analog acoustic signal into digital form.

Front-End Processor: this part can be divided into two blocks: Frame Level Processing and Block Level Processing. In the Frame Level Processing, each signal segment output by the A/D converter is divided into one or more frames. A series of subsequent processing is based on these frames. The individual frame is first windowed (using a Hamming or Kaiser window) to eliminate time-domain discontinuities at the edges of frames.

FFT analysis is then done to generate the log magnitude spectrum. Appropriate frequency smoothing and time smoothing are also implemented in this step. Frequency smoothing is done with morphological filtering which emphasizes the envelope of the spectral peaks rather than the spectral details. Time smoothing is used to reduce the temporal variability between consecutive frames. DCTC features are computed based on the smoothed spectrum of an individual frame and a series of pre-defined basis vectors over frequency (as further explained in 2.3).

In the Block Level Processing, several consecutive frames are combined as a block, and the final features, named Discrete Cosine Series (DCS), are calculated based on DCTC features of all frames in the block. This processing is explained in 2.3.

Neural Network Classifier: The features at the output of Front-End Processor are used as inputs to an artificial neural network for pattern classification. The neural network uses the back-propagation learning algorithm as described in Lippmann (1987). A typical structure of the neural network classifier used in the VSD system includes one hidden layer with 25 nodes, and 10 output nodes. For experiments reported in this thesis, the learning rate generally ranged from 0.30 to 0.35. Two hundred thousand learning iterations were generally used.

Windows Graphical Interface: In this final step, the classification output is displayed on the screen to give visual feedback to speakers. There are two primary displays: the Bargraph Display and the Ellipse Display.

The Bargraph Display shows only one significantly high bar when the corresponding vowel sound is correctly uttered; otherwise this bar is not high and/or the bars associated with some other vowels may appear. Figure 2 illustrates a correct utterance case, while Figure 3 shows an incorrect one.

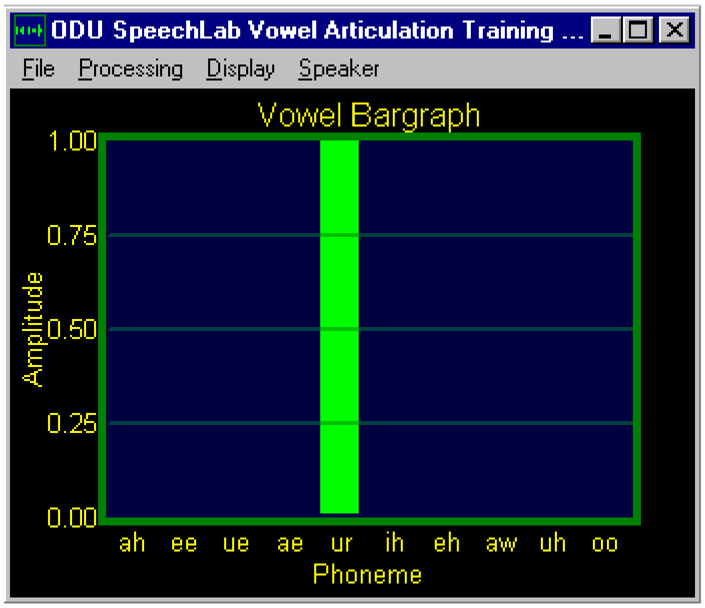


Figure 2. Bargraph Display when the vowel sound "ur" is correctly uttered

The Ellipse Display defines 10 elliptic areas on the screen corresponding to the 10 vowel sounds. When a vowel is properly uttered, a small basketball slides into the oval associated with that vowel sound, and the color of the basketball is changed to match that of the oval. The size of the ball is designed to vary with the strength of the sound as an indicator of loudness. Figure 4 gives an example of the Ellipse Display.

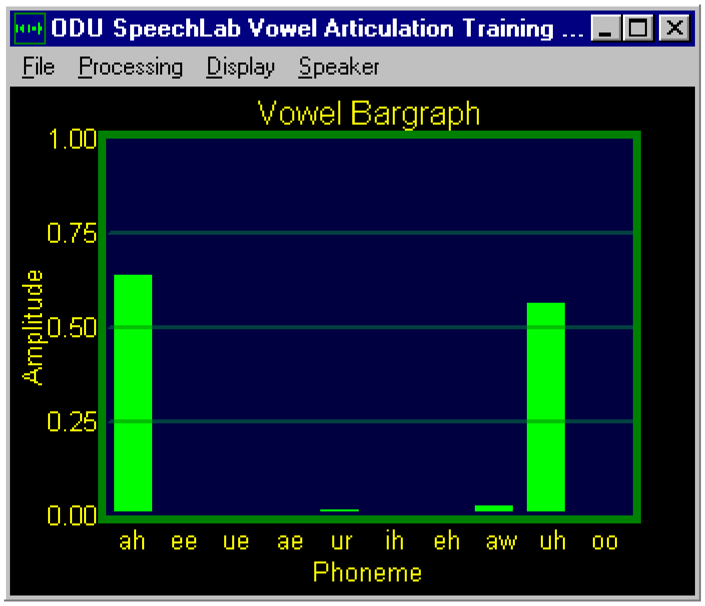


Figure 3. Bargraph Display when a vowel sound "ah" is incorrectly uttered

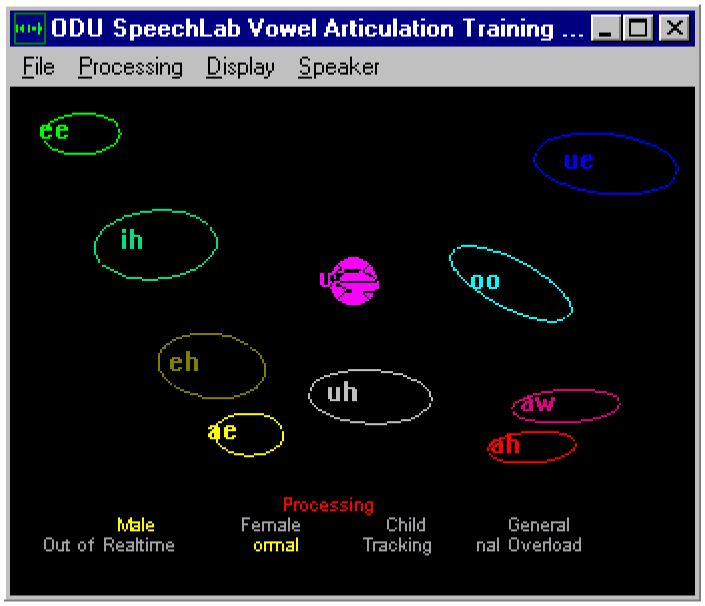


Figure 4. Ellipse Display when the vowel "ur" is correctly uttered

2.3 DCTC Feature Variability Analysis

When we evaluated the overall performance of the Visual Speech Display system as described above, we observed “jitter” in both the Bargraph Display and the Ellipse Display even if the speaker held his utterance as stable as possible. This variability could also be shown by displaying the FFT spectrum and the DCTC feature values. In contrast, the previously developed DSP-board-based system appeared to have had a more stable display. Intuitively, the more stable the display, the higher the performance. We suspected that some type of “noise” had been introduced into the system in the process of the migration work. We focused on the analysis of three major possible noise sources: coding errors, noise from background and PC components, or some inherently unstable nature of DCTC features.

2.3.1 DCTC Features

Before we go into the analysis, it is necessary to describe the DCTC features mathematically. As described in Nossair (1989), the DCTCs encode the smoothed global shape of the spectrum, which provides the primary information about the spectral pattern of the speech. The DCTCs are computed using a simple dot product equation involving the spectrum and cosine basis vectors, as given by

$$DCTC(i) = \sum_{k=0}^{N-1} X(k)\phi(i,k) \tag{2.1}$$

where $X(k)$ is the scaled log magnitude frequency sample with the sample index k ranging from 0 to (N-1), corresponding to an absolute frequency range of [0, Fs/2](Fs is the sampling rate). $\phi(i,k)$ is a pre-defined set of samples of a cosine series expansion over the same frequency range. If there is no frequency "warping," the basis vectors are obtained as sampled values of cosines, as given in Equation 2.2.

$$\phi(i,k) = \cos\left(\frac{\pi i(k+0.5)}{N}\right) \tag{2.2}$$

where i is the index of DCTC, k is the index of frequency sample, N is the total number of frequency samples. Note that if frequency "warping" is used, which leads to non-uniform frequency resolution, the equations for these basis vectors will change; otherwise, the computations are the same.

DCTCs are the features for Frame Level Processing in the system framework described in 2.2. In order to take advantage of the close correlation of neighboring frames, Discrete Cosine Series (DCS) can be calculated, based on a group of consecutive frames. DCSs are the final features to be used by the neural network for pattern classification.

$$DCS(j,i) = \sum_{l=0}^{M-1} DCTC(j,l)\Phi(l,i) \tag{2.3}$$

In Equation 2.3, the ith DCS of DCTC j is computed as the sum of the DCTC j multiplied by appropriate basis vectors, over all M frames of the block. The basis vectors used here are defined in a similar way as those for DCTC computation. The only difference is that the basis vectors in DCS calculation are computed over time rather than over frequency, as is done in DCTC feature computation. That is,

$$\Phi(l,i) = \cos\left(\frac{\pi l(i+.5)}{M}\right) \tag{2.4}$$

where l is the index of frame, i is the index of DCS, and M is the total number of frames.

Figure 5 shows a snapshot, for a single time instant, of the first 12 DCTC features in the VSD real-time display. When a vowel is uttered and held, the slight variability of the amplitude of each feature (over time) can be easily seen in the real-time display.

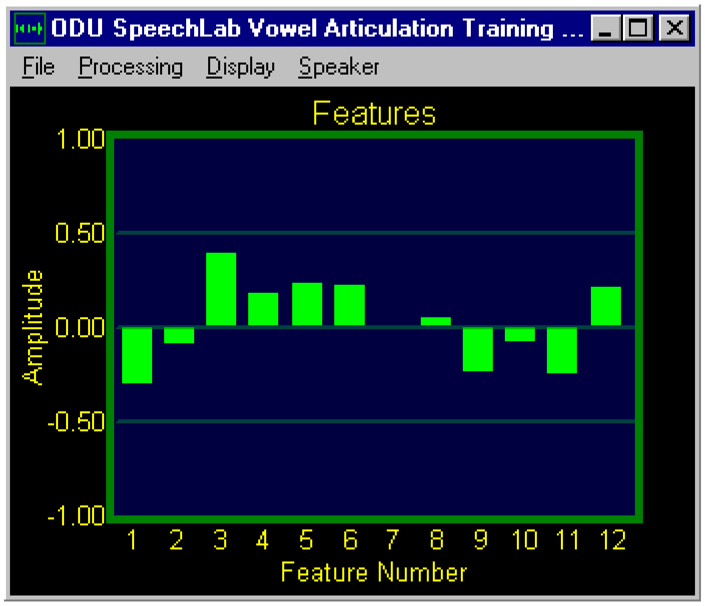


Figure 5. DCTC features displayed for vowel sound "ee"

More figures are given in 2.3.5 to better illustrate the DCTC feature variability. In the meantime, approaches to reduce this variability are explored and verified in the same section.

2.3.2 Possible Coding Errors

Although the early versions of the Visual Speech Display Software worked well, we did find some hidden flaws that could be one possible noise source causing the DCTC feature variability. The faults primarily came from modularity issues.

Modularity is the key point to a large software system, since it provides a clear structure that makes it possible for different people to add in new features or make changes. The original version of software defined global variables that were used virtually everywhere in the entire code. This made the code intertwined and greatly reduced the independence of functions, which could easily cause problems when one tried to add in, remove, or modify parts of the code. Aimed at tackling all these negative effects, a combined effort to rewrite the code of the early version was done by A.M.Zimmer, M.Karnjanadecha and myself. A new and flexible A/D and D/A converter was implemented by Karnjanadecha, which could provide full-duplex operation of the sound card. A standard fundamental Windows function template and a naming convention were defined by Zimmer. This helped standardize the programming procedure and increased the readability of the code. Zimmer and I also rewrote much of the graphical interface part, using a set of graphical APIs which we defined.

In the experimental observation and performance comparison after the careful code rewriting, we found that there was no measurable difference between the new system and the old one. This, combined with careful checking of intermediate signal calculations for known deterministic signals, implied that the DCTC feature variability was not due to coding errors.

2.3.3 Background And Component Noise

To seek the cost-effectiveness and compatibility of the entire system while maintaining good performance, a standard commercial sound card was used for I/O operation. We found that noise might come from the background and the PC components, including the sound card.

Figure 6 shows an FFT spectrum plot when there was no signal input (the sound card microphone was turned off). This power spectrum indicates the existence of noise from the sound card and other PC components.

Figure 8 is the FFT plot with the same sound card used for Figure 6, but when the microphone was turned on. It can be observed that the overall amplitude of noise power spectrum has increased, particularly in low frequency bands (< 2kHz). This indicates the existence of background noise.

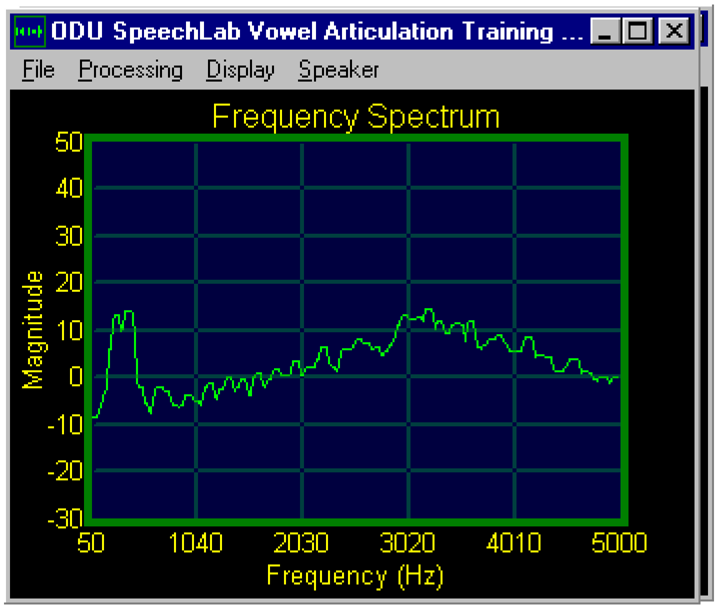


Figure 6. FFT plot on PC #1 when the sound card microphone was turned off

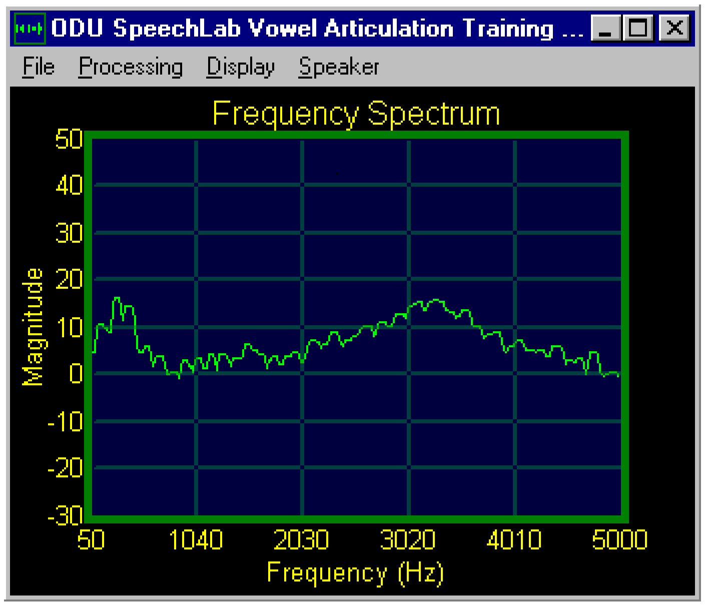


Figure 8. ***FFT plot on PC #1 when the microphone was turned on, but no signal input***

Another noticeable point is that the noise from the sound card and other PC components varies from machine to machine. Figure 6 and Figure 7 are FFT plots on PC #1 and PC #2, respectively. The microphones on both sound cards were turned off. Both sound cards had the same volume settings. It is easily seen that the noise produced on PC #2 is significantly lower than on PC #1. Note that PC #2 was a newer machine with higher quality sound card. This indicates that the noise strength depends upon the quality of PC and sound card.

The most important issue we are concerned with is the extent to which the background and component noise affects system performance. The comparison of Figure 8 and Figure 9 serves as an example for evaluating this extent. Figure 8 is the FFT plot on PC #1 with the sound card microphone turned on, but without speech signal input. Figure 9 is the FFT plot on PC #1 when a vowel sound "ee" was uttered. Hence, Figure 8 shows the noise power spectrum, while Figure 9 gives a description of power spectrum of mixed signal and noise. The quantity of Signal to Noise Ratio (SNR) can be roughly estimated by comparing Figure 8 and Figure 9. It can be seen that the noise may not affect spectral formants since the SNR around formants is generally greater than 15dB.

However, in the spectral valleys where SNR is close to 0dB or even lower, noise does play a significant role.

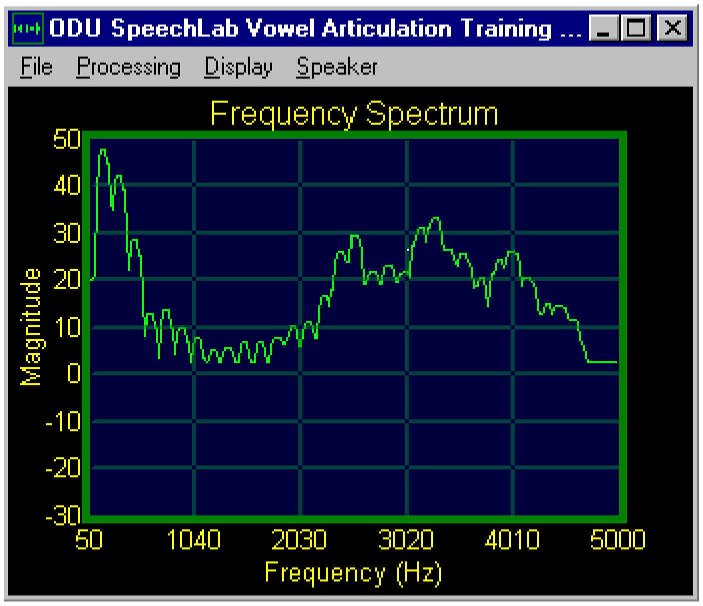


Figure 9. FFT plot on PC #1 when the vowel sound "ee" was uttered

In summary, the background noise strength may vary from place to place and from instant to instant; the noise from components depends on the quality of PC and sound card, and is different from machine to machine. Using high quality PCs and sound cards reduces the component noise. As observed in this research, noise affects spectral details in certain frequency bands, especially in spectral valleys. For speech recognition, since general spectral shape is more important than spectral details, we may use smoothing in the frequency domain to smooth out spectral details and thus reduce the negative effects of noise. Different smoothing methods are further discussed in 2.3.5.

2.3.4 Possible Inherently Unstable Nature of DCTC Features

As claimed in Zahorian and Jagharghi (1993), the DCTCs, as spectral trajectory features, are computed the same way as cepstral coefficients commonly used in speech processing except for small differences. The exploration of the oscillations of cepstral coefficient-based trajectory models may reflect some similar characteristics of DCTC features.

A recent paper by Hu and Barnard (1997) described a measurement method developed to evaluate the smoothness of cepstral coefficient-based trajectories. They found that the trajectories contain spurious oscillations over time, which suggests that the trajectory features might have a high within-class variance. This happens even when the speech signal is changing slowly and smoothly. This issue could be due to the mapping between cepstral coefficients and frequency components. The researchers set up a model and a series of relevant simulation experiments using both rising tones (generated by adding two sinusoidal signals) and real TIMIT speech data. They observed significant oscillations of the mel scale cepstral coefficient (MFCC) based trajectory features.

The analysis mentioned above closely corresponds to the instability that we observed on the display of DCTC features through the Visual Speech Display system.

Hu and Barnard (1997) also showed that formant trajectories are significantly smoother than trajectories of mel scale cepstral coefficients (MFCC). However, the classification experiments that they performed did not result in better performance using formant trajectory features versus cepstral coefficient features despite the smoothness and smaller variances of formant features. They finally claimed that adding additional information, such as the average bandwidth and contextual features (i.e. the average formant location of the three frames to the left of the left boundary and three frames to the right of the right boundary of the segment), could improve the overall classification performance based on formant features. The performance could not be improved by adding this information with cepstral coefficient based features. It seems that the additional spectral information contained in MFCCs accounts for at least some performance differences observed in experiments.

In summary, DCTC features, as a type of cepstral coefficient features, could have inherently unstable characteristics over time, which may cause the observed fluctuations over time. Therefore, this is also a possible "noise" source besides the noise from PC components, as discussed in the previous section.

2.3.5 Experimental Efforts To Mitigate Feature Variability

A group of experiments were done in an attempt to reduce the DCTC feature variability. Before these procedures are described, a definition of parameters is given below. In describing the experiments, symbols in the parentheses are used to represent their corresponding parameters.

Segment_Time (S_t): Time duration of a speech signal segment.

Frame_Time (F_t): Time duration of a frame extracted from a speech segment.

Frame_Space (S_f): Time interval between adjacent frames. Equation 2.5 shows the relationship between the starting time of adjacent frames and S_f.

$$\boldsymbol{T_{i+1} = T_i + S_f} \tag{2.5}$$

where T is the starting time of a specific frame. For example, if frame #i starts at 50ms and S_f is 15ms, frame #($i+1$) starts at (50+15)ms.

The relationship between F_t and S_f is: if $F_t=S_f$, there is no overlap between adjacent frames; if $F_t > S_f$, there is overlap between adjacent frames, and the length of overlapped part equals (F_t-S_f).

FFT_Length(N_{fft}): Length of FFT in number of samples.

Block_Length(*BL*): The length of a block in number of frames.

Block_Space(S_b): The number of frames between adjacent blocks. The following equation indicates the relationship between the starting frame number of adjacent blocks and S_b.

$$\boldsymbol{B_{j+1} = B_j + S_b} \tag{2.6}$$

where B is the starting frame number in a specific block. For example, if block #j starts at frame #k and S_b=3, block #($j+1$) starts at frame #($k+3$).

The relationship between *BL* and S_b is: if $BL=S_b$, there is no overlap between adjacent blocks; if $BL>S_b$, there is overlap between adjacent blocks, and the length of overlapped part equals (BL-S_b).

Warp_Factor(*W*): Warping factor when frequency warping is used to give non-uniform frequency resolution.

Window_Length(L_w): Length of smoothing window in number of frames.

With all these parameters defined above, several experiments were done in order to test if the adjustment of front-end processing parameters would help reduce DCTC variability. The results of these experiments are illustrated and analyzed through a group of figures as shown below.

Note that the following five DCTC feature plots, from Figure 10 through Figure 14, were all generated with Matlab programs. All features were computed through the revised real-time VSD system and based on the same vowel token from the database of ODU Speech Communications Lab.

Figure 10 shows a plot of the first 4 DCTC features of the vowel “ah” uttered by a male speaker. A total of sixteen 15-ms frames were extracted from the center of the token, with no overlap between frames. Note that for this figure, no frequency warping, time smoothing on the spectrum or time smoothing in block level was used. Expressed with the parameters defined above, S_t=240ms, F_t=15ms, S_f=15ms, BL=S_b=1, N_{fft}=512, W=0, and L_w=0.

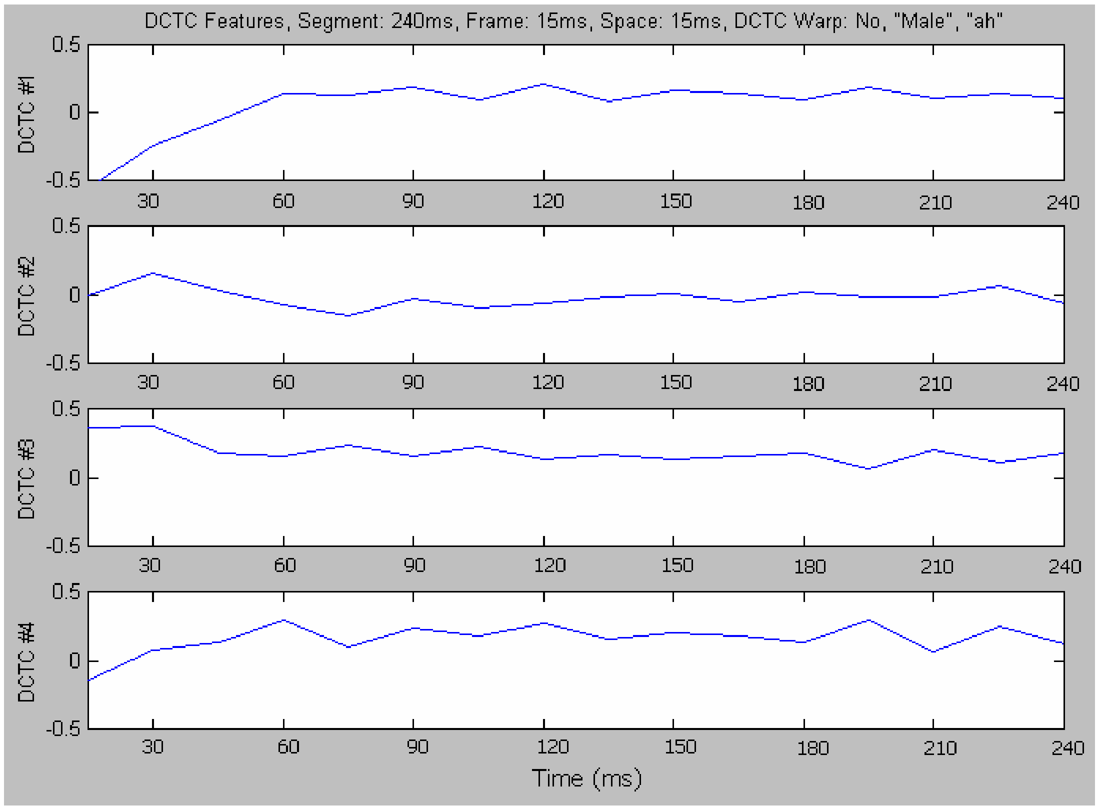


***Figure 10.* The first 4 DCTC features of token "ah" over 240 milliseconds (16 frames)**

The first experiment tested the contribution of frequency "warping" to reducing the DCTC variability. Frequency warping used here actually produces non-uniform frequency resolution and increases the resolution in low frequencies, since we believe that important spectral information is concentrated in low frequencies rather than high frequencies. Figure 11 shows a plot with appropriate frequency warping. The warping factor was set to W=0.45, with all the other parameters being the same as in Figure 10. Compared to Figure 10, it is apparent that frequency warping reduces the variability of DCTC features. However, the improvements are not very distinct.

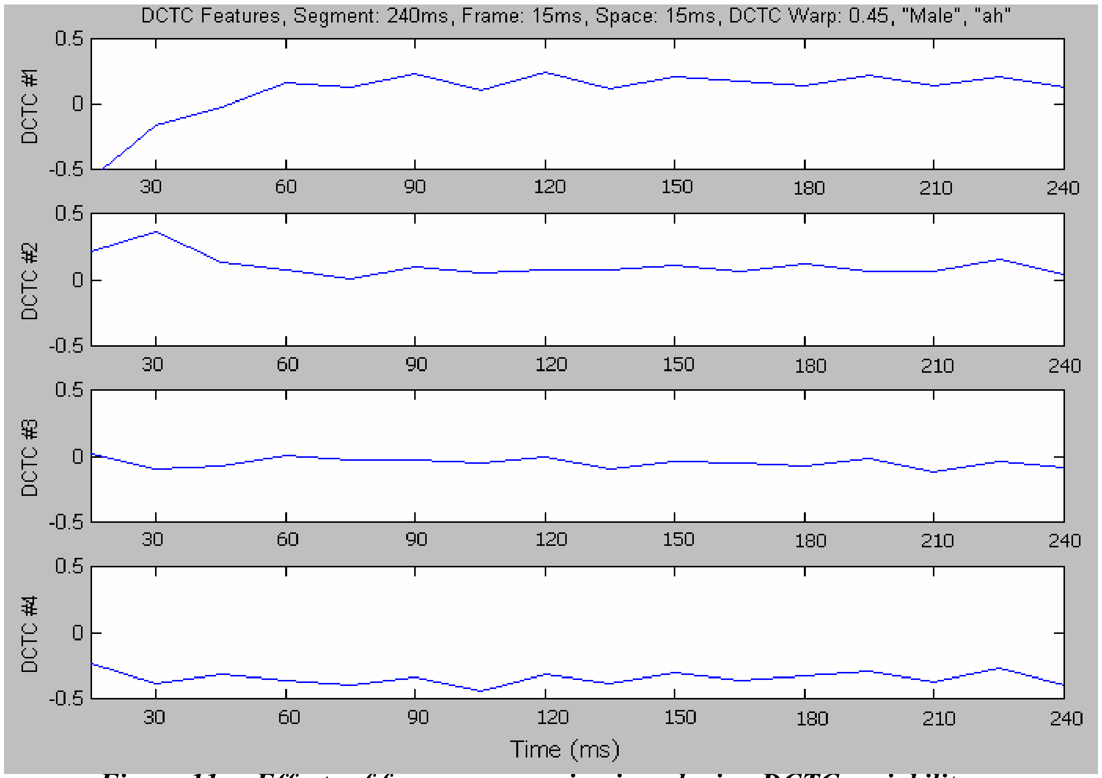


Figure 11.* *Effects of frequency warping in reducing DCTC variability

The second experiment was designed to test the effects of overlap between adjacent frames. In Figure 12, S_t=255ms F_t=30ms, S_f=15ms, BL=S_b=1, N_{fft}=512, W=0.45, L_w=0. Therefore, all parameter settings were set the same as in Figure 11, except that there was a 15-ms overlap between adjacent frames, and the segment length became 255ms due to the overlapping. If we compare Figure 12 and Figure 11, it can be observed that, by using longer frames, the variability of DCTC features is greatly reduced.

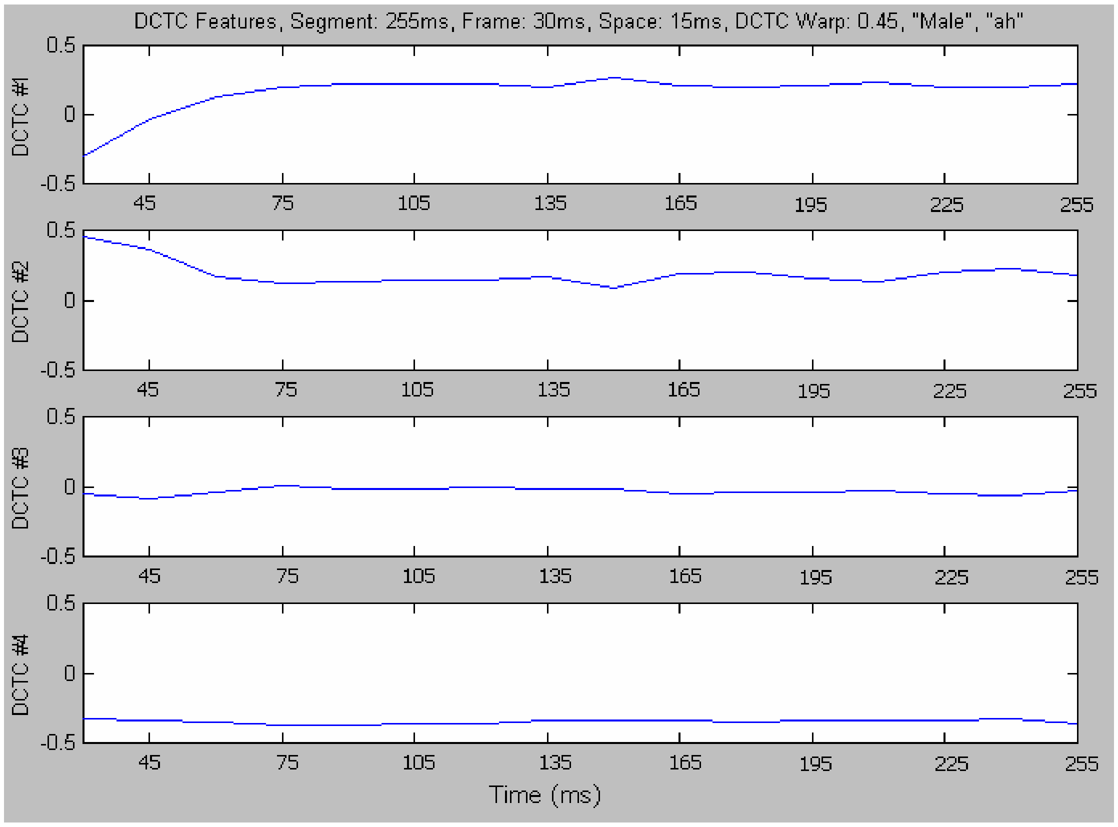


Figure 12. ***Effects of overlapping between frames in reducing DCTC variability***

The third experiment was done to evaluate if time smoothing on the spectrum would help stabilize the DCTC features. Compared to the parameter settings in Figure 12, the only difference in the third experiment is that we set a 5-frame time smoothing window to smooth out the spectrum before the DCTC features were computed, i.e. L_w=5. In our experiments, maximum smoothing, which is explained in detail in the next section, was used. It is shown, in Figure 13, that the DCTC variability was further reduced when proper time smoothing on the spectrum was performed.

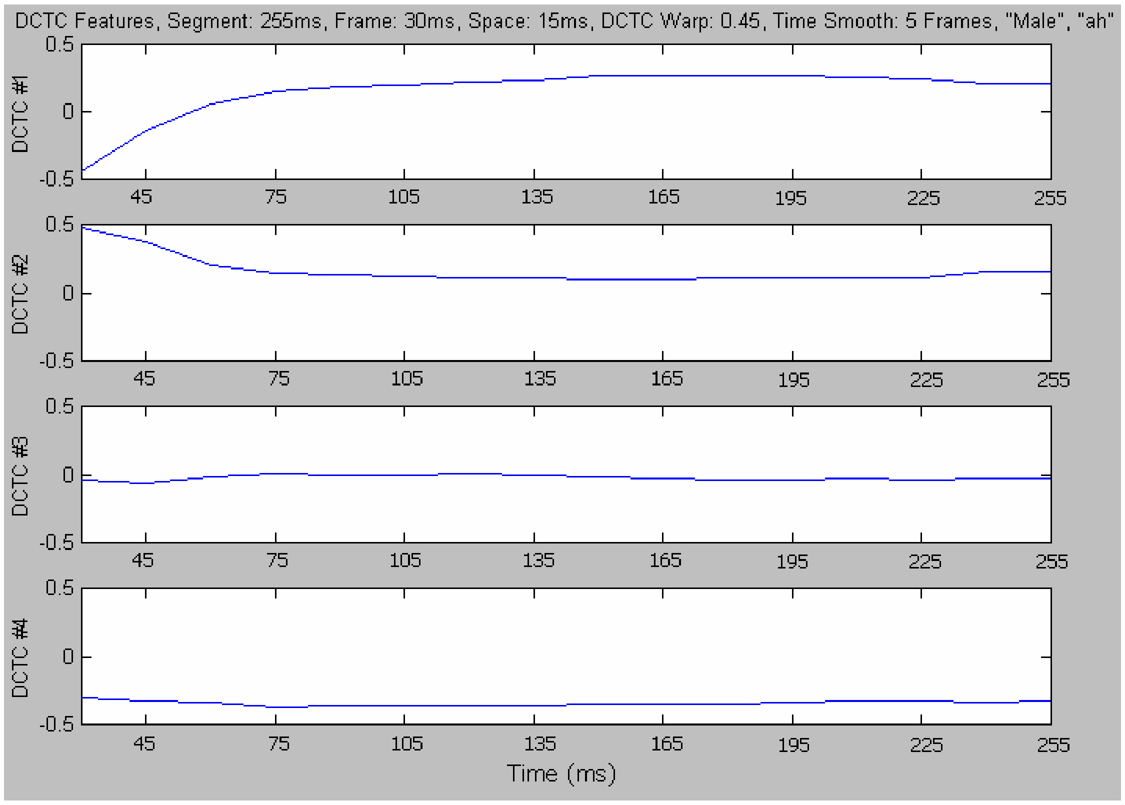


Figure 13. Effects of time smoothing on spectrum in reducing the DCTC variability

Finally, we tested the advantages of using smoothing for the DCS computations, which is in Block Level Processing as described in 2.3.1. Figure 14 illustrates a similar case as in Figure 13, expect for the use of DCS type smoothing. For this figure, BL=5 and S_b=1. Hence, the smoothing was based on a block of 5 frames, with a block spacing of 1 frame. Note that only one DCS was computed from block to block, which was equivalent to the smoothed DCTC over time. However, the benefits of the block level smoothing are not readily seen in Figure 14, since frequency warping and time smoothing over the spectrum already contribute much to reducing the DCTC variability.

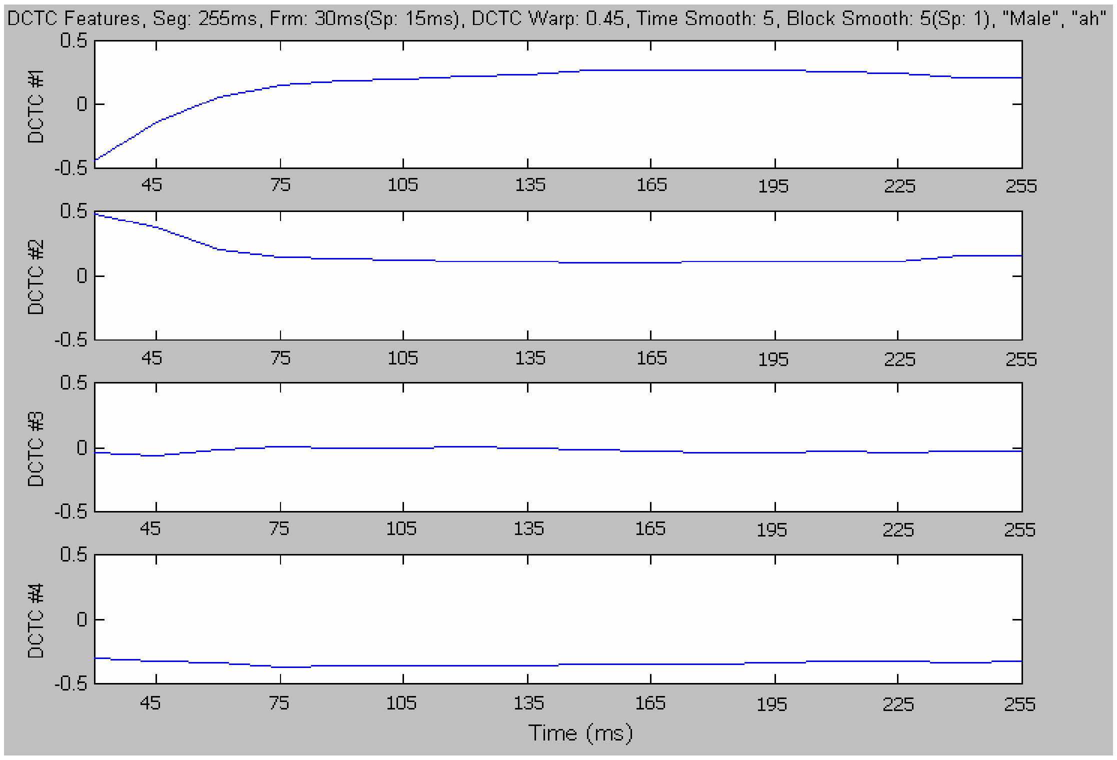


Figure 14. Effects of block level smoothing in reducing DCTC variability

2.3.6 Classification Tests With Time Smoothing On Spectrum

To further explore the approaches to be taken to reduce the DCTC feature variability and improve system performance, some classification tests were done with the real-time version of Visual Speech Display software. The main objective was to compare different time smoothing methods used before the DCTC computations.

Maximum smoothing and average smoothing are both common time smoothing methods for digital signals. Simply put, maximum smoothing finds the maximum value within a smoothing window about a specific sample index and uses it as the smoothed value for that sample. The following formula describes maximum smoothing.

$$\left|X'(i,L)\right|^2 = \underset{L-M\le l\le L+N}{MAX}\left|X(i,l)\right|^2 \tag{2.7}$$

In Equation 2.7, the left-hand side represents the ith smoothed squared spectral component of Frame #L, and the right-hand side stands for the maximum value of the ith squared spectral component of continuous frames around Frame #L, with a window length of (M+N) frames. For real-time processing, N is set to be zero, since smoothing is always based on the current and past frames.

Similarly, average smoothing computes the average value within the smoothing window of (M+N) frames, which is depicted in Equation 2.8.

$$\left|Y'(i,L)\right|^2 = \frac{1}{M+N}\sum_{L-M\le l\le L+N}\left|Y(i,l)\right|^2 \tag{2.8}$$

Time smoothing can be performed before or after log scaling of the spectrum. Maximum smoothing before log scaling is shown in Equation 2.9:

$$\left|X'(i,L)\right|^2 = \log\left(\underset{L-M\le l\le L+N}{MAX}\left|X(i,l)\right|^2\right) \tag{2.9}$$

As a comparison, maximum smoothing after log scaling is shown in Equation 2.10.

$$\left|X'(i,L)\right|^2 = \underset{L-M\le l\le L+N}{MAX} \log\left(\left|X(i,l)\right|^2\right) \tag{2.10}$$

Similarly, average smoothing before or after log scaling is described in the following two equations:

$$\left|Y'(i,L)\right|^2 = \log\left(\frac{1}{M+N}\sum_{L-M\le l\le L+N}\left|Y(i,l)\right|^2\right) \tag{2.11}$$

$$\left|Y'(i,L)\right|^2 = \frac{1}{M+N}\sum_{L-M\le l\le L+N}\log\left(\left|Y(i,l)\right|^2\right) \tag{2.12}$$

In summary, four different time smoothing methods can be performed, i.e. maximum smoothing after log scaling, maximum smoothing before log scaling, average smoothing after log scaling, and average smoothing before log scaling. We wanted to determine how the different smoothing methods affect classification results and which method is the best.

Table 1 shows the classification results from a sample experiment with maximum smoothing and average smoothing, which were both implemented after the log spectrum was computed. These results were obtained under the condition that S_t=2000ms, F_t=30ms, L_w=10 frames. There were 24 speakers in the training set of a specific gender/age group (i.e. Male, Female, Child), and 8 speakers in the test set. The "general" test set included 24 speakers from all three gender/age groups. Three iterations of each vowel by each speaker were used. As shown in Table 1, there is not much difference between the two smoothing methods. In general, we used maximum smoothing since it emphasizes spectral peaks, which are believed to carry most of the speech information.

Gender Group	**Test Recognition Rate (Maximum Smoothing after Log Spectrum is computed)**	**Test Recognition Rate (Average Smoothing after Log Spectrum is computed)**
Male	**89.5%**	**89.2%**
Female	**92.1%**	**92.2%**
Child	**86.2%**	**85.9%**
General Speakers (Male, Female, Child)	**81.2%**	**80.5%**

Table 1. Classification performance of different smoothing methods after log scaling

For maximum smoothing, the index of the maximum value before or after log scaling should be the same due to the monotonicity of the log operation. Hence, it should make no difference whether maximum smoothing is done before or after log scaling. In other words, maximum smoothing before log scaling and maximum smoothing after log scaling are the same method. As for average smoothing, however, there could be minor differences.

Our experimental observations matched the analysis stated above. Table 2 contains the classification results with maximum smoothing and average smoothing before log scaling. If we compare Table 2 with Table 1, we can see that there was no difference in the results between maximum smoothing before log scaling and maximum smoothing after log scaling. For average smoothing, only a slight difference was seen between smoothing before log scaling and smoothing after log scaling. Note that the training set and test set in this experiment were the same as in Table 1.

Gender Group	**Test Recognition Rate (Maximum Smoothing before Log Spectrum is computed)**	**Test Recognition Rate (Average Smoothing before Log Spectrum is computed)**
Male	**89.5%**	**89.2%**
Female	**92.1%**	**92.0%**
Child	**86.2%**	**85.9%**
General Speakers (Male, Female, Child)	**81.2%**	**80.8%**

Table 2. Classification performance of different smoothing methods before log scaling

A group of tests were also done in order to determine if longer windows would help improve classification performance. Figure 15 shows the training classification results (for the Bargraph Display) based on a proprietary speech database defined and established in Old Dominion University's Speech Lab. This figure clearly shows that the classification rate increases as the smoothing window length increases. However, if the window is too long, the system response is very slow, which is undesirable for the real-time display. Therefore, an appropriate smoothing window length needs to be used for acceptable system response time.

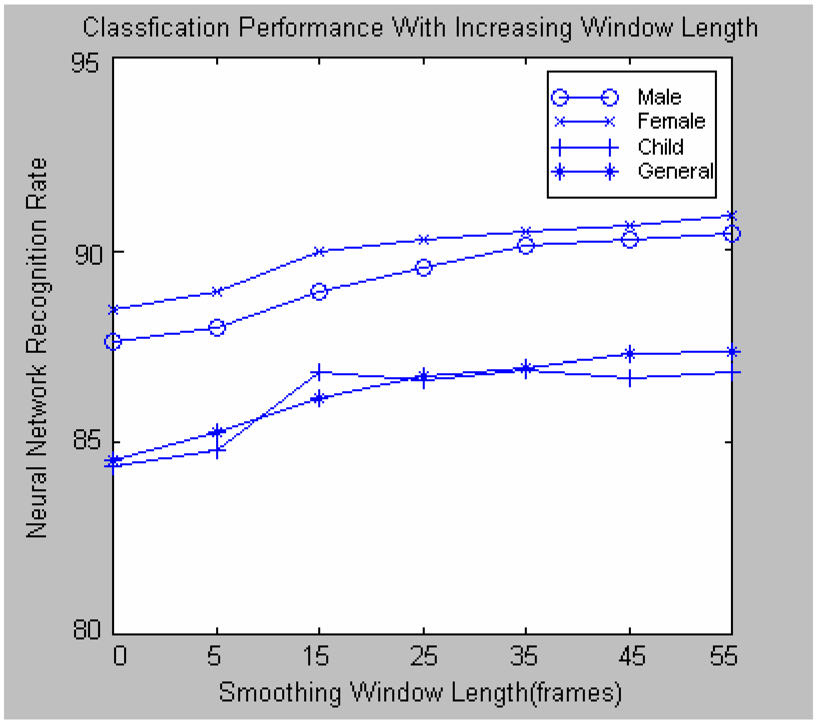


Figure 15. Classification results based on ODU Speech Lab vowel database

2.4 Conclusions

The variability of DCTC features observed in the Visual Speech Display system may be a combination of two sources: noise of the PC components and the inherently unstable nature of cepstral coefficient based (i.e., DCTC) trajectories. It seems that the combined effects are rather complex, and both noise sources cannot be easily and directly eliminated.

Although the variability of DCTC features may not contribute much to performance degradation, we still would like to have a stable display that gives better visual effects. The experiments of this section showed that the DCTC variability could be reduced through proper adjustment of some processing parameters.

Three types of time smoothing methods were tested in our experiments: overlap of signal frames, time smoothing of log spectra before DCTC computations, and time smoothing of DCTC features in the block level. All of these methods showed positive effects in reducing the DCTC instability.

To be more exact, the following approaches were found to be effective:

1. It is always helpful if we use appropriate overlapping between adjacent frames when they are extracted from the speech segment. This method reduces the temporal variations from frame to frame, which contributes to the stability of DCTC features. Overlap is typically set to be half of the frame length. For example, if the frame length (F_t) is 30ms, the frame space (S_f) should be set to 15ms, which generates a frame overlap of 15ms.
2. Utilizing proper frequency warping, typically with a warping factor (W) of 0.45, also helps reduce the DCTC feature variability.
3. When DCSs are computed from DCTC features in the block level processing, the overlap between adjacent blocks helps increase the DCTC stability. Typically, if we use a block length (BL) of 5 frames, it is desirable that the block space(S_b) is set to be 1 frame, thus generating a block overlap of 4 frames.

4. Time smoothing of the log magnitude spectrum is very helpful for improving both DCTC stability and classification results. Since maximum smoothing before log scaling is the same method as maximum smoothing after log scaling, we actually tested three time smoothing methods in our experiments. Maximum time smoothing after log scaling of the spectrum appears to be the best method, as compared to average smoothing before or after log scaling. Note that the smoothing window cannot be too long, since longer windows may greatly slow down the response of display and impair the real-time characteristics of the system. A smoothing window length of 10 to 20 frames is appropriate.

CHAPTER III

PITCH-SYNCHRONOUS DCTC FEATURE COMPUTATIONS

3.1 Overview

Most conventional spectral analysis methods, including those used for the experiments reported in the last chapter, use a window to extract a segment for processing. However, windowing results in spectral smearing, which impairs spectral resolution. In this research, we are interested in exploring a processing method that eliminates or at least reduces window effects. This chapter describes a new DCTC feature computation algorithm based on pitch-synchronous analysis. No windowing is needed in this new method, since processing is based on one or more pitch periods extracted from the speech signal.

In 3.3, we analyze the negative effects of conventional windowing method for steady-state vowel processing; we also explore the advantages and feasibility of utilizing pitch periodicity for spectral analysis.

With a system framework introduced in 3.4, 3.5 explains major functional blocks in the framework, among which three important modules are emphasized. The pitch period estimation method, which is the fundamental for the entire system, is explained in 3.5.2. In 3.5.3, a non-linear frequency scaling method is given in search of a general spectral pattern across different gender/age groups, which makes possible a general classifier for these groups (Male, Female and Child). Since the information contained in a single period may not be sufficient for spectral analysis, multiple periods of speech are extracted and used, as is described in 3.5.4. Time-domain resampling is described in 3.5.5 as a method of reducing spectral smearing when multiple periods are extracted.

3.2 Pitch Periodicity In Voiced Speech

Generally, speech signals are generated by the vibration of the human's vocal cords, which are stretched by muscles in the larynx. Vocal cords vibrate when air is forced through them. The resultant sounds are in the form of a pulse train, which is rich in harmonics. The repetition rate of these pulses determines the pitch, or the fundamental frequency, of the voice (Haddad and Parsons, 1991).

As stated in Haddad and Parsons (1991), the vibration of the vocal cords is termed phonation. Voiced speech is generated by phonation. The speech becomes unvoiced when phonation does not occur. It has been observed that the voiced speech shows quasi-periodicity, but not strict periodicity, over time, since its contents tend to vary continually. Pitch and its harmonics, which are commonly used to characterize periodic signals, can also be utilized to distinguish between quasi-periodic voiced speech.

Steady-state vowels are a type of voiced speech. Figure 16 shows a steady state vowel sound of "ah" uttered by a male speaker. The quasi-periodicity is obvious. If we carefully inspect the signal, we also find that the details in the waveform, such as peak amplitude, peak shape and period length, show slight variability of over time.

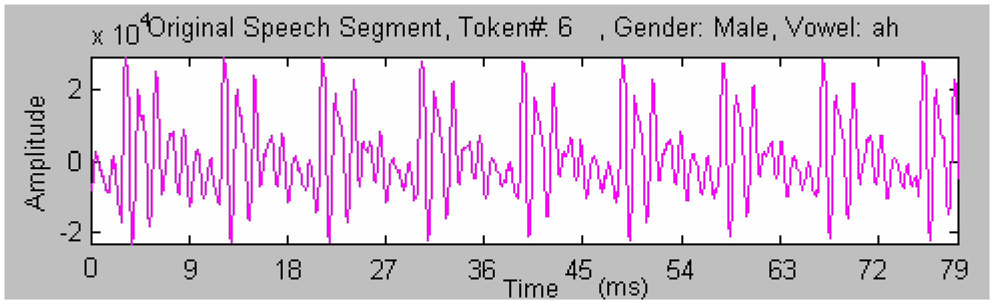


Figure 16. ***A segment of the vowel sound "ah" uttered by a male speaker***

3.3 Eliminating Windowing Effects

As stated at the beginning of this chapter, conventional spectral processing methods use windows to extract speech segment for analysis. However, this may not be a good processing method for voiced speech, such as vowels. There are two major problems.

First, if windowing is used for the spectral processing of vowels, frequency components do not line up with the fundamental and its harmonics, but are spaced at intervals of F_s/N, where F_s is the sampling rate and N is the FFT length. However, for the case of periodic signal such as steady-state vowels, the spectral information is contained at the fundamental frequency and its harmonics. Hence, the frequency components acquired by window-based processing are not entirely "correct" for vowels.

The second problem lies in the fact that windowing inevitably results in spectral smearing, which reduces spectral resolution. Hence, even if the frequency components could be accurately measured, their values might not be correct due to spectral smearing.

The basic idea, used throughout this section, is that the harmonics of periodic signals line up precisely with FFT samples, provided that the FFT length contains an integer number of cycles of the fundamental. This can easily be shown as follows.

Assume that a steady-state vowel segment has M periods, and the FFT length for this segment is N, where N is chosen to be equal to the length of M periods. Let the period be T, the fundamental frequency be f_0 and the sampling rate be F_s, it must be that

$$T = \frac{N}{M} \tag{3.1}$$

and

$$\frac{f_0}{F_s} = \frac{1}{T} \tag{3.2}$$

From Equation 3.1 and 3.2, the following equation holds:

$$\frac{1}{f_0} = \frac{N}{M} \times \frac{1}{F_s} \tag{3.3}$$

From Equation 3.3, we also have:

$$F_s = \frac{N \cdot f_0}{M} \tag{3.4}$$

Since the spacing between FFT samples is

$$\Delta f = \frac{F_s}{N} \tag{3.5}$$

we finally have

$$\Delta f = \frac{f_0}{M} \tag{3.6}$$

In particular, as shown in Equation 3.7, if there is only one period for processing, the frequency spacing between adjacent FFT samples is equal to the fundamental frequency. In other words, the FFT samples are exactly coincident with the fundamental and its harmonics.

$$\Delta f = f_0 \tag{3.7}$$

If M is greater than 1, then every Mth FFT sample is exactly coincident with a spectral harmonic.

Figure 17 clearly illustrates the windowing effects and the benefits of pitch-synchronous frequency range in reducing windowing effects. The top plot is the time-domain signal used for spectral analysis. The middle plot shows the superimposed spectra of 4 overlapped 30-ms frames (15-ms overlap), which were Hamming windowed before the spectral processing. It can be seen from this plot that the harmonics, especially in the middle to high frequency bands, are not distinct and vary from frame to frame. This implies spectral smearing due to the windowing effects. In contrast, the bottom plot, which was generated with pitch dependent frequency range, shows very sharp and clear harmonics over the whole frequency band. The pitch period for the bottom plot was 98.04Hz, and 30 harmonics were plotted in the range of [100Hz, 3000Hz]. This further illustrates that for pitch-synchronous analysis, FFT samples precisely coincide with harmonics.

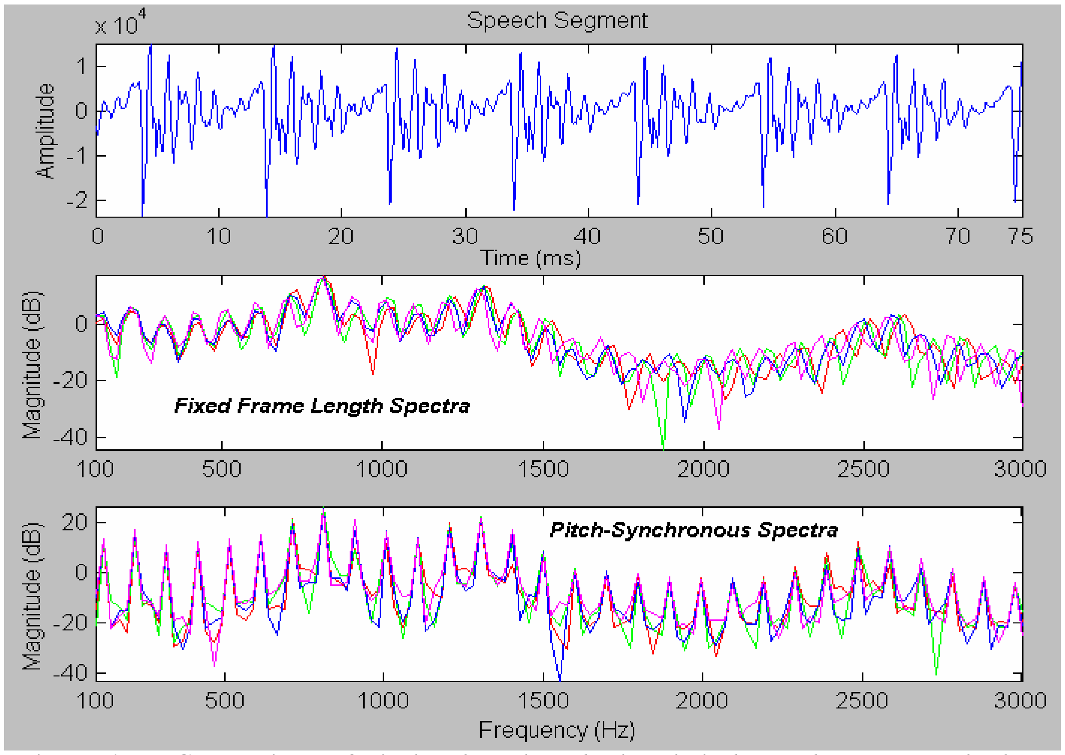


Figure 17. Comparison of window based method and pitch-synchronous method

The objective of this pitch-synchronous method is to eliminate the two negative effects caused by windowing and take advantage of sampling the spectrum at exactly the "right" places without spectral smearing. It was hoped that this type of spectral analysis would lead to more stable DCTC features, and also to higher automatic vowel classification performance.

The difficulties with pitch-synchronous analysis are that fundamental frequency must be determined very accurately, and the speech signal must then be resampled so that each frame of speech data contains an integer number of periods, and the total length equals a power of 2 (256, 512, etc.). Furthermore, this must be done using an interval that typically contains several pitch cycles, for which the pitch period is typically slowly changing.

The important part of this chapter was to investigate techniques for pitch-synchronous vowel analysis, and determine whether any of these are superior to conventional window based spectral analysis. As another issue, since pitch was required to do this analysis anyway, pitch based speaker normalization was also investigated as a method for improving vowel classification.

3.4 System Framework

A framework of the pitch-synchronous DCTC feature computation system is illustrated in Figure 18. In our experiments, two operation modes, Mode 1 and Mode 2, were implemented to simulate different processing approaches.

Processing in Mode 1 was based on a large segment consisting of multiple periods. In Mode 2, processing was based on the average spectrum of multiple single periods, each of which was extracted from a short frame. Pitch period estimation methods implemented in the two modes were also different. In particular, the pitch period estimation method for Mode 1 was more suitable for long intervals, whereas the pitch period detection method used in Mode 2 was more suitable for processing based on short frames.

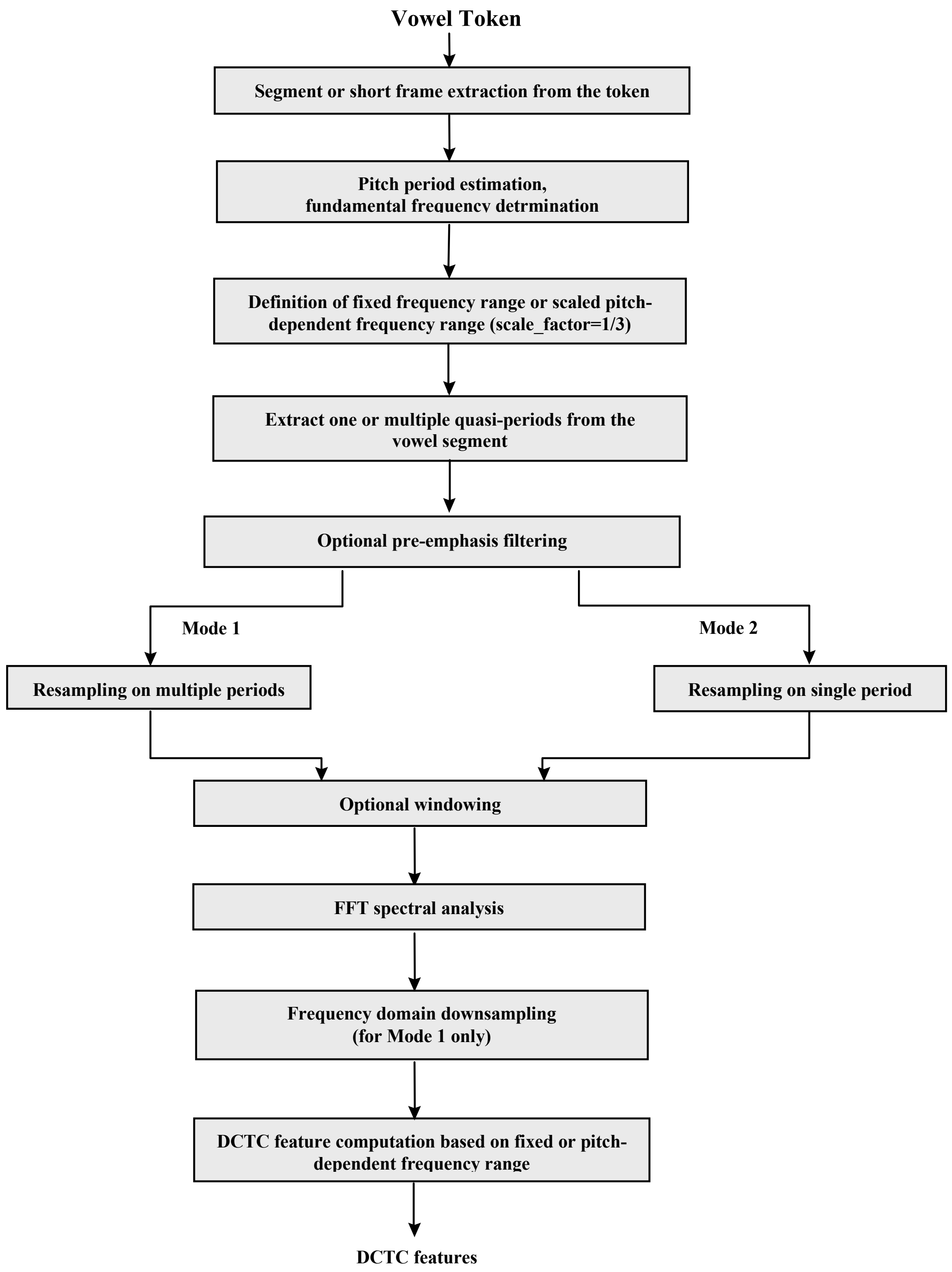

Vowel Token
Segment or short frame extraction from the token
Pitch period estimation, fundamental frequency detrmination
Definition of fixed frequency range or scaled pitch-dependent frequency range (scale_factor=1/3)
Extract one or multiple quasi-periods from the vowel segment
Optional pre-emphasis filtering
Mode 1
Mode 2
Resampling on multiple periods
Resampling on single period
Optional windowing
FFT spectral analysis
Frequency domain downsampling (for Mode 1 only)
DCTC feature computation based on fixed or pitch-dependent frequency range
DCTC features

Figure 18. ***Diagram of pitch-synchronous DCTC computation system***

In Mode 1, a large segment is extracted from the central portion of a steady-state vowel token. By computing the autocorrelation, the pitch period and the fundamental frequency are determined based on the whole segment. Multiple periods of samples are then extracted from the original segment. Time-domain resampling is performed on the multiple periods to extend them to an appropriate FFT length, which should be a power of 2. The significance of resampling is that it extends existing periods to some desirable FFT length while maintaining the periodicity. The FFT analysis is done with the resampled periodic signal. Since the resultant spectrum is based on multiple periods, appropriate frequency downsampling must be done to select only those frequency components that line up with the fundamental and its harmonics. Finally, DCTC features are computed based on the spectral components and pre-computed cosine basis vectors over a fixed frequency range or pitch dependent scaled frequency range.

Mode 2 is similar to Mode 1 except for the pitch period estimation and some details of spectral processing. Since Mode 2 is designed for pitch period estimation from short frames, the algorithm of period detection is different from that used in Mode 1. In spectral processing, the spectrum of each individual period is computed separately. Multiple spectra are then averaged. Note that, as shown in 3.3, no further frequency downsampling is required in this mode, since the interval between the FFT samples is exactly equal to the fundamental frequency. The final DCTC features are computed from the averaged spectra.

For the purposes of comparison, an extra “Control Mode” was also implemented using the conventional window based processing. In this mode, pitch period estimation was not required; no time-domain resampling and frequency downsampling were used. Instead, frames of fixed length were directly zero-padded to the desired FFT length and then windowed (using a Hamming or Kaiser window) before the spectral analysis.

In addition, a bandpass pre-emphasis filter, with a center frequency of 3200Hz given that the sampling rate was 11025 samples/sec, was generally used for all modes to filter out unwanted noise in both low and high frequencies.

3.5 Main Functional Blocks

This part describes the major functional blocks in the diagram of the pitch-synchronous DCTC feature computation system. A Matlab software implementation of this system is explained in the next chapter.

3.5.1 Segment Extraction

This block is simple and straightforward. When we have a vowel token for steady-state analysis, it is desirable that we extract a segment from about the center of the token, since this part should be the most stable. Vowel token files are digitally recorded versions of vowel sounds, which may have different formats according to different standards followed. For example, a TIMIT format file is comprised of two parts: a file header containing information about the token, and a token body that stores the speech samples. If we are dealing with token files, we need to separate the header from the token body so that the steady-state part of the token is accurately extracted. Figure 16 in 3.2 gives an example of a segment extracted from a vowel token.

3.5.2 Pitch Period Estimation

This is the most critical and difficult part of the whole system. This step is important because all the subsequent processing, especially the FFT analysis, is heavily dependent on the accuracy of locating individual pitch periods. However, this step is also difficult since the signal’s periodicity is slowly varying with time.

In this research, we implemented two methods to detect pitch periods. The first method was designed for processing based on a large segment containing multiple periods (Mode 1); while the second method was developed especially for processing based on a short frame (Mode 2), which generally contains only one or two periods.

The theoretical basis of the first method (Mode 1) is autocorrelation theory, which is defined in Equation 3.8.

$$\Phi_{xx}[l] = \sum_{n=-\infty}^{\infty} x[n]x[n+l] \tag{3.8}$$

In this equation, $\Phi_{xx}[l]$ represents the correlation of signal $x[n]$ and a shifted copy of itself with a time difference of l. It can be shown that when l equals zero, the autocorrelation value reaches the local maximum. In particular, if $x[n]$ is periodic, $\Phi_{xx}[l]$ reaches the local maximum when $l=kN$, $k=1,2,3...$, where N is the period of the signal.

In this way, we can compute an autocorrelation sequence for a multi-period segment. Peak values are expected to appear at indices that are multiples of the quasi-period. Generally, the index of the first peak value in the autocorrelation sequence can be regarded as the pitch period. Figure 19 shows the autocorrelation sequence of a segment extracted from the center of vowel token "ah" uttered by a male speaker. The vertical line marks the first peak, whose index is considered as the pitch period.

Figure 19 also shows that the periodicity of vowel speech varies slightly with time (i.e., the signal is not precisely periodic). As we know, peaks in the autocorrelation sequence correspond to multiples of period in time. If the signal is strictly periodic, the value of autocorrelation peaks should remain the same at all integer multiples of the pitch period. However, in Figure 19, it can be seen that the peaks become smaller at larger lag values, which indicates that the pitch period changes over the interval of the segment.

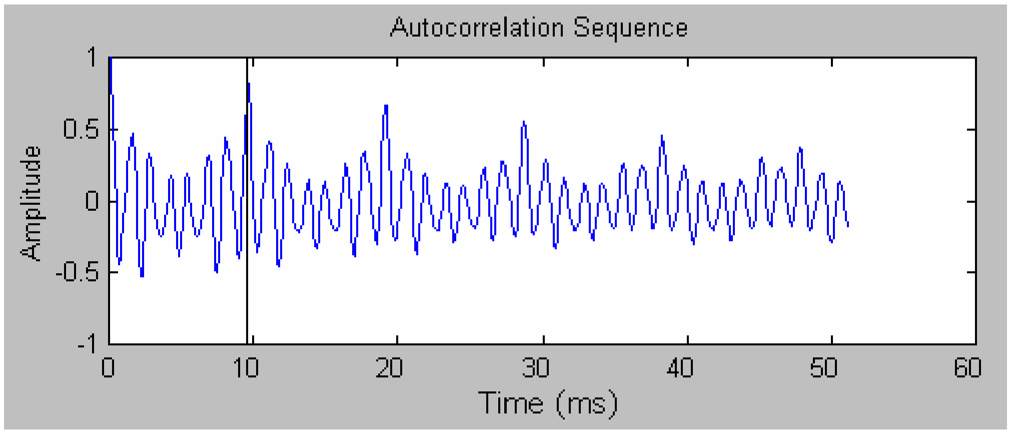


Figure 19. Autocorrelation sequence of a vowel token segment

The second pitch period estimation method is suitable for processing of short frames (Mode 2). In many cases, short frames are desirable because the spectral variability is not significant within a short period of time. However, the autocorrelation computation, as mentioned for the first method, does not perform well for very short segments. An appropriate definition is given in Equation 3.9.

$$s(k) = 2 \cdot \frac{\sum_{j=1}^{N-k} x[j] \cdot x[j+k]}{\sum_{j=1}^{N-k} \left(x[j]^2 + x[j+k]^2 \right)} \tag{3.9}$$

In Equation 3.9, *s(k)* is the resultant normalized correlation sequence, with $1 \le k \le N/2$, and N is the length of the short frame. It is clear that when k equals to the length of a period, *s(k)* is maximized. In other words, the index value that corresponds to the maximum of *s(k)* is regarded as the pitch period. Equation 3.9 was motivated by a period estimation method in Talkin (1995), in which local maximum values in the correlation sequence were found to detect the pitch period. Betancourt and Antrobus (1998) also researched a similar method. In Mode 2, however, we computed correlation sequences with peak values in order to take advantage of the same pitch period detection algorithm as used in Mode 1.

Both pitch period estimation methods described above were found to be effective and robust in our experiments, when vowel tokens were produced with acceptable periodicity. However, these two methods might not be successful in estimating periods for tokens with poor periodicity. In the Matlab implementation described in the next chapter, some additional processing is described to handle cases when the periodicity was not readily apparent.

3.5.3 Scaled Pitch-Dependent Frequency Range

The idea of using a scaled pitch-dependent frequency range originated from the "sensory reference" concepts, which were introduced by Peterson and Barney (1952). Miller (1989) further developed these concepts in search of an approach to eliminate the effects of talker age and gender. Although the use of a pitch-dependent range is secondary to the main goal of comparing spectral analysis using fixed window lengths versus a pitch period dependent length, it does fit the more general theme of determining methods to improve the accuracy of vowel classification. Since pitch has to be computed anyway for the pitch-synchronous analysis, the pitch dependent frequency range is a natural secondary issue to investigate. Finally, however, note that this normalization can also be used (as a control), for the case of window based spectral analysis, but with computation of pitch only for this explicit purpose.

In Miller (1989), the sensory reference (SR) was defined as a function of the geometric mean of the talker's fundamental frequency, such that the average ratio (SF1/SR), where SF1 refers to the first sensory formant, across all glottal-source spectra would be independent of the talker's age and sex. To estimate the form of the desired function, the geometric mean of the first formants (GMF1) and fundamental frequencies (GMF0) reported in Miller (1989) were separately calculated for men, women and children. It was found that the equation

$$k = \log\left(\frac{GMF1}{GMF0^{1/3}}\right) \tag{3.10}$$

results in a value of k that is nearly constant across talkers. Note that the exponent $1/3$ in the denominator was experimentally determined to be optimal. The sensory reference was defined as below:

$$SR = 168\left(\frac{GMF0}{168}\right)^{1/3} \tag{3.11}$$

where the value 168 is the geometric mean of adult male's average pitch 125Hz and adult female's average pitch 225Hz. The third variable defined in the auditory-perceptual space was

$$y = \log(\frac{GMF1}{SR}) = \log[(168)^{2/3}(\frac{GMF1}{GMF0^{1/3}})] \tag{3.12}$$

Note that Equation 3.12 was acquired simply by substituting Equation 3.11 in the denominator of Equation 3.10. Through this formulation, the value of SR is shifted from the absolute value of 168 to an appropriate value for the current talker, such that the average value y is nearly identical across talker groups.

The implication of the equations and experiments mentioned in the preceding paragraphs is that, if we compare the spectra of the same vowels uttered with different fundamental frequencies (such as the difference between an adult male and an adult female) on a log frequency scale, there is a shift in the spectral patterns proportional to $f_0^{1/3}$. When using a linear frequency scale, the implication is that, if the spectra from different speakers are viewed on ranges so that the starting and ending frequencies are proportional to $f_0^{1/3}$, the variability among speakers for the same vowel is reduced relative to that obtained with a fixed range for all speakers.

In the Old Dominion University Visual Speech System, we have always used a fixed frequency range for spectral processing. However, the fundamental frequency f_0 is quite different for male, female and child speakers. For example, the pitch frequency of an adult female speaker is generally twice as high as that of an adult male speaker. For the case of pitch-synchronous analysis, a fixed frequency range implies different number of harmonics. On the other hand, if we use a fixed number of harmonics, there is a huge difference in the frequency range for speakers from different gender/age group.

Motivated by the “sensory reference” theory summarized above, we explored the implementation of pitch and harmonics related scaled frequency range for the purposes of finding an age and sex independent feature set that could contribute to better classification.

As shown in 3.3, for voiced speech, the intervals between harmonics almost equal the pitch frequency due to the quasi-periodicity. If we consider a fixed number of harmonics, the overall frequency range over the pitch and harmonics can be much different for people from different gender or age group. For instance, if the voice of a male speaker has a pitch frequency of 110Hz, the frequency range over the pitch and 29 harmonics is from 110Hz to 3300Hz. Likewise, if the voice of a female speaker has a pitch frequency of 220Hz, the corresponding frequency range of the pitch and 29 harmonics is from 220Hz to 6600Hz.

We need to do some kind of non-linear frequency scaling in order to reduce the difference. The method used is described below.

First, we defined a standard pitch frequency and the number of harmonics to be considered. Based on this information, we computed two range-limiting coefficients for the non-linear scaling. Let a standard pitch frequency be *SF0*, the number of harmonics be *N*, the lower limit coefficient be *K1* and the upper limit coefficient *K2*, then

$$K1 = \frac{SF0}{SF0^{1/3}} \tag{3.13}$$

$$K2 = \frac{(N+1)\cdot SF0}{SF0^{1/3}} \tag{3.14}$$

where the exponent 1/3 was defined following the "sensory reference" concepts stated above, and was tested in our experiments to be optimal.

With a specific speaker, the scaled frequency range over the pitch and N harmonics is

$$K1\cdot(F0)^{1/3} \le F \le K2\cdot(F0)^{1/3} \tag{3.15}$$

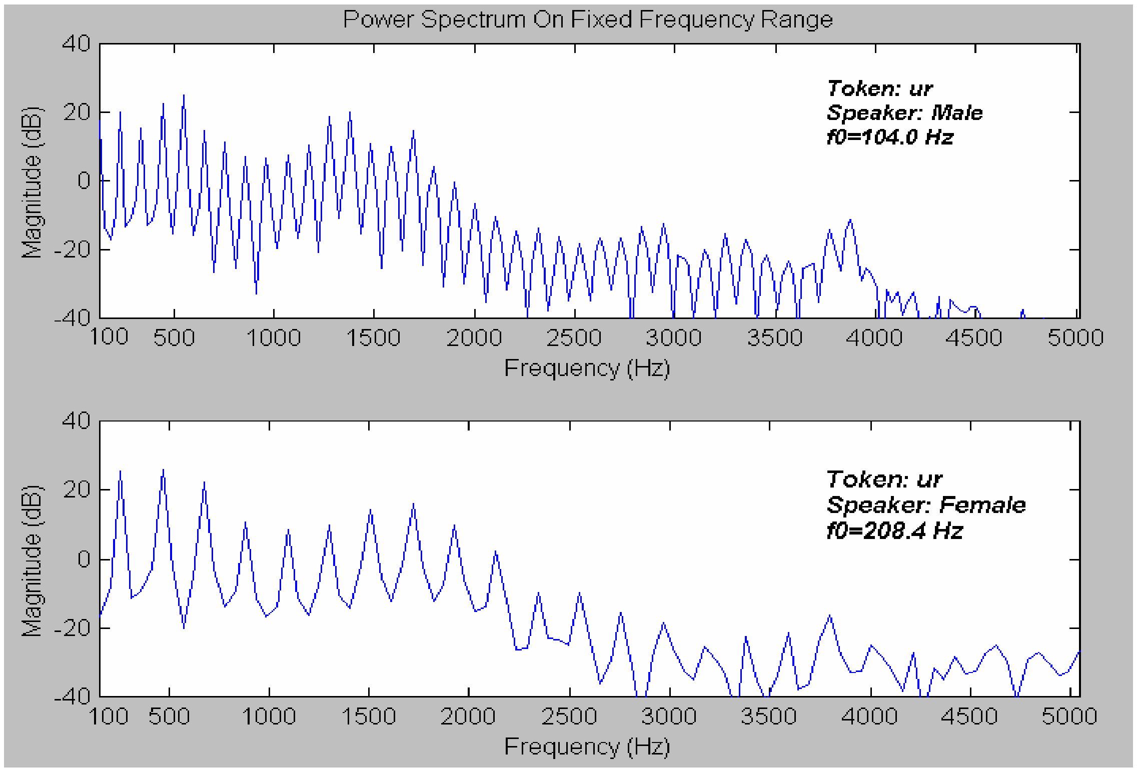


Figure 20. Spectral plots of "ur" of different speakers on fixed frequency range

Figure 20 and Figure 21 show the spectra of "ur" of a male speaker and a female speaker over fixed frequency range and $f_0^{1/3}$ scaled frequency range, respectively. It can be seen that the spectral envelopes of different speakers look more similar to each other in Figure 21 than in Figure 20. This implies that $f_0^{1/3}$ scaled frequency range helps reduce the spectral variability among speakers. Note that the plots in Figure 20 and Figure 21 were generated based on Mode 1, which is described in 3.4.

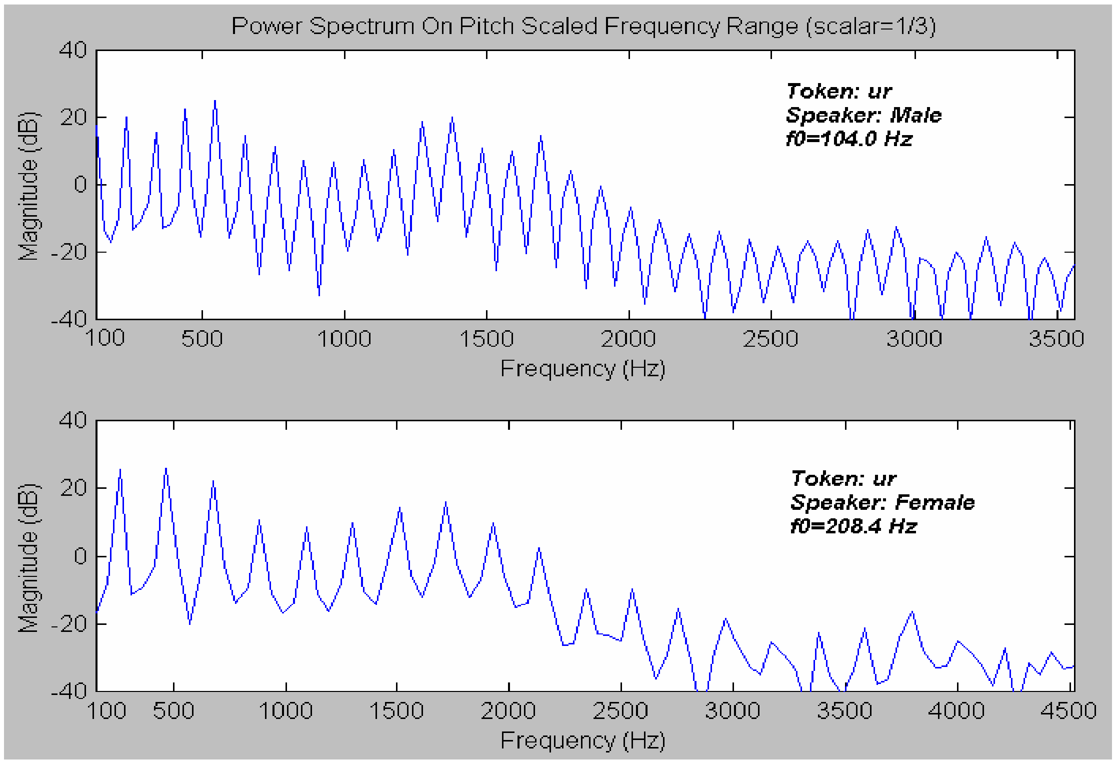


Figure 21. Spectral plots of "ur" of different speakers on scaled frequency range

3.5.4 Multiple Quasi-Periods

Based on the autocorrelation method described in 3.5.2, a single period of speech signal may not carry enough information for processing due to the possibly imprecise estimation of periods. Multiple periods need to be extracted.

Since the "period" of voiced speech is slightly varying over time, it may not be appropriate to get multiple periods simply by extracting a chunk of signal whose length is a multiple number of the estimated period. A method based on "zero-crossing detection" was used in this research.

The procedure of detecting zero-crossings is illustrated in Figure 22.

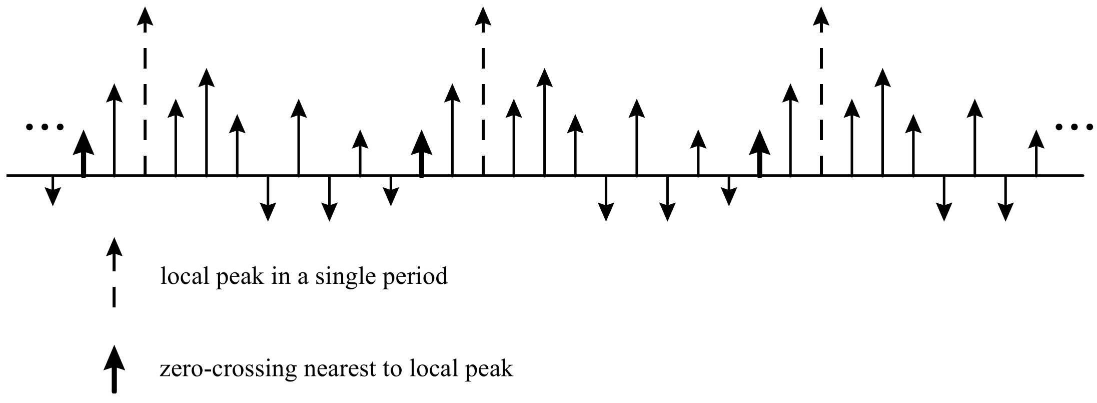

local peak in a single period
zero-crossing nearest to local peak

Figure 22. Detection of zero-crossings

Figure 22 shows a segment of a periodic signal. We first find the local peak in the first period, which is marked by the first dashed-line arrow in the diagram. Then, we detect the nearest zero-crossing in the vicinity of the peak. Since we are dealing with discrete-time signals, zero-crossings may not exactly be zero. Hence, we define zero-crossing as being the first change in signs from + to −. For the case in Figure 22, the first zero-crossing is marked by a bold arrow pointed up. Once we determine the first zero-crossing, we add the length of one period to the first zero-crossing. The second zero-crossing is found in this vicinity. By using the same method, we are able to identify all the subsequent zero-crossings.

Note that for voiced speech signal, the period itself tends to vary slightly over time. Hence, it is extremely difficult to ascertain that periods are clearly separated and extracted. However, the zero-crossing method appeared to be robust when we inspected a large group of speech tokens in our experiments.

3.5.5 Time-Domain Resampling

As we know, a period or multiple periods extracted from a signal segment generally do not have an appropriate FFT length (power of 2), and need to be extended to an FFT length before we do spectral processing. The resampling methods implemented in our experiments are illustrated below.

Method 1: (there was actually no resampling; rather, the signal segment was zero-padded to the proper FFT length. This method was used in the simulation of regular processing, which served as a comparison to the "real" resampling method (Method 2). Figure 23 illustrates the procedure of Method 1. Note that Method 1 was used only in Control Mode, as described in the next chapter.

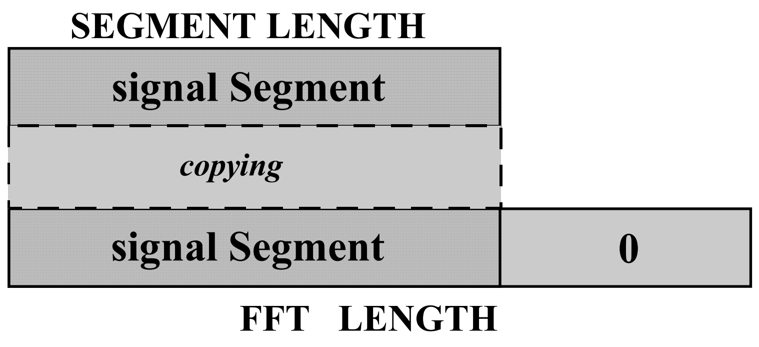


Figure 23. No resampling, simply zero-padding

<u>Method 2</u>: this method was designed for both Mode 1 and Mode 2, which are described in the next chapter. With this method, we processed one or more periods extracted from the vowel and the resampling was performed on a period-by-period basis. Each period was stretched to a desired FFT length through resampling. Since the FFT length is generally greater than the number of samples in a period, we need to "resample" the period with a new sampling rate. In the experiments, we used linear interpolation to determine the resampled value between the original sample values.

We first conducted linear interpolation between samples of a period to obtain an interpolated signal, which was continuous in time and included both sample values and interpolated values. This interpolated signal was then "resampled" at a proper sampling rate to acquire a new sequence of the desired FFT length. This step was repeated on all the periods.

Note that the resampling process may introduce errors. The effects of errors depend on the original sampling rate and interpolation method used. Generally, the higher the original sampling rate is, the more accurate the interpolation; non-linear interpolation is better than linear interpolation. In this research, since a high sampling rate of 11025 samples/sec was used, we decided to perform linear interpolation for computational simplicity and efficiency. Figure 24 illustrates the processing in Method 2.

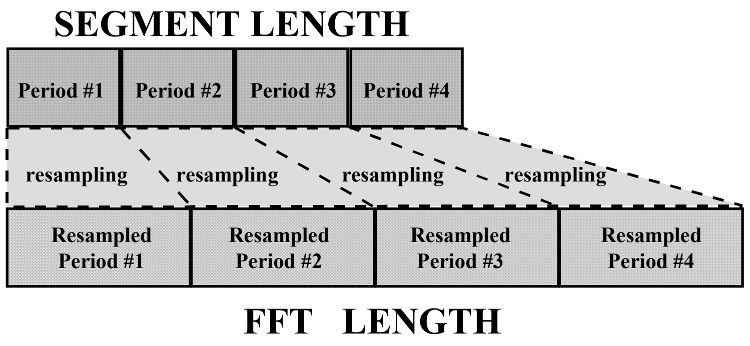


Figure 24. Resampling period by period

3.6 Conclusions

In this chapter, a system framework designed to implement the pitch-synchronous processing method was discussed. We illustrated the important modules in the framework, such as pitch period estimation, time-domain resampling and pitch-dependent scaled frequency range. Two pitch period detection methods, based on the autocorrelation, were designed for processing long frames and short frames, respectively. Spectral analysis results showed that the pitch-synchronous method helps reduce spectral smearing, which is a result of window effects in the regular window-based processing method. We also found that by using ($f_0^{1/3}$) scaled frequency range, the formants in the spectra of different speakers would line up more closely with one another. This would be helpful for the spectral analysis for general speakers.

In the next chapter, a Matlab software implementation of this system framework is described.

CHAPTER IV

MATLAB IMPLEMENTATION OF PITCH-SYNCHRONOUS DCTC FEATURE COMPUTATIONS

4.1 Overview

In order to verify and illustrate the pitch-synchronous algorithm described in the previous chapter, the algorithms mentioned in the last chapter were programmed using Matlab V5.1 for Windows. This chapter first explains the implementation of main functional modules and illustrates the results of important processing steps. Then, classification experiments with the pitch-synchronous method are described and compared to experiments performed using the regular DCTC computation method. Finally, conclusions are reached.

4.1 focuses on the description of major modules in the Matlab application. Classification results based on the pitch-synchronous approach are shown and discussed in 4.2. Conclusions regarding the pitch-synchronous method are given in 4.3.

4.2 Implementation Of Pitch-Synchronous DCTC Computations

The Matlab implementation of the pitch-synchronous DCTC feature computations was based on the system framework described in 3.4. Two segment extraction and pitch period estimation methods were explored. They were defined as Mode 1 and Mode 2, respectively.

In Mode 1, a large segment is first extracted from the center of a vowel token, and the pitch period is determined based on the whole segment. Then, continuous multiple periods are taken out of the segment. Spectral analysis and the subsequent DCTC feature computations are both based on these multiple periods.

In Mode 2, we evaluated another approach for pitch-synchronous analysis. Instead of a single pitch computation for the entire large segment, we used a series of short, overlapped frames from the token center. The period estimation was conducted separately on each individual frame in the series. The estimated periods were revised through median smoothing to reduce the likelihood of any major errors (such as pitch doubling or halving) in the pitch track. Only one period was extracted from each frame for spectral analysis. The spectrum and DCTC features were computed on each frame separately, and the final DCTC feature for a specific token was an average of the DCTC features from individual frames.

What is common to both methods is that we used both pitch-dependent and pitch-independent frequency ranges for analysis and comparison, as explained in the following sections. Note that we use the terms “Mode 1” and “Mode 2” repeatedly in following sections while describing the Matlab implementation.

4.2.1 Generation Of Vowel Token Files

In the simulation, we first used WINSEL2.EXE, which was an application developed in the Old Dominion University Speech Lab, to generate two token list files—one for the training set and the other for the test set--for the final classification. The list files contained paths to all token files in the speech database.

Since the Matlab application was designed to read and process tokens of each vowel separately, the training and test list files generated by WINSEL2.EXE could not be used directly. An extra step was added to create 10 different "sub-list" files for the training set and the test set, respectively.

4.2.2 Segment Extraction

As stated above, two operation modes were implemented separately in order to compare different processing and classification results. As a matter of fact, the segment extraction methods implemented in the two modes were different from each other.

A vowel token file is usually composed of two parts. The first part is the header field, and the second is the data field. The header field provides information about the token, such as token length, header length, sampling rate, and bytes per sample, etc. With the information in the header field, we were able to identify the center of the data field and extract a segment from this portion.

In Mode 1, multiple periods were extracted for spectral analysis and DCTC computations. A long segment, typically 80 milliseconds, was generally taken from the center of a vowel token. This segment would have to be long enough for further extraction of multiple periods. Figure 25 shows an 80-ms signal segment extracted from the center of vowel token "ah."

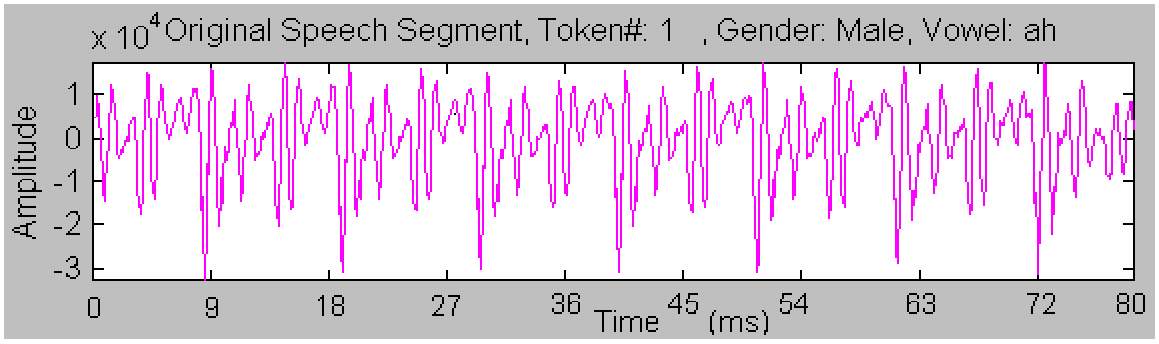
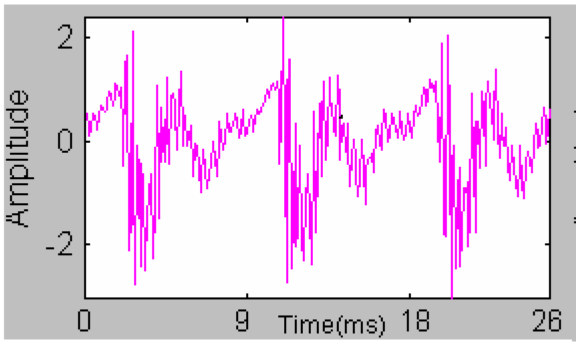
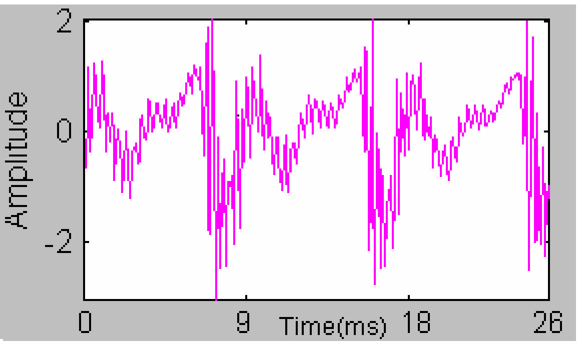


Figure 25. ***80-ms segment extracted from the center of token "ah"***

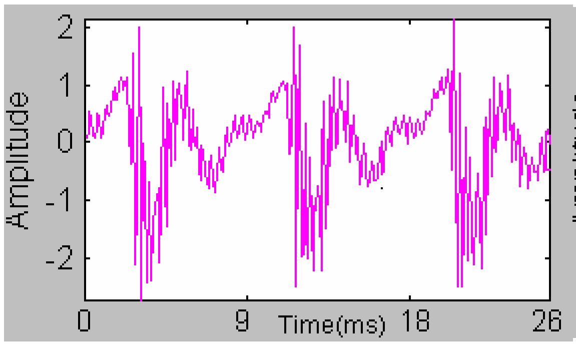
For Mode 2, a long segment (typically 300 ms) was first extracted from the center of a token. This segment, however, was not directly used for pitch period estimation and additional processing. Instead, a number of short frames were further extracted from this segment. By default, we would extract up to 16 frames, with each frame being 26 milliseconds long. In an attempt to increase the correlation between adjacent frames, frames were overlapped, typically by half of the frame length. Figure 26 gives an example of four consecutive 26-ms frames, with a 13-ms overlap, extracted from the same token "ah" as shown in Figure 25.



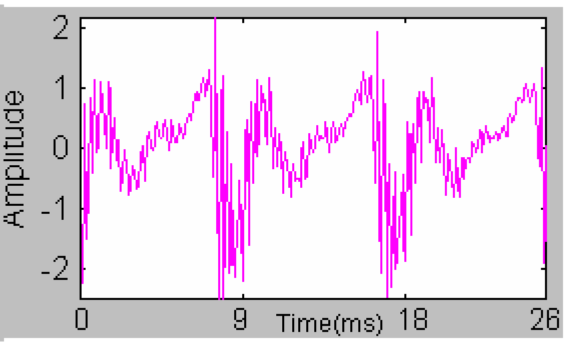
Frame # 1



Frame # 2



Frame # 3



Frame # 4

Figure 26. 4 consecutive frames (26 ms long) with a 13-ms overlap

4.2.3 Pitch Period Estimation

The basic methods for both Mode 1 and Mode 2 were to compute the pitch period based on the autocorrelation calculations (with some variations between the two modes), as was explained in 3.5.2. The actual Matlab implementation was also different between Mode 1 and Mode 2.

The pitch period estimation for Mode 1 was based on the theory that the inverse FFT of the power spectrum of a random input signal is the autocorrelation of the input sequence (Proakis and Manolakis, 1996). The mathematical expression is given in Equation 4.1.

$$\gamma_{xx}(\tau) = \int_{-\infty}^{\infty} \Gamma_{xx}(F) \cdot e^{j2\pi F\tau} dF \qquad (4.1)$$

where γ_{xx} represents the autocorrelation of the input random signal, and Γ_{xx} stands for the power spectral density of the input signal. Since the speech signal can be regarded as a special type of random signal, this equation also applies to our implementation.

An example of the output autocorrelation sequence of an 80-ms signal segment for the token "ah" is illustrated in Figure 27. Since an 80-ms segment is equivalent to 882 samples at a sampling rate of 11025 samples/sec, we chose an FFT length of 1024 points, which resulted in an autocorrelation sequence of 512 points.

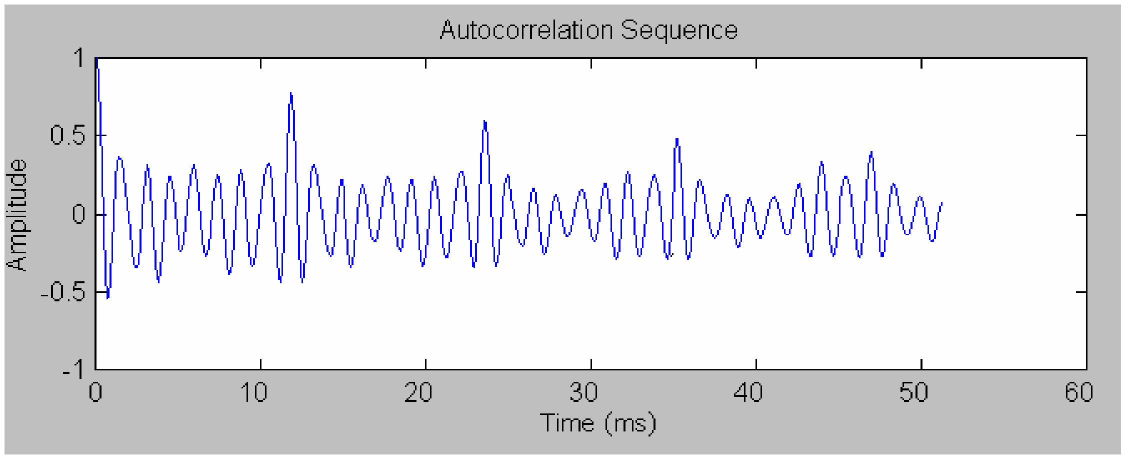


Figure 27. Autocorrelation sequence for an 80-ms signal segment of token "ah"

As we know, the statistical reliability of autocorrelation calculations for the speech signal is better in large segments than in short frames. Therefore, Equation 4.1 is not likely to yield accurate estimates in Mode 2, which is based on short frames. Alternatively, we used Equation 3.9 in 3.5.2 for Mode 2, which was believed to be more suitable for short frames.

Figure 28 shows the correlation sequence based on Equation 3.9. The same vowel token was used as in Figure 27, but the input signal was a 26-ms short frame rather than on an 80-ms segment. Note that this short frame is 286 points long at a sampling rate of 11025 samples/sec, and the length of the correlation sequence is 143 points.

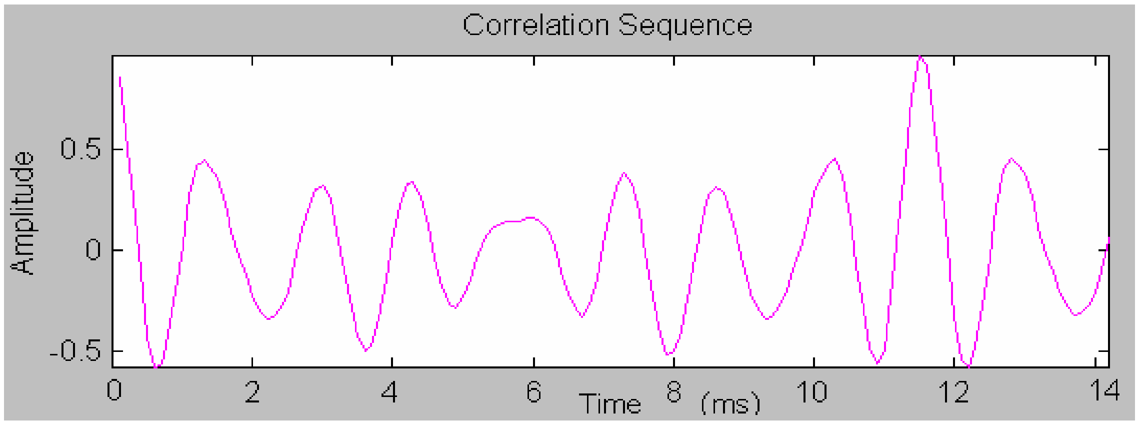


Figure 28. Autocorrelation sequence for a 26-ms signal frame of token "ah"

Once the autocorrelation sequence is defined, the next step is to identify the peak index in the sequence, and use this index as the pitch period. One or more periods can be identified in the original signal segment.

The way of finding the autocorrelation peak value in the sequence is straightforward and applies to both Mode 1 and Mode 2. A typical implementation is: starting from the first index, go through the whole sequence, and the first index with a value above a certain threshold is deemed as the pitch period. In the experiment, 0.8 was chosen as the default threshold, which was empirically determined to be appropriate for most cases.

Since we were dealing with steady-state vowel speech signal, which is quasi-periodic, some measures were taken to prevent obviously incorrect period estimations.

The first measure was setting a reasonable range for searching for the peak value index in the autocorrelation sequence. Observations of the autocorrelation sequences of a number of tokens showed that for some tokens, there were spurious peaks at very small indices. To eliminate these kinds of errors, we defined a revised frequency range between 70 Hz and 500 Hz, which corresponds to an index range between 22 samples and 157 samples at a sampling rate of 11025 samples/sec. By inspecting hundreds of tokens from all types of genders, we made sure that this was an appropriate index range that helped avoid incorrect period estimation.

There are also some cases in which all the peak values in the autocorrelation sequence are below the threshold, because of overall “poor” periodicity. Hence, it is possible that when we go through the entire sequence, we cannot find an index corresponding to a peak value above the threshold. To avoid this case, we visually inspected a number of tokens from male, female and child speakers, and selected three separate fundamental frequency ranges as shown below. More than 96% of the speakers inspected fell into these fundamental frequency ranges.

For male speakers: [80Hz, 155Hz];

For female speakers: [135Hz, 275Hz];

For child speakers: [185Hz, 370Hz]

For the training set, the speaker gender information was obtained from the token list files. This way, whenever we could not find an index that corresponded to an autocorrelation peak value above the threshold, we would detect the peak value in an index range corresponding to one of the fundamental frequency ranges defined above. The index that corresponded to this peak value was used as the pitch period.

For the test set, however, speaker gender information was not available. Hence, we could not take advantage of the frequency ranges above. In this case, we detected the autocorrelation peak in the general pitch frequency range for all genders, which was defined as [70Hz, 500Hz], and used the corresponding index as the pitch period.

For some 600 tokens that we visually inspected in the experiment, we were able to obtain reasonable pitch period identification for over 96% of all tokens, with the frequency range revision measures being taken as described above. Still, there were less than 4% of the tokens whose periodicity was too "bad" and could not be determined correctly in any way. In this case, we simply used the index, which corresponded to the maximum autocorrelation value in the general frequency range of [70Hz, 500Hz], as the pitch period. These "poorly periodic" tokens, however, were still included in the classification experiments in order that the database be identical for pitch-synchronous and non pitch-synchronous cases. This presumably degraded the classification results for the pitch-synchronous cases.

Once the pitch period was determined, we were able to extract one or more periods from the original signal segment, as described in the following section.

4.2.4 Single Or Multiple Period Extraction

In Mode 1, we extracted multiple periods for further spectral analysis, since we believed that one quasi-period might not provide complete information.

In our experiments, we used zero-crossing detection to help identify the edges of period. This method is illustrated in 3.5.4. To implement this method, we found the peak in the first pitch period of the signal segment, then detected zero-crossing points before and after the peak, respectively. The index of the zero-crossing point closest to the peak was used as the starting index for the extraction of multiple periods. We then added the length of one pitch period to the starting index. The zero-crossing in the nearest vicinity was the ending index for the first period.

In the meantime, the ending index of the first period was deemed as the starting index of the second period. By adding the length of one pitch period and finding zero-crossings in its vicinity, we were able to determine the ending index of the second period. This process was conducted repeatedly until we found the starting and ending indices for all the periods to be extracted. In our experiment, 4 periods were generally extracted. Figure 29 shows an example of zero-crossing based period-edge detection. Vertical lines in this figure mark the period edges, which are also zero-crossing points. As can be seen in Figure 29, 6 complete periods were identified.

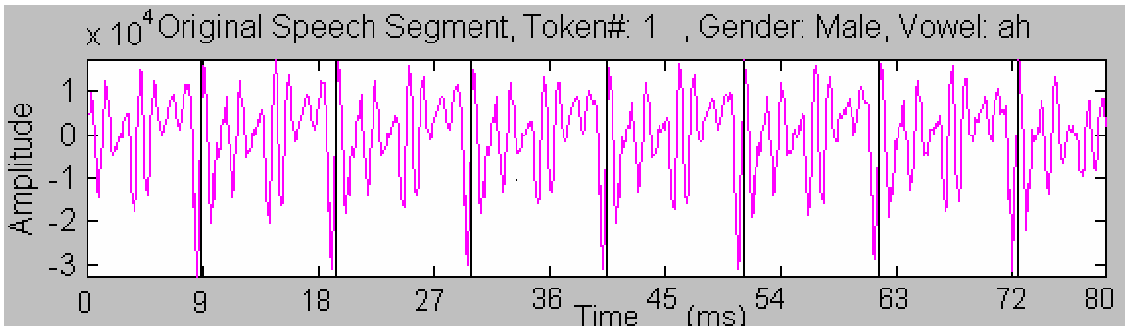


Figure 29. Multiple period extraction from a 80 ms segment of token "ah"

In Mode 2, we used exactly the same zero-crossing based method, as in Mode 1, to detect the period edges. An example is shown in Figure 30. The two vertical lines mark the edges of a single pitch period.

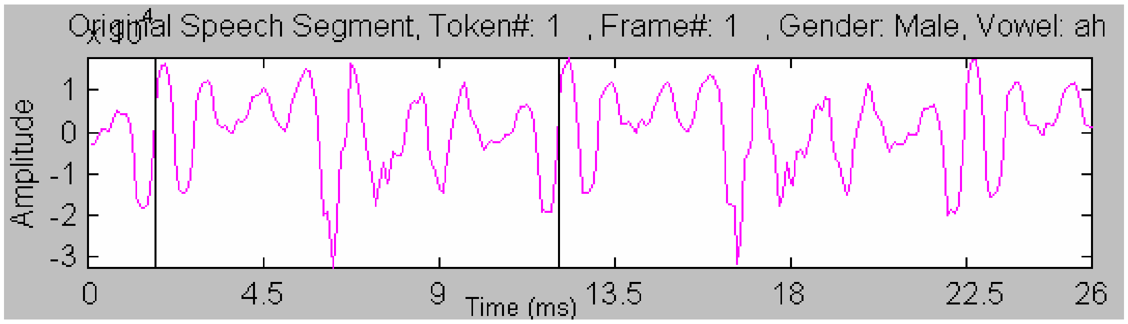


Figure 30. Single period extraction from a 26-ms segment of token "ah"

As compared to Mode 1, which extracted multiple periods from the same large segment, we extracted only one pitch period out of a short signal frame in Mode 2. Subsequent spectral processing was based on a single pitch period from each frame, instead of on multiple periods.

Since we used consecutive, overlapping frames, the pitch period should not vary much from frame to frame. In order to avoid accidental large variations in the pitch period estimation and ensure more accurate period extraction, we used an extra median smoothing step in Mode 2, which was not necessary in Mode 1.

Except for end points, we used a symmetrical median smoothing window, which was generally 3 points or 5 points long. Smoothing was performed on a pitch period value only when it was M% in deviation from the median value within the corresponding smoothing window.

In the Matlab implementation for Mode 2, we first estimated the pitch period for each of the overlapped frames. These pitch periods formed a "period vector." Median smoothing was then performed on this vector to acquire a smoothed "period vector". The following Matlab capture shows how median smoothing was done on the estimated pitch periods from 16 overlapped frames. The smoothing window length was 5, and the threshold for smoothing was set to 3%(M=3). Note that smoothing was actually done only to the underlined values, where large variations (greater than 3%) occurred.

Original_Period_Vec =

111 112 112 112 112 112 113 113 112 113 112 <u>104 117 101 108 115</u>

Smoothed_Period_Vec =

111 112 112 112 112 112 113 113 112 113 112 <u>112 108 108 108 108</u>

After smoothing, more accurate pitch period values for all frames were obtained. We then began to extract a single pitch period from each of the frames, and conduct spectral processing on this period. The final spectrum for a specific token was the average of the spectra of all the individual frames extracted from this token.

4.2.5 Resampling

As discussed in 3.5.5, time-domain resampling is required for the proper spectral processing in both Mode 1 and Mode 2. In the experiments, we used a resampling method on a period-by-period basis. This resampling method applied to both Mode 1 and Mode 2. The only difference is that there are multiple periods to be resampled one by one in Mode 1, while there is only one period to be resampled in Mode 2.

As a comparison, an extra case named the Control Mode was also implemented. For the Control Mode, there was actually no resampling, but just zero-padding. The algorithms of both resampling and zero-padding have been illustrated in 3.5.5. Resampling results for different methods are shown in Figure 31 and Figure 32.

In Figure 31, the input signal is a 4-period triangular signal, with the period being 9 milliseconds and a sampling rate of 11025Hz. The top graph shows Control Mode, in which the original signal was zero-padded to an appropriate FFT length of 512. The bottom plot illustrates the “resampled” signal, which was extended to an FFT length of 512 based on period-by-period resampling.

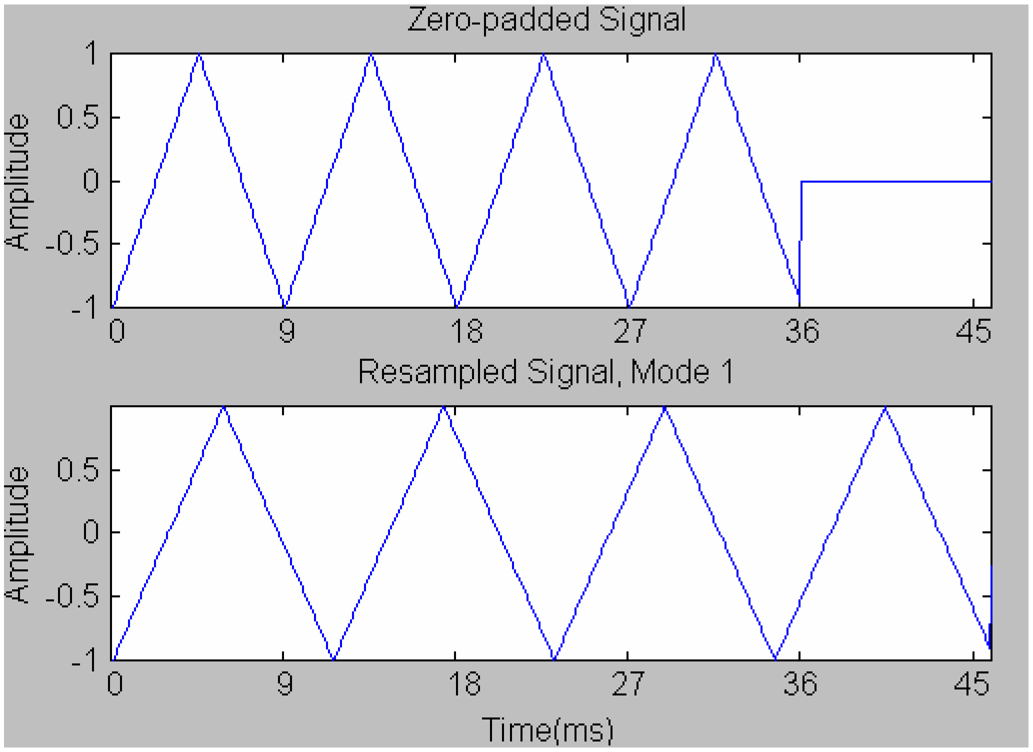


Figure 31. Time-domain resampling of triangular signal

Figure 32 further illustrates the spectral results of resampling. The two figures are the spectra of their corresponding time-domain waveforms in Figure 31.

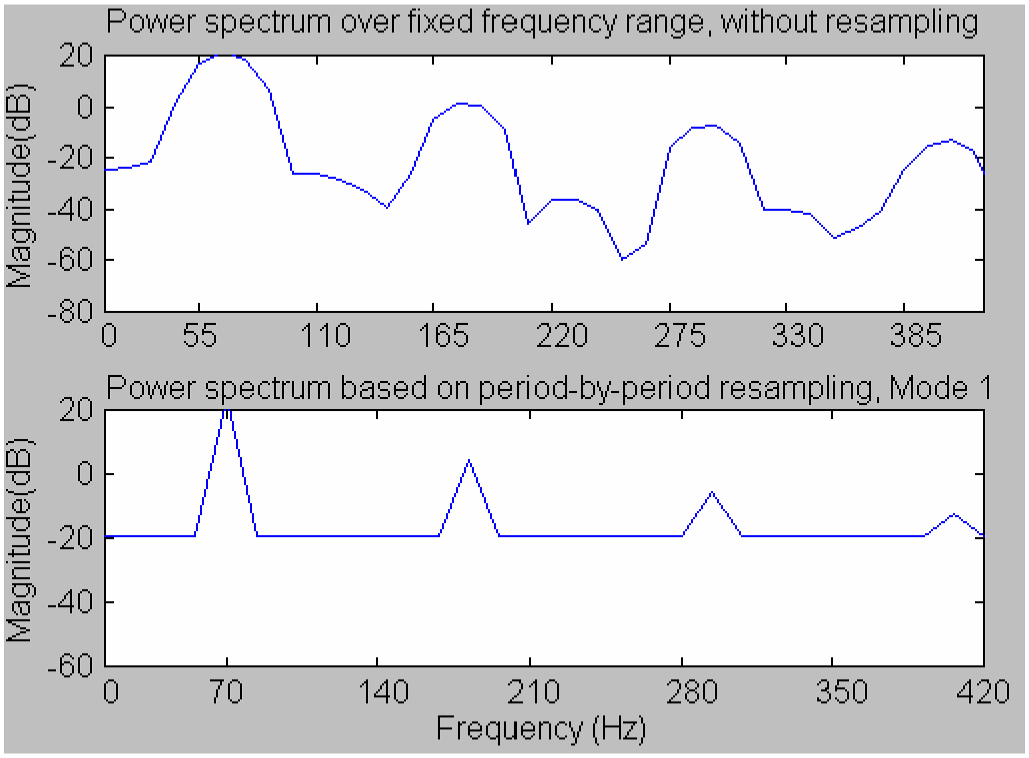


Figure 32. Spectra of Triangular Waveforms in Figure 31

As we can see from the examples in Figure 31 and Figure 32, when proper resampling was used, fundamental frequency and its harmonics were emphasized and spurious spectral peaks were reduced. This implies that resampling is desirable for spectral processing based on fundamental frequency and its harmonics.

4.2.6 Spectral Analysis

As stated above, we explored two methods of pitch-related processing: Mode 1 was based on multiple periods extracted from a large segment, while Mode 2 was based on single periods taken from short frames. Spectral analysis was performed the same way for both Mode 1 and Mode 2, except for one extra step implemented in Mode 1. That is, frequency downsampling was conducted to acquire the spectrum of a single period from the spectrum based on multiple periods. An explanation of frequency downsampling was given in 3.3.

It is obvious that no frequency downsampling is required in Mode 2, since it is always based on single periods.

As a comparison, a regular method that was not based on pitch periods was first implemented. Figure 33 shows the spectrum of a token processed using the regular method, in which multiple periods were zero-padded to a proper FFT length. In the regular method, no frequency scaling was implemented, and the whole spectral index range corresponded to a fixed frequency range of [100Hz, 5000Hz].

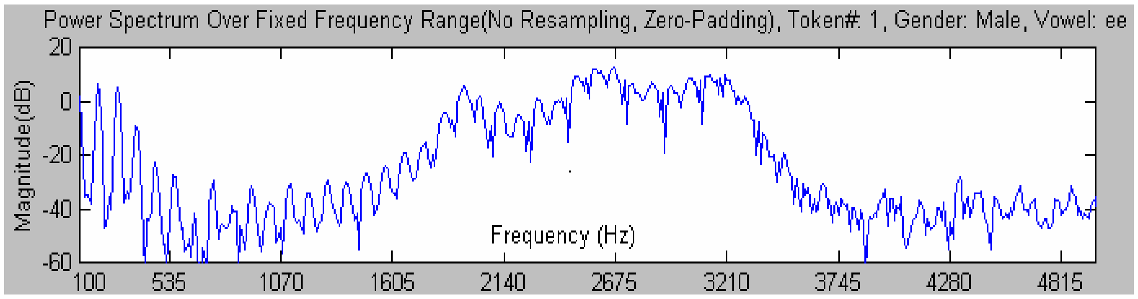


Figure 33. Power spectrum of token "ee" without time-domain resampling

A sample spectrum generated in Mode 1 is illustrated in Figure 34. Note that time-domain resampling was conducted period by period in this mode.

Non linear frequency scaling, as described in 3.5.3, was conducted in Mode 1. Note that this figure is based on the ($f_0^{1/3}$) scaled frequency range of [103Hz, 3603Hz], which includes the fundamental frequency 103 Hz and its 35 harmonics.

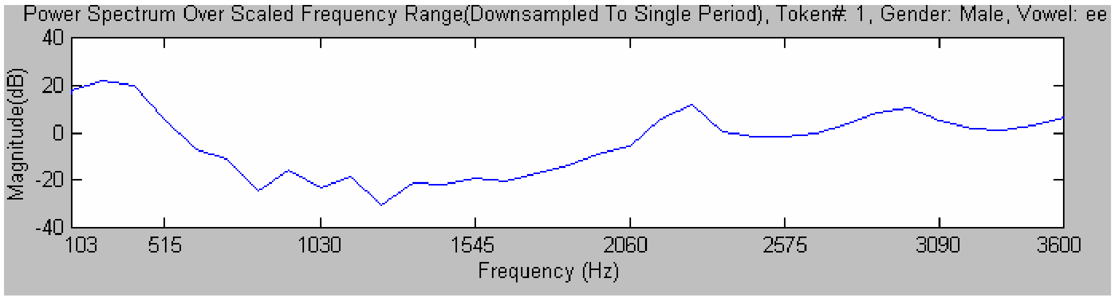


Figure 34. Power spectrum over scaled frequency range for Mode 1

Figure 34 actually shows the envelope of the fundamental frequency and its harmonics. Although Figure 34 cannot be directly compared to Figure 33 due to different frequency ranges, it is obvious that the overall frequency envelope remains consistent in both figures on the same frequency range, which implies the validity of Mode 1.

Since Mode 2 was based on single periods from the individual frames, we used the same resampling method as in Mode 1 to resample only one period extracted from each individual frame. Figure 35 illustrates the spectrum of a single period extracted from a 26-ms short frame for the same token "ee" as in Figure 34 and Figure 33. It can be observed that Figure 35 shows a quite similar spectrum compared to that in Figure 34, despite the fact that the processing in Mode 1 and Mode 2 was quite different. Obviously the processing in Mode 2 was consistent with the implementation in Mode 1.

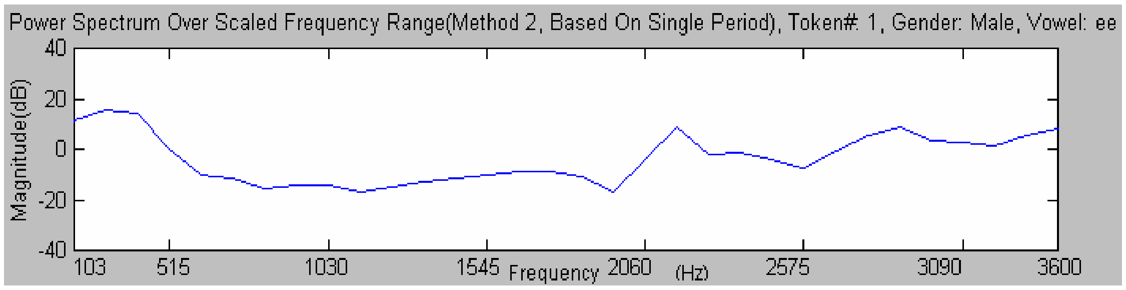


Figure 35. Power spectrum over scaled frequency range for Mode 2

4.2.7 Generation Of DCTC Features

Before the DCTC features were determined, a pre-defined cosine basis vector was generated according to the algorithm described in 2.3.1.

When we computed DCTC features, we explored two frequency ranges: one was fundamental frequency dependent, or f_0 dependent; the other was fundamental frequency independent, or f_0 independent. For the f_0 dependent range, we simply calculated pre-defined basis vectors over the whole frequency range, namely, [0, $Fs/2$]; for the f_0 independent range, we computed pre-defined basis vectors over a user-defined frequency range [F_{min}, F_{max}].

The pre-defined basis vectors and spectra were used to compute the final DCTC features according to the formula given in Equation 2.1 in 2.3.1.

In generating DCTC features, it was essential that frequency components be multiplied by their corresponding basis vector components on exactly the same frequency range and with the same frequency spacing. The rest of this section explains how the frequency range and its spacing were determined in different operation modes.

When we considered the f_0 independent frequency range, the pre-defined basis vectors were computed over [F_{min} , F_{max}], where F_{min} and F_{max} were the lower and upper frequency limits defined in the configuration file. As implemented in the simulation, a control mode and two new operation modes, Mode 1 and Mode 2, are described below:

1. Control mode: in this mode, the spectrum had the same frequency spacing as in pre-defined basis vectors. In this case, spectral components over the range of [F_{min} , F_{max}] were directly multiplied by basis vector components over the same range to generate the DCTC features.
2. Mode 1 and Mode 2: for these two modes, the frequency samples had a spacing of f_0. However, the pre-defined basis vector components were computed with a frequency spacing of

$$\frac{F_s}{N_{fft}} \tag{4.3}$$

where F_s was the sampling rate, and N_{fft} was the FFT length per period. Due to the difference in frequency spacing, interpolation was used to make the spectrum have a frequency spacing as shown in Expression 4.3. This extra processing step produced a one-to-one mapping between the frequency components and basis vector components over the same scaled frequency range. Thus, DCTC features could be properly computed.

For the f_0 dependent frequency range, the $(f_0^{1/3})$ scaled frequency range varies from token to token. Hence, the pre-defined basis vectors could not be computed over a fixed frequency range. Instead, the basis vectors were computed over the whole frequency range $[0, F_s /2]$, where F_s was the sampling rate. Again, the processing for different modes was also different.

1. Control mode: in this mode, fixed frequency range $[F_{min}, F_{max}]$ was first nonlinearly scaled to the frequency range of $[F_{min\_scaled}, F_{max\_scaled}]$, with the algorithm described in 3.5.3. Then, basis vectors on the same scaled frequency range were multiplied by the corresponding spectral components to generate DCTC features. Note that in this mode, the frequency spacing between frequency samples was the same as expressed in Equation 4.3.
2. Mode 1 and Mode 2: for these two modes, the scaled frequency range for spectral computations still had a spacing of f_0. Therefore, we interpolated the f_0-spaced, scaled spectrum to a new spectrum, which had the same frequency spacing as basis vectors. The DCTC features were computed by multiplying the frequency samples with their corresponding basis

vector components on the same range and with the same frequency spacing.

4.3 Experiments

To verify the effects of the Matlab simulation described in 4.2, the generated DCTC features (15 features were used in the experiments) were used with an artificial neural network classifier for classification. Experiment results are summarized and analyzed in this part.

4.3.1 Speech Database

In this research, we used a speech database recorded by the Old Dominion University Speech Lab. The speakers included male, female and child. For each of the speakers, up to 3 iterations were recorded for the same vowel token.

In the experiments, we chose 24 speakers from male, female and child group, respectively, to form the training set. For each vowel, 3 iterations of the same speaker were used. Similarly, we chose 8 speakers from different gender/age groups to generate the training set. There were also 3 iterations for each of the 10 vowel tokens. Note that the training set and the test set were entirely different from each other to ensure that these results correspond to a speaker independent case.

In summary, the training set contained 72 speakers and a total of 2160 vowel tokens; the test set contained 24 speakers and a total of 720 vowel tokens.

4.3.2 Neural Network Classifier

Neural networks are widely used in the field of speech recognition because they are powerful and flexible tools for pattern recognition. In this research, we used a Multiple-Layer Perceptron (MLP) neural network for classification. The MLP network contains a number of nodes called neurons. A group of neurons forms a layer. There may be one or more layers in a MLP network.

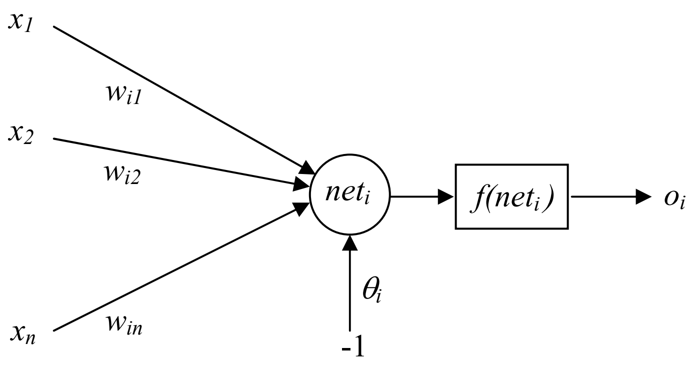


Figure 36. The structure of a neuron

A neuron is comprised of one or more inputs, associated weights, an internal offset and one output. In Figure 36, x_1 -x_n

$$o_i = f(net_i) = f\left[\left(\sum_{k=1}^{n} w_{ik} \cdot x_k\right) - \theta_i\right] \tag{4.4}$$

are input values to the *i*th neuron, w_{i1}-w_{in} represent the weights associated with their corresponding input values. θ_k is the internal offset, and *f(net$_i$)* stands for the activation function. o_i is the output of the neuron. The structure in Figure 36 can be expressed in the equation below.

The neural network classifier used in our experiments included one input layer, one output layer and one hidden layer. The overall structure of the neural network classifier is illustrated in Figure 37.

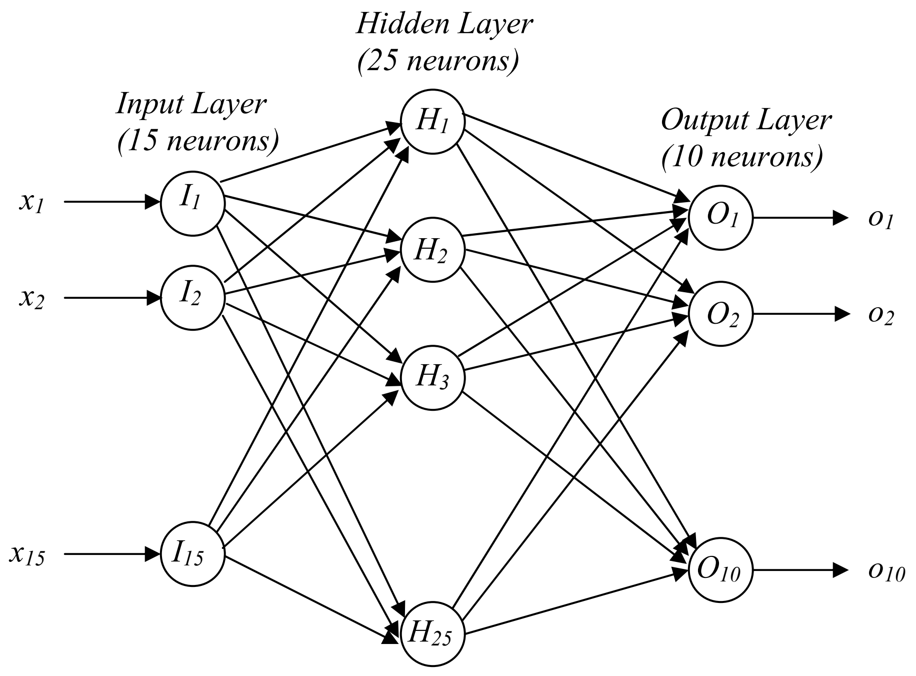


Figure 37. Structure of the neural network classifier in the experiments

The input layer consisted of 15 neurons corresponding to 15 DCTC features; and the output layer consisted of 10 neurons that output associated normalized weights for classifying the input features. The hidden layer with 25 hidden neurons was located between the input layer and the output layer.

4.3.3 Experiment Results

Table 3 shows the classification results with general speakers for different operation modes in the experiments.

General Speakers (24 x3 speakers for training, 8x3 speakers for test)					
Processing based on fixed frequency range			Processing based on f_0 dependent frequency range		
Control Mode (regular processing)	Training Set	Test Set	Control Mode (regular processing)	Training Set	Test Set
	91.2%	79.9%		90.4%	82.3%
Mode 1(period-by-period resampling on multiple periods)	Training Set	Test Set	Mode 1(period-by-period resampling on multiple periods)	Training Set	Test Set
	89.2%	80.6%		88.9%	81.9%
Mode 2(resampling on a single period)	Training Set	Test Set	Mode 2(resampling on a single period)	Training Set	Test Set
	89.1%	80.5%		89.8%	81.2%

Table 3. Classification results in Control Mode, Mode 1 and Mode 2, for general speakers

In Table 3, the Control Mode was used to simulate the regular processing method developed in the Old Dominion University Visual Speech Display system, which was not based on integer multiples of the pitch period. In the Control Mode, we used a 20-ms fixed frame length, with a 15 ms overlap. Frequency warping, with a warping factor of 0.45, was also used. The sampling rate was 11025 samples/sec, and the FFT length used was 256. In the meantime, the results with Mode 1 and Mode 2 are also shown in the same table. In Mode 1, we extracted 4 periods from an 80-ms segment, and used an FFT length of 1024 for these 4 periods. For Mode 2, we used sixteen 26-ms frames, with a 13-ms overlap between frames.

Male Speakers (24 speakers for training, 8 speakers for test)					
Processing based on fixed frequency range			Processing based on f_0 dependent frequency range		
Control Mode (regular processing)	Training Set	Test Set	Control Mode (regular processing)	Training Set	Test Set
	98.2%	88.7%		98.6%	90.0%
	Training Set	Test Set		Training Set	Test Set

Mode 1(period-by-period resampling on multiple periods)	Training Set	Test Set	Mode 1(period-by-period resampling on multiple periods)	Training Set	Test Set
	98.9%	90.0%		98.2%	87.5%
Mode 2(resampling on a single period)	Training Set	Test Set	Mode 2(resampling on a single period)	Training Set	Test Set
	99.2%	89.5%		98.6%	87.9%

Table 4. Classification results in Control Mode, Mode 1 and Mode 2, for male speakers

Table 4, Table 5 and Table 6 contain the classification results of three additional tests for male speakers, female speakers and child speakers, respectively. Note that the tokens used in these tests come from the same database as in the test with general speakers.

As can be seen in Table 4, the use of f_0 dependent frequency range did not lead to better classification performance in Mode 1 and Mode 2 for the case of male speakers. In contrast, for female and child speakers as in Table 5 and Table 6, f_0 dependent frequency range did result in higher performance. We found that the tokens of male speakers generally have better periodicity and more distinctive harmonics in spectrum than those of female and child speakers. This implies that the (f_0 $^{1/3}$) frequency scaling may not be necessary for processing tokens that have good periodicity.

Female Speakers (24 speakers for training, 8 speakers for test)					
Processing based on fixed frequency range			Processing based on f_0 dependent frequency range		
Control Mode (regular processing)	Training Set	Test Set	Control Mode (regular processing)	Training Set	Test Set
	97.4%	88.1%		97.2%	89.8%
Mode 1(period-by-period resampling on multiple periods)	Training Set	Test Set	Mode 1(period-by-period resampling on multiple periods)	Training Set	Test Set
	98.0%	81.8%		96.2%	84.8%

Mode 2(resampling on a single period)	Training Set	Test Set	Mode 2(resampling on a single period)	Training Set	Test Set
	96.4%	85.6%		97.3%	89.0%

Table 5. Classification results in Control Mode, Mode 1 and Mode 2, for female speakers

The test results in Table 6 shows inferior performance with child speakers as compared to male and female speakers. It was found that, by visual inspection, the tokens of child speakers in the database generally have worse periodicity than those of male and female speakers. This implies that f_0 dependent processing may not result in good performance when the periodicity of speech signal is not good.

If we study the test results for the Control Mode in Table 4 through Table 6, we may find that the use of f_0 dependent frequency range helps increase the classification performance for the regular processing method with all types of speakers. This shows again that f_0 could be a useful clue for speech recognition.

It can be seen that no significant performance improvement was obtained when using the multiple-period based method instead of the regular method. There could be three possible reasons.

Child Speakers (24 speakers for training, 8 speakers for test)					
Processing based on fixed frequency range			Processing based on f_0 dependent frequency range		
Control Mode (regular processing)	Training Set	Test Set	Control Mode (regular processing)	Training Set	Test Set
	95.6%	77.8%		96.2%	79.1%
Mode 1(period-by-period resampling on multiple periods)	Training Set	Test Set	Mode 1(period-by-period resampling on multiple periods)	Training Set	Test Set
	87.2%	72.6%		86.7%	75.7%
	Training Set	Test Set		Training Set	Test Set

	Training Set	Test Set		Training Set	Test Set
Mode 2(resampling on a single period)	**89.5%**	**69.7%**	**Mode 2(resampling on a single period)**	**87.5%**	**71.1%**

Table 6. Classification results in Control Mode, Mode 1 and Mode 2, for child speakers

The first reason was related to the random characteristics of speech signal. The speech signal by nature is random and quasi-periodic. The quasi-periodicity of a speech token contributed to the inaccuracy of recognition in the experiments. In this research, we visually inspected hundreds of tokens, and found that some 4% of the total tokens did not have an acceptable periodicity suitable for period-based processing. These tokens, however, were included in the processing and classification to maintain generality. Errors could be introduced due to those poorly periodic tokens.

The second reason might due to the pitch period estimation. In our experiments, we utilized a method of autocorrelation and zero-crossing detection. This method was proved to be effective through visual inspection. However, we were concerned about the fact that the random characteristics of speech signal could lead to the incorrect detection of period edges. This could also result in overall degradation in classification performance.

The third reason might come from the interpolation process before DCTC features were computed. Since the frequency spacing of the basis vector computations might not be the same as that for spectrum computations, linear interpolation was required to obtain unique frequency spacing. The interpolation process could also bring in some errors in spectrum, thus making the final DCTC features inaccurate.

As can be seen in Table 3, the processing based on single periods extracted from short frames, which was implemented in Mode 2, contributed to roughly the same recognition results as compared to Mode 1, where spectral processing was based on multiple periods. This shows that the processing based on short frames and spectral averaging did not excel in performance, compared to the method based on multiple periods extracted from a large segment.

One interesting point we found in all the operation modes is that, the f_0 dependent frequency range was helpful in getting better recognition results regardless of what method was used. If we compare the test results for the f_0 dependent range and the fixed frequency range, it is obvious that there was an increase in recognition rate when the f_0 dependent frequency range was used to compute DCTC features. This implied that f_0, the fundamental frequency, could be an important clue in recognition no matter what specific method was used.

Another interesting observation was the DCTC feature stability over time. For Mode 1, it is hard to illustrate this point, since the multiple periods were extracted from the center of the token only once. However, because Mode 2 extracted a series of short, overlapped frames from the token, we were able to illustrate the variations of the DCTC feature values over time.

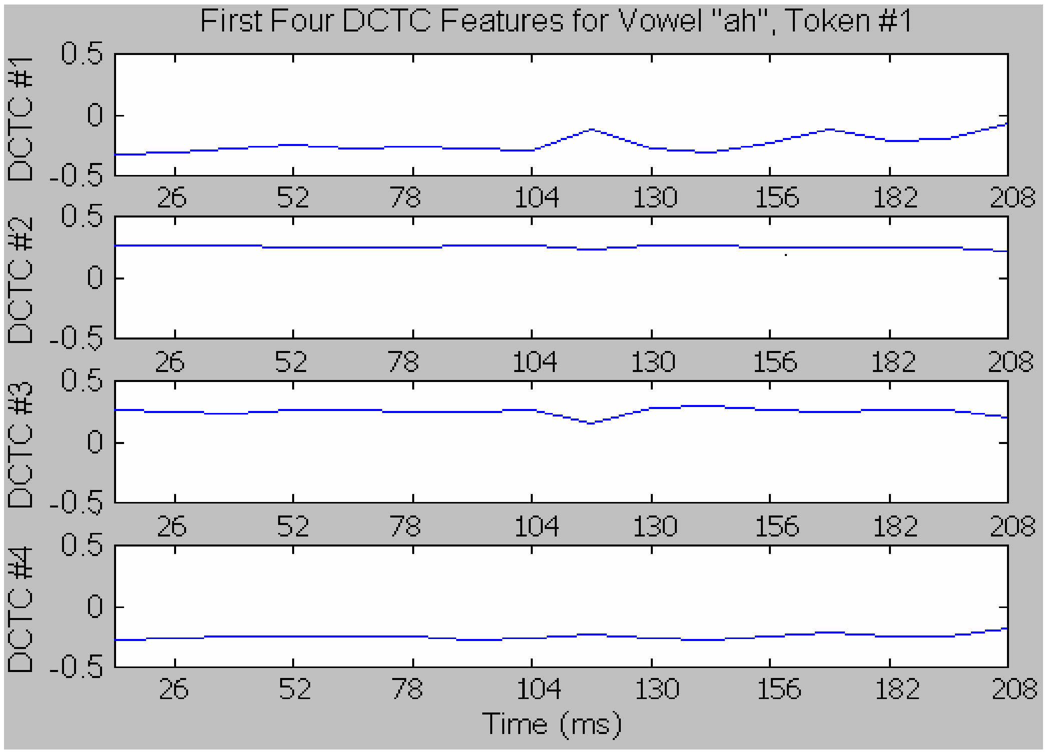


Figure 38. ***First four DCTC feature values over time for token "ah"***

As shown in Figure 38, sixteen overlapped frames were extracted from the center of the token, with each frame being 26 ms and the overlap between frames being 13 ms. Figure 10 in Chapter II illustrated the first four DCTC features of the same token generated in the Control Mode, in which 16 non-overlapping, 15-ms frames were used.

If we compare Figure 38 with Figure 10, it can be seen that the short-frame based pitch-synchronous method, implemented in Mode 2, reduces the variability of the DCTC features, although the final classification results did not show significant improvements compared to the regular method implemented in the Control Mode.

4.4 Conclusions

In this chapter, we explored the Matlab software implementation of the system described in Chapter III. The implementation of the pitch period estimation, which is the fundamental of this system, was visually examined to be robust. We plotted the signal waveforms of 600 speech tokens, in which estimated periods were marked, and found that this period estimation method worked well for over 96% of all the tokens. For less than 4% of the tokens, whose periodicity could not even be identified by visual inspection, the pitch period estimation method did not work.

The final classification results showed that the pitch-synchronous processing did not outperform the regular window-based method, although the pitch-synchronous method was shown to help reduce spectral smearing. This implies that window effects could be a minor factor in speech recognition. However, we found that the spectral analysis on pitch-dependent, ($f_0^{1/3}$) scaled frequency ranges leads to better classification results, as compared to pitch-independent frequency ranges. In addition, pitch-synchronous method also helped reduce the DCTC variability.

CHAPTER V

ACHIEVEMENTS AND FUTURE WORK

5.1 Achievements

The whole work in this research was implemented in two major steps. The first step was focused on the exploration of DCTC feature variability and the algorithms to reduce this variability. In the second step, we implemented a pitch-synchronous system aimed at reducing DCTC feature variability from a new perspective.

We found that the DCTC feature variability, which showed up in both real-time Visual Speech Display and non real-time analysis, was due to the noise and inherently unstable characteristics of the DCTC features. Noise may come from the background and PC components, including the sound card and other components.

We also explored various methods to reduce DCTC variability. Undoubtedly, using a high quality sound card and PC will help reduce this variability. Our experimental results also indicated it always helped reduce DCTC variability when appropriate length overlapping frames, proper frequency warping and time smoothing were used.

In our second attempt to reduce DCTC variability, we implemented a new pitch period based algorithm and simulated it in Matlab. This algorithm consisted of extracting a long segment or a short frame from the center of a vowel, estimating the pitch period based on these segments or short frames using the autocorrelation, and generating DCTC features utilizing pitch-dependent or pitch-independent spectra. The Matlab simulation not only fully implemented this algorithm, but also included extra steps to handle exceptions when some poorly periodic tokens were processed.

Visual inspections on hundreds of tokens showed that the pitch period estimation method was successful for over 96% of all tokens. We also showed that time-domain resampling helped increase frequency-domain resolution, and that spectral analysis based on pitch-dependent frequency range produced better classification results than regular pitch-independent frequency range.

5.2 Future Work

The utilization of pitch-synchronous analysis in vowel token recognition is a new and promising approach. However, this method may be limited by the inherently random characteristics of speech signals. This probably is the reason why this method does not show improvements in classification results as compared to conventional non pitch-synchronous methods.

For the pitch-synchronous method, the key point is the estimation of pitch period. Although the estimated pitch periods have been visually inspected to be accurate, we are not able to check temporal details at the edges of period. It is likely that for some tokens we could not correctly determine the period edges due to the randomness of temporal details in the token.

Future work may focus on a better algorithm in determining the pitch period from a segment, and in detecting the period edges when multiple periods are required to be extracted from the token for analysis.

Further experiments can also be done using only those tokens with "good" periodicity, i.e. the tokens whose autocorrelation peak value exceeds the threshold 0.8, as discussed in Chapter IV. The results of these experiments will provide more information about the extent of performance degradation in the presence of "poorly periodic" tokens.

REFERENCES

Betancourt, O., and Antrobus, J. (1998). "Using natural harmonics as acoustic features in speech recognition: a vowel classification example", J. Acoust. Soc. Am. Manuscript, JASA97/015

Bladon, R.A.V., (1982). "Arguments against formants in the auditory representation of speech," from *The Representation of Speech in the Peripheral Auditory System*, edited by R. Carlson and B. Granstrom (Elsevier, Amsterdam), pp. 95-102

Haddad, R.A., and Parsons, T.W. (1991). "Digital signal processing: theory, applications, and hardware," Computer Science Press, 1991, pp. 367-370

Hu, Z., and Barnard, E. (1997). "Smoothness analysis for trajectory features," ICASSP-97, Volume 2, 979-982

Kelkar, S.U., (1992). "Formant estimation from DCTC's using a feedforward neural network," Master's Thesis, Old Dominion University, May 1992

Klein, W., Plomp, R., and Plos, L. (1970). "Vowel spectra, vowel spaces, and vowel identification," J. Acoust. Soc. Am. 48, 999-1009

Lippmann, R., (1987), "An introduction to computing with neural nets," IEEE ASSP Magazine, April, 4-22

Miller, J.D. (1989). "Auditory-perceptual representation of the vowel," J. Acoust. Soc. Am. 85, 2114-2134

Nossair, Z.B., (1989). "Dynamic spectral shape features as acoustic correlates for stop consonants," Research Report, Old Dominion University Research Foundation, December 1989

Nossair, Z., Silsbee, P., and Zahorian, S. (1995). "Signal modeling enhancements for automatic speech recognition," ICASSP-95, pp. 824-827

Peterson, G.E., and Barney, H.L. (1952). "Control methods used in a study of the vowels," J. Acoust. Soc. Am.24, 175- 184

Plomp, R. Plos, L.C.W., and van de Geer, J.P. (1967). "Dimensional analysis of vowel spectra," J. Acoust. Soc. Am. 41, 707-712

Pols, L.C.W., van der Kamp, L.J.Th., and Plomp, R. (1969). "Perceptual and physical space of vowel sounds," J.Acoust. Soc. Am. 46, 458-467

Proakis, J.G, and Manolakis, D.G. (1996). "Digital Signal Processing: Principles, Algorithms, and Applications," 3rd Edition, Prentice Hall, pp. 327-330

Talkin, D., (1995). "A robust algorithm for pitch tracking (RAPT)," from *Speech Coding and Synthesis*, edited by W.B. Kleijn and K.K. Paliwal (Elsevier Science B.V.), pp. 502-507

Zahorian, S.A., and Amir Jalali Jagharghi(1993). "Spectral-shape features versus formants as acoustic correlates for vowels," J. Acoust. Soc. Am. 94(4), 1966-1982

Zahorian, S.A., and Gordy, P.E.(1983). "Finite impluse response(FIR) filters for speech analysis and synthesis," ICASSP-83, 808-811

General Speakers (24 x3 speakers for training, 8x3 speakers for test)					
Processing based on fixed frequency range			Processing based on f_0 dependent frequency range		
Control Mode (regular processing)	Training Set	Test Set	Control Mode (regular processing)	Training Set	Test Set
	91.2%	79.9%		90.4%	82.3%
Mode 1(period-by-period resampling on multiple periods)	Training Set	Test Set	Mode 1(period-by-period resampling on multiple periods)	Training Set	Test Set
	89.2%	80.6%		88.9%	81.9%
Mode 2(resampling on a single period)	Training Set	Test Set	Mode 2(resampling on a single period)	Training Set	Test Set
	89.1%	80.5%		89.8%	81.2%

Table 3. Classification results with Control Mode, Mode 1 and Mode 2, for general speakers

Male Speakers (24 speakers for training, 8 speakers for test)					
Processing based on fixed frequency range			Processing based on f_0 dependent frequency range		
Control Mode (regular processing)	Training Set	Test Set	Control Mode (regular processing)	Training Set	Test Set
	98.2%	88.7%		98.6%	90.0%
Mode 1(period-by-period resampling on multiple periods)	Training Set	Test Set	Mode 1(period-by-period resampling on multiple periods)	Training Set	Test Set
	98.9%	90.0%		98.2%	87.5%
Mode 2(resampling on a single period)	Training Set	Test Set	Mode 2(resampling on a single period)	Training Set	Test Set
	99.2%	89.5%		98.6%	87.9%

Table 4. Classification results with Control Mode, Mode 1 and Mode 2, for male speakers

Female Speakers (24 speakers for training, 8 speakers for test)					
Processing based on fixed frequency range			Processing based on f_0 dependent frequency range		
Control Mode (regular processing)	Training Set	Test Set	Control Mode (regular processing)	Training Set	Test Set
	97.4%	88.1%		97.2%	89.8%
Mode 1(period-by-period resampling on multiple periods)	Training Set	Test Set	Mode 1(period-by-period resampling on multiple periods)	Training Set	Test Set
	98.0%	81.8%		96.2%	84.8%
Mode 2(resampling on a single period)	Training Set	Test Set	Mode 2(resampling on a single period)	Training Set	Test Set
	96.4%	85.6%		97.3%	89.0%

Table 5. Classification results with Control Mode, Mode 1 and Mode 2, for female speakers

Child Speakers (24 speakers for training, 8 speakers for test)					
Processing based on fixed frequency range			**Processing based on f_0 dependent frequency range**		
Control Mode (regular processing)	**Training Set**	**Test Set**	**Control Mode (regular processing)**	**Training Set**	**Test Set**
	95.6%	**77.8%**		**96.2%**	**79.1%**
Mode 1(period-by-period resampling on multiple periods)	**Training Set**	**Test Set**	**Mode 1(period-by-period resampling on multiple periods)**	**Training Set**	**Test Set**
	87.2%	**72.6%**		**86.7%**	**75.7%**
Mode 2(resampling on a single period)	**Training Set**	**Test Set**	**Mode 2(resampling on a single period)**	**Training Set**	**Test Set**
	89.5%	**69.7%**		**87.5%**	**71.1%**

Table 6. Classification results with Control Mode, Mode 1 and Mode 2, for child speakers

Table 4, Table 5 and Table 6 contain the classification results of three additional tests for male speakers, female speakers and child speakers, respectively. Note that the tokens used in these tests come from the same database as in the test with general speakers.

As can be seen in Table 4, the use of f_0 dependent frequency range did not lead to better classification performance in Mode 1 and Mode 2 for the case of male speakers. In contrast, for female and child speakers as in Table 5 and Table 6, f_0 dependent frequency range did result in higher performance. We found that the tokens of male speakers generally have better periodicity and more distinctive harmonics in spectrum than those of female and child speakers. This implies that the ($f_0^{1/3}$) frequency scaling may not be necessary for processing tokens that have good periodicity.

The test results in Table 6 shows inferior performance with child speakers as compared to male and female speakers. It was found that, by visual inspection, the tokens of child speakers in the database generally have worse periodicity than male and female speakers. This implies that f_0 dependent processing may not result in good performance when the periodicity of speech signal is not good.

If we study the test results for the Control Mode in Table 4 through Table 6, we may find that the use of f_0 dependent frequency range helps increase the classification performance for the regular processing method with all types of speakers. This shows again that f_0 could be a useful clue for speech recognition.