Time-Scale Modification of Speech
via Short-Time Fourier Analysis

by

Kirk Lauritz Johnson

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degrees of

Bachelor of Science

and

Master of Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 1989

© Massachusetts Institute of Technology 1989

Signature of Author... 

Department of Electrical Engineering and Computer Science
May 16, 1989

Certified by...... 

Jae Lim
Associate Professor
Thesis Supervisor

Accepted by...... 

Arthur C. Smith
Chairman, Departmental Committee on Graduate Students

JUL 1 11 1989
Time-Scale Modification of Speech
via Short-Time Fourier Analysis

by

Kirk Lauritz Johnson

Submitted to the Department of Electrical Engineering and Computer Science
on May 22, 1989, in partial fulfillment of the
requirements for the degrees of
Bachelor of Science
and
Master of Science

Abstract

A class of algorithms for Time-Scale Modification (TSM) of speech based on short-time Fourier analysis is described. Performance of two specific algorithms is investigated for various frequency resolutions.

In [13], Portnoff describes an algorithm for TSM based on short-time Fourier analysis. Holtzman provides a reformulation of Portnoff's system in [2]. These algorithms are examined in some detail; details of the implementation used in this thesis are provided. It is found that this implementation does not provide the same level of processed speech quality provided by the implementations of Portnoff and Holtzman; this would seem to indicate some fallacy in either the derivation or implementation of the Portnoff/Holtzman algorithm as used in this thesis. A revised algorithm which provides somewhat higher speech quality is developed. Results of a subjective listening test designed to investigate perceived differences between speech processed using various frequency resolutions are presented.

Thesis Supervisor: Jae Lim
Title: Associate Professor
Acknowledgments

This thesis is dedicated to my mother. Without her undying support, none of this would be possible.

I would also like to thank Ben Wilbanks, my supervisor at ROLM. Ben proved to be far more than an extremely capable manager; he also became a good friend.

Finally, thanks to John Buck for putting up with my insanity over the years.
Contents

1 Introduction ......................................................... 8
   1.1 Background .................................................. 8
   1.2 Scope ....................................................... 9
   1.3 Overview .................................................... 10
   1.4 Miscellany ................................................... 11

2 Short-Time Fourier Analysis ....................................... 12
   2.1 The STFT .................................................... 12
   2.2 The DSTFT .................................................. 13
   2.3 Interpreting the DSTFT .................................... 15
   2.4 Decimating the DSTFT ...................................... 16
   2.5 Efficient Computation ...................................... 18
      2.5.1 Analysis ............................................... 18
      2.5.2 Synthesis .............................................. 19
      2.5.3 Implementation ...................................... 19

3 Portnoff’s Algorithm ............................................. 21
3.1 A Model for Speech ........................................... 21
3.2 Model of Time-Scale Modified Speech ......................... 26
3.3 Modification via the DSTFT .................................. 28
  3.3.1 Analysis .................................................. 28
  3.3.2 Modification ............................................. 32
  3.3.3 Synthesis .................................................. 37
3.4 Implementation Details ....................................... 37
  3.4.1 Analysis Filter .......................................... 37
  3.4.2 Synthesis Filter ......................................... 38
  3.4.3 Decimation and Interpolation Factors .................... 39
  3.4.4 Frequency Resolution ................................... 40
3.5 Results ....................................................... 40
  3.5.1 Periodic Signals ........................................ 41
  3.5.2 Quasi-Periodic Signals .................................. 42
  3.5.3 Integer Rates of Slowdown ................................ 44
3.6 Explanation .................................................. 44

4 Revised Algorithm ............................................. 48
  4.1 How it Works ............................................... 48
  4.2 Output Quality .............................................. 50
  4.3 Varying the Frequency Resolution ........................... 51

5 Conclusions .................................................... 53

A Simulation Codes .................................................. 55
List of Figures

2-1 Filter bank interpretation of DSTFT analysis ........................................ 16
2-2 Inverse filter bank interpretation of DSTFT synthesis ............................ 17
3-1 Speech production model (after Portnoff) ............................................. 22
3-2 Quasi-periodic impulse train ................................................................. 23
3-3 Original periodic test signal ................................................................. 41
3-4 Periodic test signal processed by Portnoff/Holtzman algorithm, $\beta = \frac{29}{37}$ ... 42
3-5 Original quasi-periodic test signal .......................................................... 43
3-6 Quasi-periodic test signal processed by Portnoff/Holtzman algorithm, $\beta = \frac{29}{37}$ ... 43
3-7 Quasi-periodic test signal processed by Portnoff/Holtzman algorithm, $\beta = \frac{1}{2}$ ... 45
4-1 Quasi-periodic test signal processed by revised algorithm, $\beta = \frac{29}{37}$ .......... 51
Chapter 1

Introduction

1.1 Background

Time-Scale Modification (TSM) algorithms provide a means for changing the apparent rate of recorded speech passages without changing the pitch or distorting the frequency spectrum significantly. A good TSM algorithm is one that produces "natural-sounding" processed speech over the range of playback rates that is of interest to the end user (possibly from two to three times slow down to two to three times speed up). Intelligibility, tonal quality, and speaker recognizability should be preserved, and processing artifacts (pops, clicks, burbles, reverberation, etc.) should be kept to a minimum.

A number of TSM algorithms have been forwarded. The sampling method developed by Garvey and Fairbanks in the 1950's is possibly the simplest [7]. This algorithm achieves time-scale modification by periodically discarding or repeating voice segments. The length of these segments is fixed and determined a priori, chosen to be longer than the expected pitch period duration (in order to preserve tonal quality), but shorter than a phoneme (in order to preserve intelligibility). Typical segment lengths lie in the 15 to 50 millisecond range. Although the sampling method produces intelligible output over a wide range of playback rates, it introduces unacceptable degradation due to discontinuities at inter-segment boundaries.

Pitch-synchronous algorithms are a variant of the sampling method. Instead of using fixed-size
segments, these algorithms attempt to size segments dynamically (through the use of pitch period estimation) in an effort to reduce the magnitude of inter-segment discontinuities. Although this sophistication leads to improved performance, pitch period estimate errors often cause noticeable degradation.

One mechanism for improving pitch-synchronous algorithm performance is some type of frame-to-frame "smoothing" to further reduce inter-segment discontinuities. Although Malah justifies his Time Domain Harmonic Scaling (TDHS) algorithm [8] with some reasobably complex mathematics, the end result is effectively a pitch-synchronous algorithm which utilizes a clever form of smoothing. The smoothing mechanism used in TDHS improves output speech quality significantly, but objectionable artifacts are still present in processed speech.

Another broad class of algorithms is characterized by the use of analysis/synthesis techniques. These techniques provide a means of transforming a speech signal into a set of time-varying speech parameters; time-scale modification can be accomplished by time-scaling these parameters. The algorithms developed by Portnoff [13] (based on short-time Fourier analysis), Quatieri [17] (based on a summed-sinusoid modeling of speech), and Griffin and Lim [5] (based on magnitude-only phase-reconstruction of modified short-time Fourier analysis data) fall into this category. These algorithms tend to provide much higher quality output than algorithms based in the time domain, but often at the cost of drastically increased computational complexity.

1.2 Scope

Portnoff describes an algorithm for TSM based on short-time Fourier analysis/synthesis in [13] (see also [14, 15, 16]). In [2], Holtzman reformulates Portnoff's algorithm to allow non-uniform time-scale modification and to significantly reduce storage requirements.

The original goal of this thesis project was to investigate a system for TSM proposed by Galand in [5]. Galand's algorithm is similar to that of Portnoff and Holtzman; the primary difference lies in the frequency resolution afforded by the means of analysis/synthesis. For this purpose, Galand suggests use of Quadrature Mirror Filter (QMF) banks which provide sub-bands spaced at 100 Hz intervals; Portnoff and Holtzman utilize the DSTFT to obtain sub-bands spaced at
approximately 20 Hz intervals (by using 512-point DFTs and assuming a 10 kHz sampling rate). This author's implementation of Galand's algorithm provided low-quality, reverberant processed speech. It was hoped that implementing the system described by Holtzman in [2] and investigating its performance for various frequency resolutions might provide insight into the observed poor performance of Galand's algorithm.

Unfortunately, in the course of trying to duplicate the work described by Holtzman, there were numerous problems. Although the algorithm he describes is reasonably straightforward to implement and seemingly without ambiguity, the quality of his results, as demonstrated on an audio tape recorded near the end of his research, remained elusive and somewhat out of reach, although fairly high output quality was achieved in some individual cases.

Because of this problem, the focus of this thesis project changed towards trying to understand why the implementation of Holtzman's system used in this thesis provided a certain level of output speech quality and what kept it from providing higher quality. This information was then applied to developing a revised formulation which provided similar or somewhat higher speech quality in most cases.

1.3 Overview

Chapter two provides a brief discussion of short-time Fourier analysis (and, in particular, the Discrete Short-Time Fourier Transform or DSTFT).

Chapter three provides a reasonably detailed description of the Portnoff/Holtzman algorithm and its underlying mathematical framework. A parametric model of speech is developed, and, using this model, it is shown that time-scale modification can be accomplished through appropriate manipulation of a signal's STFT. This is followed by a description of the specific implementation used in this study and a brief discussion of the speech quality provided by this implementation. It is found that this implementation does not provide the same level of processed speech quality provided by the implementations of Portnoff and Holtzman; this would seem to indicate some fallacy in either the derivation or implementation of the Portnoff/Holtzman algorithm as used in this thesis. Finally, two possible problems with the algo-
rithm are described.

Chapter four presents a revised algorithm which attempts to correct one of the problems of the Portnoff/Holtzman system identified at the end of chapter three. A short discussion of the speech quality provided by this algorithm follows, including results of a subjective listening test designed to investigate the effect of changing the frequency resolution at which speech segments are processed (with the revised algorithm).

Finally, chapter five concludes with a summary of the major results of the thesis and a brief survey of some TSM algorithms which have followed the Portnoff/Holtzman work.

Appendix A contains listings of the simulation codes used in this thesis.

1.4 Miscellany

The bulk of this work was done while the author was on a co-op assignment at IBM Corporation, ROLM Systems Division (now ROLM Systems, Inc.) in Santa Clara, CA.

Analog-to-digital and digital-to-analog conversions of the audio segments used in this study were done with proprietary hardware which utilized a sampling frequency of 8 kHz.
Chapter 2

Short-Time Fourier Analysis

The TSM algorithms discussed in this thesis utilize the Discrete Short-Time Fourier Transform (DSTFT) and its inverse, the discrete-frequency cousins of the classical short-time Fourier analysis/synthesis relations. This chapter provides an overview of the pertinent details for the reader unfamiliar with this framework.

2.1 The STFT

The classical (continuous frequency) Short-Time Fourier Transform (STFT) of a signal $x[n]$ is given by

$$X[n, \omega] = \sum_{m=-\infty}^{+\infty} h[n - m]x[m]e^{-j\omega m}$$

Equation (2.2) gives the corresponding inverse STFT equation.

$$x[n] = \frac{1}{2\pi} \int_{-\pi}^{+\pi} \sum_{m=-\infty}^{+\infty} f[n - m]X[m, \omega]e^{j\omega n} d\omega$$

In (2.1) and (2.2), $h[n]$ and $f[n]$ are called the analysis filter and synthesis filter, respectively. Note that, in the more general case, these could be time-varying filters (e.g. $h[n, m]$ and
\( f[n, m] \). However, since the algorithms discussed in this thesis use only time-invariant analysis and synthesis filters, no further discussion of this issue is provided here.

Portnoff provides further detail in [13] and [14].

### 2.2 The DSTFT

Due to the use of a continuous frequency variable \((\omega)\), the STFT turns out to be quite unwieldy for actual use in numeric simulations. Thus we derive the Discrete Short-Time Fourier Transform (DSTFT) by sampling \(\omega\) at some set of frequencies (in much the same way that the Discrete Fourier Transform (DFT) can be "derived" from the Discrete-Time Fourier Transform (DTFT) [11]).

Observe that \(X[n, \omega]\) is periodic in \(\omega\) with period \(2\pi\). Let \(M\) be the desired number of frequency samples. To obtain the DSTFT from the STFT, then, \(X[n, \omega]\) should be sampled in \(\omega\) with a sampling interval of \(2\pi/M\). Thus \(X[n, k]\), the DSTFT of \(x[n]\), can be written as

\[
X[n, k] = \sum_{m=-\infty}^{+\infty} h[n - m]x[m]e^{-j(2\pi/M)km} \tag{2.3}
\]

where the frequency index, \(k\) is expected to be in the interval \([0, M - 1]\) and a particular index \(k\) corresponds to the frequency \(\omega = (2\pi/M)k\).

It is convenient to rewrite this equation in terms of \(W_M\) defined as

\[
W_M = e^{-j(2\pi/M)} \tag{2.4}
\]

Thus, (2.3) can be rewritten as

\[
X[n, k] = \sum_{m=-\infty}^{+\infty} h[n - m]x[m]W_M^{km} \tag{2.5}
\]

Similarly, the DSTFT synthesis equation can be obtained from the STFT synthesis equation (2.2).
Equations (2.5) and (2.6) comprise the DSTFT transform pair for some analysis and synthesis windows \( h[n] \) and \( f[n] \).

To ensure the validity the transform pair, \( h[n] \) and \( f[n] \) must be appropriately constrained. The nature of this constraint can be determined by substituting (2.5) into (2.6) and simplifying. The result is

\[
x[n] = \frac{1}{M} \sum_{k=0}^{M-1} \sum_{r=-\infty}^{+\infty} f[n - r]X[r, k]W_M^{-kn} = \sum_{m=-\infty}^{+\infty} x[m]h[r-m]W_M^{km}W_M^{-kn}
\]

For \( m = n \), the parenthesized term clearly simplifies to 1. Further, since \( W_M^q = W_M^{-qM} \) for all integers \( q \), a more general statement is that for \( m = n - qM \), the parenthesized term simplifies to 1. It is also straightforward to show that for \( m \neq n - qM \), the parenthesized term is 0. Thus, (2.7) can be transformed to

\[
x[n] = \sum_{q=-\infty}^{+\infty} x[n - qM] \sum_{r=-\infty}^{+\infty} f[n - r]h[r - n + qM]
\]

which is satisfied by assuming that

\[
\sum_{r=-\infty}^{+\infty} f[n - r]h[r - n + qM] = \delta[q]
\]

which can be simplified through a change of variables to obtain

\[
\sum_{m=-\infty}^{+\infty} f[qM - m]h[m] = \delta[q]
\]

Thus, the validity of the DSTFT transform pair (2.5) and (2.6) is ensured if \( h[n] \) and \( f[n] \) satisfy (2.10) for all integers \( q \).
2.3 Interpreting the DSTFT

The DSTFT provides a means of obtaining a two-dimensional sequence \( X[n, k] \) from the one-dimensional sequence \( x[n] \). Considering \( X[n, k] \) for fixed \( n \) or for fixed \( k \) leads to two interpretations.

By considering fixed values of \( n \), one obtains an interpretation of \( X[n, k] \) as a sequence of local spectra indexed by \( n \). For \( n = n_0 \), (2.5) can be written as

\[
X[n_0, k] = \sum_{m=-\infty}^{+\infty} y[m] W_M^{km}
\]

where

\[
y[n] = h[n_0 - n] x[n]
\]

That is, if \( y[n] \) is the result of applying the analysis window \( h[n] \) (time-reversed) to \( x[n] \) with \( h[0] \) centered on \( n = n_0 \), \( X[n_0, k] \) is more or less the DFT of \( y[n] \).\(^1\) So \( X[n_0, k] \) can be interpreted as a local spectrum of \( x[\cdot] \) around \( n_0 \) (where “locality” is determined by the duration and shape of \( h[n] \)).

By considering fixed values of \( k \), DSTFT analysis can be interpreted as the application of a filter bank to some input signal. For some \( k = k_0 \), equation (2.5) can be rewritten as the convolution

\[
X[n, k_0] = (h[n]) \ast (x[n] W_M^{k_0 n})
\]

Recalling the definition of \( W_M \), this equation implies that \( X[n, k_0] \) could be viewed as the result of filtering \( x[n] \) demodulated by \( e^{-j(2\pi/M)k_0 n} \) with the analysis filter (See figure 2-1).

Similarly, DSTFT synthesis (equation (2.6)) can be viewed as a process of filtering each of the outputs of the analysis filter bank with the synthesis filter, remodulating appropriately, and

\(^1\)More precisely, \( X[n_0, k] \) could be obtained by computing the DTFT of \( y[n] \) and then sampling at \( \omega = (2\pi/M)k \). By saying “more or less,” the text skirts the slippery issues of time aliasing and circular shifting which often accompany use of the DFT. Section 2.5 treats this issue in somewhat more detail.
then summing (see figure 2-2).

2.4 Decimating the DSTFT

The filter bank interpretation of the DSTFT leads to an interesting result: by selecting \( h[n] \) and \( f[n] \) appropriately, it is possible to recover \( x[n] \) exactly from a decimated (in \( n \)) version of \( X[n, k] \).

This is most easily understood by considering the case where \( h[n] \) is a low-pass filter with cutoff no greater than \( \pi / R \) (for some integer \( R \)). For each possible \( k = k_0 \), then, \( X[n, k_0] \) is a sequence that has been filtered by \( h[n] \) and is therefore band-limited to \( \pi / R \). Such a sequence can be decimated by a factor of \( R \) without loss of information; the original signal can be recovered exactly by upsampling and applying an appropriate interpolation filter [12].

By following this line of reasoning, we obtain equations (2.14) and (2.15), which comprise the decimated DSTFT relations.
Figure 2.2: Inverse filter bank interpretation of DSTFT synthesis

\[ X[pR, k] = \sum_{m=-\infty}^{+\infty} x[m]h[pR - m]W_M^{km} \]  \hspace{1cm} (2.14)

\[ x[n] = \frac{1}{M} \sum_{k=0}^{M-1} \sum_{p=-\infty}^{+\infty} f[n - pR]X[pR, k]W_M^{-kn} \]  \hspace{1cm} (2.15)

Again, this transform pair is only valid for appropriately constrained analysis and synthesis filters. Equation (2.16) articulates this constraint, the derivation of which parallels that shown for equation (2.10).

\[ \sum_{p=-\infty}^{+\infty} f[n - pR]h[pR - n + qM] = \delta[q] \]  \hspace{1cm} (2.16)

Although the earlier discussion was couched in terms of \( h[n] \) being a low-pass filter, it turns out that any analysis and synthesis filter pair which satisfy equation (2.16) will lead to perfect reconstruction. For the purpose of this thesis, however, \( h[n] \) is assumed to be a low-pass filter with cutoff \( \omega_H \). Both Portnoff and Holtzman argue that, given this assumption about \( h[n] \), a suitable choice for \( f[n] \) is a 1:p interpolating filter. In chapter three, we'll see that the
2.5 Efficient Computation

Section 2.3 hinted that the bulk of the computation required by the DSTFT could be reformulated in terms standard DFTs. This is advantageous, because it allows the use of “fast” DFT algorithms to significantly reduce the computational complexity of the DSTFT.

The following is based primarily on [1], with notational changes for consistency.

2.5.1 Analysis

Through a change of variables, equation (2.14) can be rewritten as

\[
X[pR, k] = \sum_{r=-\infty}^{+\infty} h[-r]z[r + pR]W_M^{k(r+pR)} = W_M^{pRk} \hat{X}[pR, k] \quad (2.17)
\]

Where \(X[pR, k]\) is a short-time transform taken with respect to a fixed time reference \((m = 0)\) and \(\hat{X}[pR, k]\) is a short-time transform taken with respect to a sliding time reference \((m = pR)\) which corresponds to the origin of the analysis window. \(\hat{X}[pR, k]\) can be expressed as a DFT

\[
\hat{X}[pR, k] = \sum_{m=0}^{M-1} \bar{z}_{pR}[m]W_M^{km} \quad (2.18)
\]

Where \(\bar{z}_{pR}[m]\) is obtained by applying the time-reversed analysis window, centered at \(n = pR\), to \(z[n]\) and time-aliasing with a “period” of \(M\).

\[
\bar{z}_{pR}[m] = \sum_{i=-\infty}^{+\infty} h[-m - iM]z[m + iM + pR] \quad (2.19)
\]
In equation (2.17), multiplication by \( W_{M}^{pRk} \) (in the frequency domain) corresponds to a circular shift by \( pR \) samples in the time domain. Thus \( X[pR, k] \) could also be obtained by taking the DFT of \( z_{pR}[m] \)

\[
X[pR, k] = \sum_{m=0}^{M-1} z_{pR}[m] W_{M}^{km} \tag{2.20}
\]

where \( z_{pR}[m] \) is obtained through a circular rotation of \( \tilde{z}_{pR}[m] \) by \( pR \) samples

\[
z_{pR}[m] = \tilde{z}_{pR}[(m - pR)_{M}] \tag{2.21}
\]

### 2.5.2 Synthesis

To compute the inverse DSTFT of a decimated short-time transform \( Y[pR, k] \), note that equation (2.15) can be written as

\[
y[n] = \sum_{p=-\infty}^{+\infty} f[n - pR] \frac{1}{M} \sum_{k=0}^{M-1} Y[pR, k] W_{M}^{-kn} \tag{2.22}
\]

In this form, the underlying DFT computation is readily apparent. The summation over \( k \) is simply the inverse DFT (with respect to \( k \)) of \( Y[pR, k] \)

\[
y_{pR}[n] = \frac{1}{M} \sum_{k=0}^{M-1} Y[pR, k] W_{M}^{-kn} \tag{2.23}
\]

Thus

\[
y[n] = \sum_{p=-\infty}^{+\infty} f[n - pR] y_{pR}[n] \tag{2.24}
\]

### 2.5.3 Implementation

In [1], Crochiere presents a weighted-overlap structure for DSTFT analysis/synthesis which takes advantage of the aforementioned efficiencies. The simulation codes used in this thesis
utilize that structure, with minor variations, to implement decimated DSTFT analysis/synthesis (see appendix A for details).
Chapter 3

Portnoff's Algorithm

The purpose of this chapter is to provide an overview of the system for time-scale modification described by Portnoff in [13, 16] and reformulated by Holtzman in [2]. Holtzman’s formulation differs from that of Portnoff primarily in method of implementation; in terms of the computation performed, the two are essentially identical.

3.1 A Model for Speech

Portnoff uses a simple model for speech production (see figure 3-1). The speech waveform $x[n]$ is considered to be the output of a time-varying linear filter $t[n, m]$ that is driven either by a stationary random sequence (white noise) $u[n]$ in the case of unvoiced speech or a quasi-periodic impulse train $v[n]$ in the case of voiced speech. The spacing of impulses in $v[n]$ corresponds directly to the pitch period of $x[n]$.

The time-varying filter $t[n, m]$ approximates the vocal tract transfer function; its variations correspond to the movement of physical articulators in the human speech system. As such, these time-variations are expected to be relatively slow compared to the rapid variations of $u[n]$, $v[n]$, and $x[n]$. Further, we assume that $t[n, m]$ is essentially time-invariant for the duration of its memory.

For voiced speech, we say that $t[n, m]$ is excited by a quasi-periodic impulse train $v[n]$, where
quasi-periodicity implies that \( v[n] \) is locally periodic. That is, if we consider the behavior of \( v[n] \) in the vicinity of some \( n_0 \) by examining \( v[n_0 + l] \) for small \( l \), we should find that it is periodic in \( l \).

Let \( P[n_0] \) denote the pitch period of \( v[n] \) in the vicinity of \( n = n_0 \). Let \( n_{imp} \) be the greatest index of \( v[n] \) such that \( v[n_{imp}] \) is an impulse and \( n_{imp} \leq n_0 \). Define \( D[n_0] \) to be the quantity \( (n_0 - n_{imp}) \) (see figure 3-2). Given these definitions, the quasi-periodicity of \( v[n] \) can be expressed more concretely as (an approximation valid for small \( l \)):

\[
v[n_0 + l] \approx \sum_{r=-\infty}^{+\infty} \delta \left( l + D[n_0] + rP[n_0] \right)
\]

By considering the Discrete Fourier Series (DFS) representation of \( v[n] \), equation (3.1) can be rewritten as a sum of harmonically related complex exponentials.

\[
v[n_0 + l] = \frac{1}{P[n_0]} \sum_{k=0}^{P[n_0]-1} e^{j2\pi k(l+D[n_0])/P[n_0]}
\]
Figure 3-2: Quasi-periodic impulse train

\[ = \frac{1}{P[n_0]} \sum_{k=0}^{P[n_0]-1} e^{j k (\phi[n_0] + \Omega[n_0] I + \phi_0)} \]  

(3.2)

where

\[ \Omega[n] = \frac{2\pi}{P[n]} \]  

(3.3)

\[ \phi_0 = \Omega[0] D[0] \]  

(3.4)

\[ \phi[n] = \Omega[n] D[n] + 2\pi I[n] - \phi_0 \]  

(3.5)

and \( I[n] \) is an integer whose value depends on \( n \).

Because we assume \( P[n] > 1 \), \( \Omega[n] \) can only take on values in the range

\[ 0 < \Omega[n] < 2\pi \]  

(3.6)

We call \( \Omega[n] \) the \textit{instantaneous frequency} of the fundamental harmonic. Notice that because \( P[n] \) is assumed to vary slowly with respect to \( n \) (because \( v[n] \) is quasi-periodic), \( \Omega[n] \) also varies slowly with respect to \( n \).

Portnoff introduces \( \phi_0 \) to ensure that \( \phi[0] = 0 \). By ensuring this, the time origin of a waveform is preserved when processed by Portnoff's TSM algorithm.

\( \phi[n] \) is referred to as the \textit{instantaneous phase} of the fundamental harmonic; the heretofore
unspecified quantity $I[n]$ can be determined by examining the relationship between $\phi[n + 1]$ and $\phi[n]$. When $D[n + 1] \neq 0$, we know that

\[
D[n + 1] = D[n] + 1 \\
\Omega[n + 1] \approx \Omega[n]
\]

and thus

\[
\phi[n + 1] \approx \Omega[n] + \Omega[n]D[n] + 2\pi I[n + 1] - \phi_0 \tag{3.7}
\]

Choosing $I[n + 1] = I[n]$ leads to

\[
\phi[n + 1] \approx \Omega[n] + \phi[n] \tag{3.8}
\]

For $D[n + 1] = 0$, we know that

\[
D[n + 1] = 0 \\
= D[n] + 1 - P[n] \\
\Omega[n + 1] \approx \Omega[n] 
\]

and thus

\[
\phi[n + 1] \approx \Omega[n] + \Omega[n]D[n] - \Omega[n]P[n] + 2\pi I[n + 1] - \phi_0 \tag{3.9}
\]

By definition, $\Omega[n]P[n] = 2\pi$. Choosing $I[n + 1] = I[n] + 1$ thus provides

\[
\phi[n + 1] \approx \Omega[n] + \phi[n] \tag{3.10}
\]

which is consistent with the previous case.
Note that, not unexpectedly, the instantaneous frequency is approximately the first difference of the instantaneous phase, and the instantaneous phase is a running summation of the instantaneous frequency.

Portnoff takes advantage of this interdependency by defining \( \phi[n] \) in terms of \( \Omega[n] \) as

\[
\phi[n] = \begin{cases} 
\sum_{r=1}^{n} \Omega[r] & \text{for } n > 0 \\
0 & \text{for } n = 0 \\
\sum_{r=n}^{0} \Omega[r] & \text{for } n < 0 
\end{cases}
\] (3.11)

This allows us to express \( v[n] \) as

\[
v[n] = \frac{1}{P[n]} \sum_{k=0}^{P[n]-1} e^{jk(\phi[n]+\phi_0)}
\] (3.12)

Recall that the speech waveform \( x[n] \) is modeled as the output of \( t[n, m] \) driven by the excitation waveform, as expressed by the following convolution sum.

\[
x[n] = \sum_{m=-\infty}^{+\infty} t[n, m] v[n - m]
\] (3.13)

Substituting equation (3.1) (the local harmonic approximation of \( v[n - m] \)) provides\(^1\)

\[
x[n] \approx \sum_{m=-\infty}^{+\infty} t[n, m] \left( \frac{1}{P[n]} \sum_{k=0}^{P[n]-1} e^{jk(\phi[n]-\Omega[n]m+\phi_0)} \right)
\approx \sum_{k=0}^{P[n]-1} c_k[n] e^{jk\phi[n]}
\] (3.14)

where \( \phi[n] \) is given by (3.11) and

\[
c_k[n] = \frac{1}{P[n]} \sum_{m=-\infty}^{+\infty} t[n, m] e^{jk(\phi_0-\Omega[n]m)}
\] (3.15)

\(^1\)which is valid because we assume that \( t[n, m] \) is essentially time-invariant for the duration of its memory.
Thus we see that a voiced speech waveform $z[n]$ can be represented as a sum of harmonically related complex exponentials whose complex amplitudes are given by the $c_k[n]$.

Both Portnoff [13, 16] and Holtzman [2] proceed from this point by developing a model of unvoiced speech. However, they both later conclude that their TSM algorithm does not require differentiation between voiced and unvoiced speech segments; that treating unvoiced speech segments in the same manner as voiced speech segments provides results that are consistent with the model of unvoiced speech. (Similar conclusions are presented in [6, 8, 17].) Because of this, this thesis provides no discussion of Portnoff's model of unvoiced speech. Further, all manipulations will henceforth assume a purely voiced model of speech.

### 3.2 Model of Time-Scale Modified Speech

Given a speech signal $z[n]$, let $z^\beta[n]$ represent a time-scale modified version of $z[n]$ where the articulation rate has been multiplied by the factor $\beta$. (Thus, $\beta > 1$ corresponds to speedup, while $\beta < 1$ corresponds to slowdown.) Further, let $z[\beta n]$ denote the signal obtained through a linear time-scaling of $z[n]$ by $\beta$ (dividing the sampling rate by $\beta$ through interpolation/decimation methods, as described in [12, 19]).

Portnoff's model of speech represents a speech waveform $z[n]$ by an initial fundamental harmonic phase, $\phi_0$, a pitch contour $\Omega[n]$, and $t[n, m]$, a time-varying filter which models the vocal tract response. The operation of time-scale modifying $z[n]$ to obtain $z^\beta[n]$ corresponds to linearly time-scaling the pitch contour $\Omega[n]$ and the time-varying features of $t[n, m]$. The time-scale modified excitation waveform $v^\beta[n]$ is thus a quasi-periodic impulse train with instantaneous frequency $\Omega[\beta n]$.

$$ v^\beta[n] = \frac{1}{P[\beta n]} \sum_{k=0}^{P[\beta n]-1} e^{jk(\phi[\beta n]/\beta + \phi_0)} \quad (3.16) $$

Equation (3.16) provides a harmonic representation of the time-scale modified excitation waveform. Observe that because $\phi[0] = 0$, the initial fundamental harmonic is $\phi_0$, as desired. Further, since $\Omega[n]$ is assumed to be slowly-time varying, $\phi[n + 1]$ can be approximated as
\[ \phi[n + 1] \approx \phi[n] + \Omega[n] \] (3.17)

By linearly time-scaling, we obtain

\[ \phi[\beta(n + 1)] \approx \phi[\beta n] + \Omega[\beta n] \beta \] (3.18)

Rearranging provides

\[ \Omega[\beta n] \approx \frac{\phi[\beta(n + 1)]}{\beta} - \frac{\phi[\beta n]}{\beta} \] (3.19)

Thus, \( \phi[\beta n]/\beta \), the instantaneous phase of the excitation waveform \( v^\beta[n] \), corresponds to an pitch contour of \( \Omega[\beta n] \), as desired.

The time-scale modified waveform \( x^\beta[n] \) is obtained by filtering \( v^\beta[n] \) with \( t[\beta n, m] \).

\[ x^\beta[n] = \sum_{m=\infty}^{+\infty} t[\beta n, m] v^\beta[n - m] \] (3.20)

Substituting a local approximation for \( v^\beta[n - m] \) (based on equation (3.16)) leads to

\[ x^\beta[n] \approx \sum_{m=\infty}^{+\infty} t[\beta n, m] \left( \frac{1}{P[\beta n]} \sum_{k=0}^{P[\beta n]-1} e^{ik(\phi[\beta n]/\beta - \Omega[\beta n]m + \phi_0)} \right) \]

\[ \approx \sum_{k=0}^{P[\beta n]-1} c_k[\beta n] e^{ik\phi[\beta n]/\beta} \] (3.21)

where

\[ c_k[\beta n] = \frac{1}{P[\beta n]} \sum_{m=\infty}^{+\infty} t[\beta n, m] e^{ik(\phi_0 - \Omega[\beta n]m)} \] (3.22)

Conveniently, (3.22) is simply a linearly time-scaled version of (3.15).

Note that the instantaneous phase \( \phi[n] \) in the preceding equations is an “unwrapped phase” angle (as defined by equation (3.11)), not the corresponding principal value. This differentiation
is particularly important in equation (3.21) because while adding multiples of $2\pi$ to the exponent is effectively invisible, adding multiples of $2\pi/\beta$ is not, for $1/\beta$ is not, in general, an integer.

### 3.3 Modification via the DSTFT

Although Portnoff and Holtzman utilize the previously described parametric model of speech in the derivation of their TSM algorithm, it is important to note that their actual modification scheme is not based on estimating and then modifying the parameters of that model. Instead, they argue that manipulation of a set of parameters more directly related to short-time Fourier analysis is an equivalent operation.

The manipulations described by Portnoff and Holtzman take advantage of the filter bank interpretation of the DSTFT (see figure 2-1). In their system, a conceptually simple modification is applied uniformly and independently to each of the "sub-band" signals. First, the instantaneous phase of each sub-band signal is estimated and divided by $\beta$; this new phase is substituted for the old phase to yield a phase-modified sub-band signal. Second, each phase-modified sub-band signal is linearly time-scaled by a factor of $\beta$. These modified sub-band signals are used as inputs to an inverse DSTFT filter bank (as per figure 2-2).

The following sections describe analysis, parameter modification, and synthesis, respectively. Note that while the equations shown in these sections demonstrate manipulations of the STFT (which is continuous in frequency), actual simulations approximate these manipulations by using the DSTFT (which is discrete in frequency) as described in Chapter 2.

#### 3.3.1 Analysis

Consider $x[n]$ to be a voiced speech waveform (as per equation (3.14)).

$$
x[n] = \sum_{k=0}^{P[n]-1} c_k[n] e^{j\phi[n]} 
$$

(3.23)

(where $c_k[n]$ is as defined in equation (3.22))
The STFT of $x[n]$ is given by

$$X[n, \omega] = \sum_{m=-\infty}^{+\infty} h[n - m] x[m] e^{-j\omega m}$$

$$= \sum_{m=-\infty}^{+\infty} h[n - m] \left( \sum_{k=0}^{P[n]-1} c_k[m] e^{j k \phi[m]} \right) e^{-j\omega m}$$

$$= \sum_{m=-\infty}^{+\infty} \sum_{k=0}^{P[n]-1} h[n - m] c_k[m] e^{j k \phi[m]} e^{-j\omega m} \quad (3.24)$$

This equation can be simplified somewhat by assuming a finite impulse response (FIR) analysis filter $h[n]$. This ensures that $h[n - m]$ will be zero when $|n - m| > L$ (where $L$ is on the order of half the duration of $h[n]$, assuming $h[n]$ is centered about $n = 0$). Thus we need only concern ourselves with values of $m$ which are “close” to $n$. Since our quasi-stationary model of speech assumes an excitation pitch period $P[n]$ which is slowly time-varying, we approximate the pitch period as constant over the duration of $h[n]$.

$$P[m] \approx P[n] \quad (3.25)$$

Since the instantaneous frequency of the fundamental harmonic is also assumed to be slowly time-varying, a linearly time-varying approximation of the instantaneous phase can be used:

$$\phi[m] \approx \phi[n] + \Omega[n](m - n) \quad (3.26)$$

Finally, by assuming that the vocal tract is roughly stationary for the duration of $h[n]$, we can further assume that

$$c_k[m] \approx c_k[n] \quad (3.27)$$

Substituting these assumptions into equation (3.24) yields

$$X[n, \omega] \approx \sum_{m=-\infty}^{+\infty} \sum_{k=0}^{P[n]-1} h[n - m] c_k[n] e^{j k (\phi[n] + \Omega[n] (m - n))} e^{-j\omega m}$$

$$\approx \sum_{k=0}^{P[n]-1} c_k[n] e^{j k (\phi[n] - n \Omega[n])} \left( \sum_{m=-\infty}^{+\infty} h[n - m] e^{j (k \Omega[n] - \omega) m} \right) \quad (3.28)$$
The summation over \( m \), after a change in variables, can be recognized as the DTFT of \( h[n] \) evaluated at a particular frequency:

\[
\sum_{m=-\infty}^{+\infty} h[n - m]e^{j(k\Omega[n]-\omega)m} = \sum_{m'=-\infty}^{+\infty} h[m']e^{j(k\Omega[n]-\omega)(n-m')}
\]

\[
= e^{j(k\Omega[n]-\omega)n} \sum_{m'=-\infty}^{+\infty} h[m']e^{-j(k\Omega[n]-\omega)m'}
\]

\[
= e^{j(k\Omega[n]-\omega)n} H(k\Omega[n] - \omega)
\]

(where \( H(\omega) \) is the DTFT of the analysis filter \( h[n] \))

Substituting this back into our approximation for \( X[n, \omega] \) provides

\[
X[n, \omega] \approx \sum_{k=0}^{P[n]-1} c_k[n]H(k\Omega[n] - \omega)e^{j(k\phi[n]-\omega_n)}
\]

(3.29)

Thus, for a particular \( n = n_0 \), the STFT of \( x[n] \) can be viewed to be the sum of \( P[n_0] \) images of \( H(\omega) \), each shifted by \( k\Omega[n_0] \) in frequency and weighted by \( c_k[n_0]e^{j(k\phi[n]-\omega_n)} \).

If we assume that the bandwidth of \( H(\omega) \) is less than the smallest possible fundamental pitch \( \Omega[n] \), then the images of \( H(\omega) \) don’t overlap and \( X[n, \omega] \) can be expressed as

\[
X[n, \omega] = \begin{cases} 
  c_k[n]H(k\Omega[n] - \omega)e^{j(k\phi[n]-\omega_n)} & \text{for } |w - k\Omega[n]| < \omega_h \\
  0 & \text{otherwise}
\end{cases}
\]

(3.30)

where \( 0 \leq k \leq P[n] - 1 \) and \( \omega_h \) is the cutoff frequency of the (low-pass) analysis filter \( h[n] \).

For the cases where \( X[n, \omega] \) does not go to zero, then, it can be expressed as

\[
X[n, \omega] = c_k[n]H(k\Omega[n] - \omega)e^{j(k\phi[n]-\omega_n)}
\]

(3.31)

for some appropriate value of \( k \). For fixed \( \omega \), this provides a sequence in \( n \) which corresponds to one of the “sub-bands” of the filter-bank interpretation of the DSTFT (see figure 2-1).

When \( X[n, \omega] \) does go to zero, its phase is undefined. Portnoff doesn’t seem to treat this issue. Holtzman argues that it is acceptable to treat \( X[n, \omega] \) as if it did not vanish, assuming zero
phase when $X[n, \omega]$ goes to zero; any discontinuities in the phase are masked by a negligibly small amplitude. We therefore assume that equation (3.31) is valid for all values of $\omega$.

Each such sub-band signal can be represented as

$$X[n, \omega] = a[n, \omega]e^{i\nu[n, \omega]}$$  \hspace{1cm} (3.32)

where

$$a[n, \omega] = c_k[n]H(k\Omega[n] - \omega)$$  \hspace{1cm} (3.33)
$$\nu[n, \omega] = k\phi[n] - \omega n$$  \hspace{1cm} (3.34)

The component $a[n, \omega]$ is a product of the magnitude of $X[n, \omega]$ and what Portnoff refers to as the "phase-modulation" component of the phase of $X[n, \omega]$. Because it is a product of slowly time-varying signals, $a[n, \omega]$ is slowly time-varying.

The component $\nu[n, \omega]$ is what Portnoff refers to as the "frequency-modulation" component of the phase of $X[n, \omega]$. Because it is a sum of components that locally vary linearly with $n$, $\nu[n, \omega]$ varies linearly with $n$. It is this "instantaneous phase" that must be estimated and modified after the sub-band signal has been linearly time-scaled.

According to equation (3.11), $\phi[n]$ is an unwrapped phase, not the principal value of that phase. This implies that $\nu[n, \omega]$ is also an unwrapped phase, which becomes especially important during parameter modification, because, in general, $\text{PV}\{\nu[n, \omega]/\beta\} \neq \text{PV}\{\nu[n, \omega]\}/\beta$. Further, combining equations (3.11) and (3.34) provides

$$\nu[n, \omega] = \begin{cases} 
0 & \text{for } n = 0 \\
\sum_{r=1}^{n} k\Omega[r] - \omega & \text{for } n > 0 
\end{cases}$$  \hspace{1cm} (3.35)

This can be viewed as an analogue to equation (3.11) where the instantaneous phase $\phi[n]$ is expressed as a running sum of the instantaneous frequency $\Omega[n]$. Such an interpretation leads us to define the instantaneous frequency $\Omega[n, \omega]$ of $X[n, \omega]$ as

$$\Omega[n, \omega] = k\Omega[r] - \omega$$  \hspace{1cm} (3.36)
3.3.2 Modification

In order to effect time-scale modification using the (D)STFT, two steps are necessary. First, the instantaneous phase \( \nu[n, \omega] \) must be estimated, divided by \( \beta \), and substituted for the phase of \( X[n, \omega] \) to obtain \( Y[n/\beta, \omega] \). Second, \( Y[n/\beta, \omega] \) must be linearly time-scaled to obtain \( Y[n, \omega] \), the STFT of the desired time-scale modified waveform.

Phase Modification

Equation (3.32) can be rewritten in polar form as

\[
X[n, \omega] = M[n, \omega]e^{j\theta[n, \omega]} \tag{3.37}
\]

where

\[
M[n, \omega] = |a[n, \omega]| \tag{3.38}
\]

\[
\theta[n, \omega] = \text{arg}\{a[n, \omega]\} + \nu[n, \omega] \tag{3.39}
\]

Taking the first backward difference with respect to \( n \) provides

\[
\nabla_n \theta[n, \omega] = \nabla_n \text{arg}\{a[n, \omega]\} + \nabla_n \nu[n, \omega] \tag{3.40}
\]

Since \( a[n, \omega] \) is slowly time-varying, so is its phase (except, possibly, for jumps of \( \pi \) and \( 2\pi \); these are treated later). Therefore, we argue that \( \nabla_n \text{arg}\{a[n, \omega]\} \) is a negligible quantity and that

\[
\nabla_n \theta[n, \omega] \approx \nabla_n \nu[n, \omega] \tag{3.41}
\]

From equation (3.36) we know that the first backward difference of \( \nu[n, \omega] \) is \( \Omega[n, \omega] \), the instantaneous frequency of \( X[n, \omega] \). Thus, a reasonable estimate \( \tilde{\Omega}[n, \omega] \) of the instantaneous
The corresponding estimate $\tilde{\nu}[n, \omega]$ of the instantaneous phase is given by

$$
\tilde{\nu}[n, \omega] = \begin{cases} 
0 & \text{for } n = 0 \\
\sum_{r=1}^{n} \tilde{\Omega}[n, \omega] & \text{for } n > 0
\end{cases} \tag{3.43}
$$

($n$ is assumed to be non-negative)

To produce $Y[n/\beta, \omega]$, we substitute $\tilde{\nu}[n, \omega]/\beta$ for $\nu[n, \omega]$ in equation (3.32).

$$
Y[n/\beta, \omega] = a[n, \omega] e^{j \tilde{\nu}[n, \omega]/\beta} \tag{3.44}
$$

This “phase substitution” of $\tilde{\nu}[n, \omega]/\beta$ for $\nu[n, \omega]$ is accomplished by modulating each sub-band with an appropriate complex exponential.

$$
Y[n/\beta, \omega] = X[n, \omega] e^{j (\frac{1}{\beta} - 1) \tilde{\nu}[n, \omega]} = a[n, \omega] e^{j \nu[n, \omega]} e^{j \frac{1}{\beta} \tilde{\nu}[n, \omega]} = a[n, \omega] e^{j \nu[n, \omega]/\beta} = P[n] \sum_{k=0}^{P[n]-1} c_k[n] H(k\Omega[n] - \omega) e^{j(k\phi[n]/\beta - \omega n)} \tag{3.45}
$$

Our estimate $\tilde{\nu}[n, \omega]$ of the instantaneous phase of $X[n, \omega]$ is defined in terms of $\theta[n, \omega]$. Unfortunately, $\theta[n, \omega]$ can not be obtained directly from $X[n, \omega]$. Because of equation (3.39) and the fact that $\nu[n, \omega]$ is an unwrapped phase, $\theta[n, \omega]$ is also an unwrapped phase. From $X[n, \omega]$ we can only obtain the principal value of the phase:

---

2This is one of the few points where the works of Portnoff and Holtzman differ. While Holtzman uses the first backward difference of $\theta[n, \omega]$ as an estimate of the instantaneous frequency (as suggested here), Portnoiff uses the mean central difference operator to obtain his estimate. Comparison of these methods through simulation revealed no significant difference in terms of output speech quality.
\[ \theta_{PV}[n, \omega] = PV\{\arg X[n, \omega]\} \]  
(3.46)

Considered as a function of \(n\), the principal value of the phase can contain jumps of \(2\pi\).\(^{3}\) We can express this as

\[ \theta_{PV}[n, \omega] = \theta[n, \omega] + 2\pi I_1[n, \omega] \]  
(3.47)

where \(I_1[n, \omega]\) is an integer-valued function which represents the jumps of \(2\pi\) in \(\theta_{PV}[n, \omega]\). The first backward difference of the principal value is

\[ \nabla_n \theta_{PV}[n, \omega] = \nabla_n \theta[n, \omega] + 2\pi I_2[n, \omega] \]  
(3.48)

where \(I_2[n, \omega]\) is another integer-valued function.

Again, \(X[n, \omega]\) can be interpreted as the output of a low-pass filter with cutoff \(\omega_h\). Given equation (3.41) and our assumption that \(\Omega[n, \omega]\) is slowly time-varying, it seems reasonable to assume that

\[ |\nabla_n \theta[n, \omega]| < \omega_h \]  
(3.49)

Since \(\omega_h < \pi\), \(\nabla_n \theta[n, \omega]\) can be obtained from \(\nabla_n \theta_{PV}[n, \omega]\) by adding or subtracting integer multiples of \(2\pi\) until equation (3.49) is satisfied. And, according to equation (3.42), the resulting value is \(\tilde{\Omega}[n, \omega]\), our estimate of the instantaneous frequency of \(X[n, \omega]\).

Thus we have demonstrated that it is reasonably straightforward to obtain the phase modified STFT \(Y[n/\beta, \omega]\) from \(X[n, \omega]\). However, as suggested at the end of chapter 2, the simulation codes used in this thesis do not calculate the (D)STFT for all values of \(n\). The (D)STFT is only computed for every \(R\)th value of \(n\) in order to avoid redundant computation.

\(^{3}\)Here lies the other significant difference between the works of Portnoff and Holtzman. Portnoff also concerns himself with jumps of \(\pi\) which occur when the real and imaginary parts of \(a[n, \omega]\) change sign at the same time. Holtzman ignores this without comment. It would seem safe to ignore these jumps of \(\pi\) since \(a[n, \omega]\) is slowly time-varying, they should only occur when the magnitude of \(X[n, \omega]\) is small. The simulation codes used in this thesis do not attempt to detect and correct for these jumps of \(\pi\).
With the decimated transform,

\[ \nabla_s \theta[sR, \omega] \approx R \nabla_n \theta[n, \omega] \bigg|_{n=sR} \]  

(3.50)

because \( \nabla_n \theta[n, \omega] \) is slowly time-varying. Combining with (3.49) provides

\[ |\nabla_s \theta[sR, \omega]| < \omega_h R \]  

(3.51)

Assuming \( \omega_h \) and \( R \) have been chosen such that \( \omega_h R < \pi \), we can obtain \( \nabla_s \theta[sR, \omega] \) from \( \nabla_s \theta_{PV}[sR, \omega] \) by adding or subtracting integer multiplies of \( 2\pi \) until this constraint is satisfied, much as with the undecimated case. The estimate of the instantaneous frequency is, then,

\[ \bar{\Omega}[sR, \omega]R = \nabla_s \theta[sR, \omega] \]  

(3.52)

The corresponding estimate for the instantaneous phase is

\[ \bar{\varphi}[sR, \omega] = \begin{cases} 
0 & \text{for } s = 0 \\
\sum_{r=1}^{n} \bar{\Omega}[rR, \omega]R & \text{for } n > 0 
\end{cases} \]  

(3.53)

\( s \) is assumed to be non-negative.

Phase substitution is accomplished, as before, by modulating each sub-band signal with an appropriate complex exponential.

\[ Y[sR/\beta, \omega] = X[sR, \omega]e^{i(\frac{1}{\beta}-1)\bar{\Omega}[sR, \omega]} \]

\[ = a[sR, \omega]e^{i\bar{\varphi}[sR, \omega]/\beta} \]

\[ = \sum_{k=0}^{P[sR]-1} c_k[sR]H(k\Omega[sR] - \omega)e^{i(k\varphi[sR]/\beta - \omega sR)} \]  

(3.54)
Linear Time-Scaling

Chapter two described the decimated DSTFT analysis/synthesis methods used in this work. Recall that decimation was justified by arguing that since each of the signals $X[n, k_0]$ can be viewed as the output of a low-pass filter with cutoff $\omega_h$, those signals are each band-limited to $\omega_h$. As long as the decimation factor does not exceed $\pi/\omega_h$, the original signals $X[n, k]$ can be recovered exactly by upsampling and interpolating. In this context, linear time-scaling is accomplished simply by using different decimation/interpolation factors in the analysis and synthesis stages of the TSM system; decimating by a factor of $R_A$ and interpolating by a factor of $R_S$ leads to an effective linear time-scaling by a factor of $\beta = R_A/R_S$.4

Thus, the time-scale modified decimated short-time transform is obtained trivially:

$$Y[sR_S, \omega] = Y[sR_A/\beta]$$  \hspace{1cm} (3.55)

Of course, the use of differing decimation/interpolation factors in DSTFT analysis and synthesis changes the equations which provided a basis for (2.16), the constraint which dictated the “shape” and behavior of the synthesis filter $f[n]$. Instead of deriving an analogous constraint for use with differing decimation/interpolation factors, Portnoff and Holtzman follow the tone of the argument presented at the end of section 2.4 and utilize a 1-to-$R_S$ interpolating filter as their synthesis filter. Thus $Y[sR_S, \omega]$ corresponds to an undecimated transform

$$Y[n, \omega] = \alpha[\beta n, \omega]e^{j\nu[\beta n, \omega]/\beta}$$

$$= \sum_{k=0}^{P[\beta n]-1} c_k[\beta n]H(k\Omega[\beta n] - \omega)e^{j(k\phi[\beta n]/\beta - \omega n)}$$  \hspace{1cm} (3.56)

4Since the analysis and synthesis decimation/interpolation factors must be integers, this system can only implement linear time-scaling by rational values of $\beta$. This is characteristic of interpolation/decimation based rate-change schemes [12]. It is not a significant limitation since any real number can be approximated by a rational number with arbitrary precision.
3.3.3 Synthesis

Let \( Y[n, \omega] \) denote a modified short-time transform from which a time-scale modified speech waveform is to be synthesized. From the previous section, we know that

\[
Y[n, \omega] = \sum_{k=0}^{P[\beta n] - 1} c_k[\beta n] H(k \Phi[\beta n] - \omega) e^{i(k \phi[\beta n]/\beta - \omega n)}
\]  
(3.57)

Both Portnoff and Holtzman show that this corresponds to the desired time-scale modified speech (equation (3.21)) by grinding through the inverse STFT manipulations. It is quite a bit easier to recognize that by starting with our harmonic representation for time-scale modified speech (equation (3.21)) and paralleling the manipulations shown in section 3.3.1, we would obtain precisely equation (3.57). Thus, after modifying the (D)STFT as described in the previous section, the desired time-scale modified speech signal is obtained by simply computing the inverse STFT of that modified transform.

3.4 Implementation Details

This section provides specific details about the simulation of Portnoff/Holtzman's TSM system used in this thesis study. Readers wishing further detail are referred to the simulation code listings provided in appendix A.

3.4.1 Analysis Filter

Design of an analysis filter requires considering the trade-off between time and frequency resolution. In terms of time resolution, we would like the duration of the filter to be as short as possible to ensure that speech waveforms are very nearly stationary over that duration. In terms of frequency resolution, the bandwidth of the analysis filter should be small enough to ensure each DSTFT sub-band contains contributions from at most one harmonic component of a speech waveform. Recalling the discussion in section 3.3.1, we know that to satisfy this frequency constraint, the bandwidth of the analysis filter \( H(\omega) \) must be less than the smallest
expected fundamental frequency $\Omega[n]$.

Following the lead of Portnoff and Holtzman, we use a Hamming window for the analysis filter. According to [11], the bandwidth of an $N$-point Hamming window is $4f_s/N$. (This thesis uses a sampling frequency $f_s$ of 8 kHz.) The fundamental pitch of speech rarely falls below 100 Hz. Using this as a lower bound on $\Omega[n]$, we see that our analysis filter must be at least $\approx 20$ samples (40 ms) long.

In [16], Portnoff reports use of 25 to 50 ms Hamming windows as analysis filters (corresponding to filter bandwidths of approximately 80 to 160 Hz). In [2], Holtzman suggests a 256 point Hamming window (10 kHz sampling rate), although he mistakenly uses filter cutoff instead of filter bandwidth to justify this duration.

The simulations discussed in this thesis utilize a 321 point Hamming window for analysis.

### 3.4.2 Synthesis Filter

As discussed previously, the synthesis filter should be a 1-to-$R_S$ interpolating filter (where $R_S$ is the interpolation factor used in the synthesis stage). Both Portnoff and Holtzman use interpolating filters designed using the Oetken's algorithm [10].

In [6], Griffin and Lim suggest a "LSEE-MSTFT" synthesis filter which is derived by applying a simple transformation to the corresponding analysis filter:

$$f[n] = \frac{h[n]}{\sum_{r=-\infty}^{+\infty} h^2[n + rR_S]}$$  \hspace{1cm} (3.58)

This simple method generates reasonably good interpolation filters which satisfy equation (2.16) (the decimated DSTFT analysis/synthesis filter constraint) exactly for large enough $M$ (where "large enough" is as discussed in section 3.4.4).

This thesis uses synthesis filters derived from the corresponding analysis filters using the LSEE-MSTFT transformation. Simulation revealed no discernible qualitative difference between voice processed with such a synthesis filter and voice processed with one of Oetken's filters.
3.4.3 Decimation and Interpolation Factors

In section 3.3.2, to ensure proper estimation of time-unwrapped phase, we required \( \omega_h R_A < \pi \), which can be rewritten as

\[
R_A < \frac{\pi}{\omega_h}
\]  

(3.59)

which simply states that (not surprisingly), \( R_A \) must not exceed the Nyquist interval for a sequence which has been band-limited to \( \omega_h \).

According to [11], an \( N \)-sample Hamming window has a cutoff of \( \omega_h = 4\pi/N \); thus dictating \( R_A < N/4 \). In our case, with \( N = 321 \), this implies \( R_A \leq 80 \).

For this thesis, we (conservatively) restrict \( R_A \) to be less than 40.

The linear time-scaling and phase modification operations affect the bandwidth of sub-band signals \( X[n, \omega] \). In order to avoid aliasing (in \( \psi \), the frequency variable corresponding to \( n \)) due to bandwidth expansion, further constraints must be placed on \( R_A, R_S, \) and \( \beta \).

In particular, the bandwidth of a signal which has been linearly time-scaled by a factor of \( \beta \) is \( \beta \) times the bandwidth of the original signal. On the other hand, the phase modification process (dividing the instantaneous phase of the sub-band signal by \( \beta \)) results in a signal which is approximately \( 1/\beta \) times the bandwidth of the original signal. Thus, taken as a whole, sub-band signal modification provides modified signals which have approximately the same bandwidth as the original signals.

If, however, either \( \beta R_A \) or \( \frac{1}{\beta} R_A \) is greater than the Nyquist interval for \( X[n, \omega] \), we run the risk of aliasing (in \( \psi \)). Recalling that \( \beta = R_A/R_S \), we see that in order to avoid aliasing, we must restrict \( R_S \) according to

\[
\frac{R_A^2}{T_{nyq}} < R_S < T_{nyq}
\]  

(3.60)

Use of \( N \) sample Hamming windows for (D)STFT analysis provides sub-band sequences with

\[\text{Because Portnoff attempts to remove jumps of } \pi \text{ as well as those of } 2\pi, \text{ he requires that } R_A < N/8 \text{ (see [16]).}\]
3.4.4 Frequency Resolution

Assuming $h[n]$ and $f[n]$ possess even symmetry, the analysis/synthesis filter constraint expressed by equations (2.10) and (2.16) can be trivially satisfied for $q \neq 0$ by choosing $M$, the number of frequency samples, such that

$$M > (L_h + L_f)/2$$

(3.61)

where $L_h$ and $L_f$ are the durations (in samples) of $h[n]$ and $f[n]$, respectively.

In our case, $L_h = L_f$, so this can be simplified to requiring $M > L_h$. Since we choose $L_h = 321$, $M = 512$ should be satisfactory. (Both Portnoff and Holtzman also suggest using 512 frequency samples.)

Note that equations (2.10) and (2.16) could be satisfied by $(M, h[n], f[n])$ tuples which do not satisfy equation (3.61). Further, tuples which satisfy equation (3.61) but not (2.10) and (2.16) are easy to imagine. This constraint on $M$ simply allows us to, for example, simplify equation (2.10) as

$$\sum_{m=-\infty}^{+\infty} f[-m]h[m] = 1$$

(3.62)

Such simplification makes the task of designing $h[n]$ and $f[n]$ somewhat easier.

3.5 Results

This section describes the results obtained using the algorithm described in the previous sections. Plots of several original and processed test signals are shown.
3.5.1 Periodic Signals

When operating on periodic signals (or, more generally, signals which are sampled representations of continuous-time periodic signals), the Portnoff/Holtzman TSM algorithm provides high-quality output for all possible values of $\beta$.

From figure 3-1, we know that an impulse train convolved with some filter response is a reasonable first approximation of a (voiced) speech waveform. Figure 3-3 shows such a waveform; it is the result of convolving a 131 Hz impulse train with an exponentially decaying response. (131 Hz is chosen as a representative fundamental frequency of speech.)

Figure 3-4 shows the result of subjecting that synthetic test signal to a mild slowdown using the Portnoff/Holtzman algorithm. Except for a slight amount of background noise and a start-up transient, the time-scale modified waveform is virtually indistinguishable from the original; even waveform shape is preserved.
Figure 3.4: Periodic test signal processed by Portnoff/Holtzman algorithm, $\beta = \frac{10}{37}$ (200 ms duration)

3.5.2 Quasi-Periodic Signals

Of course, speech waveforms are not, in general, periodic signals such as those discussed in the previous section. Our first approximation of a speech waveform could be improved by allowing the fundamental frequency of the impulse train to oscillate slowly.

Figure 3.5 shows such a test signal. The instantaneous frequency of the underlying impulse train oscillates (at 10 Hz) between 131 and 181 Hz; this impulse train is convolved with an exponentially decaying response to produce the test signal. (Once again, these parameters are chosen to be representative of a possible speech waveform.)

Figure 3.6 shows the quasi-periodic test signal after being subjected to the same mild slowdown used in figure 3.4. Note that waveform shape is not preserved; the signal quickly becomes incoherent and barely recognizable as locally periodic. Although the signal remains somewhat recognizable to the ear, it sounds extremely "scratchy" and is easily distinguished from the original.

Since speech passages are similarly quasi-periodic, they suffer similar degradations when pro-
Figure 3-5: Original quasi-periodic test signal (200 ms duration)

Figure 3-6: Quasi-periodic test signal processed by Portnoff/Holtzman algorithm, $\beta = \frac{20}{47}$ (200 ms duration)
cessed by this TSM algorithm. Processed speech sounds hollow (as if spoken through a short tube) and scratchy (although not as scratchy as the processed quasi-periodic test waveform). A high degree of intelligibility is maintained as well as a significant amount of speaker-recognizability. When compared directly with the original speech passage, the processed speech sounds somewhat unnatural and is easily distinguishable. In contrast, speech passages processed by Portnoff and Holtzman sound nearly natural and are harder to distinguish from the original speech. This would seem to indicate a fallacy in either the derivation or implementation of the Portnoff/Holtzman system used in this thesis.

3.5.3 Integer Rates of Slowdown

Curiously, for integer rates of slowdown ($\beta = 1/Q$ for some integer $Q$), the algorithm provides somewhat better output quality. Figure 3-7 shows the quasi-periodic test waveform from figure 3-5 slowed two times with the Portnoff/Holtzman TSM algorithm ($\beta = 0.5$). Comparing this with figure 3-6 reveals significant differences: although waveform shape is not entirely preserved with $\beta = 0.5$, the resulting waveform clearly retains its local periodicity and much more of the original waveform shape in the non-integer slowdown case (figure 3-6). These differences are apparent to the human ear; the signal shown in figure 3-7 sounds far more like the unmodified test signal than that shown in figure 3-6.

This effect is apparent with processed speech files as well. Speech processed at integer rates of slowdown sounds somewhat better than speech processed at non-integer rates of slowdown (or speech processed for speedup). The processed speech retains some degradation, however, and despite the improvement remains fairly easily distinguishable from natural speech.

3.6 Explanation

The output speech quality problems described in the previous section might be attributable to two things:

- The fact that an arbitrary $X[n, \omega]$ is not necessarily a valid STFT.
Figure 3-7: Quasi-periodic test signal processed by Portnoff/Holtzman algorithm, $\beta = \frac{1}{2}$ (200 ms duration)

- A failure to maintain the proper harmonic phase relationship among individual sinusoidal components.

The STFT of a speech waveform is a redundant representation of the information present in the original waveform. The existence of redundancy implies that for an arbitrary $X[n, \omega]$ to be a valid STFT, it must be self-consistent; it must possess a certain structure. Equation (3.34) hints at the form of that structure, but, for the most part, a complete understanding of the structure required for validity does not seem to exist. However, since we model a speech waveform as the sum of discrete sinusoidal components, we can attempt to understand the required structure by understanding the structure required for each sinusoidal component. In particular, equation (3.34) specifies that in the vicinity of a sinusoidal component, phase variations across $n$ and $\omega$ should follow a predictable pattern. Observation of modified phase waveforms during simulation of the Portnoff/Holtzman algorithm indicates that this simple constraint is often violated when processing non-periodic signals.

Recall the argument used to justify the suitability equation (3.31) for values of $\omega$ which don’t satisfy $|\omega - k\Omega[n]| < \omega_h$: any phase discontinuities that occur in these cases should be masked
by the negligibly small amplitude of $X[n, \omega]$. If we concern ourselves only with an individual sub-band signal (corresponding to some $\omega = \omega_0$), this argument is sound; phase discontinuities when $|X[n, \omega_0]|$ is small should have little effect on the continuity of $X[n, \omega]$ considered as a sequence in $n$. What this argument neglects is the possible introduction of discontinuities in phase across $\omega$. Recognizing that our frequent quasi-periodic approximations are not quite exact, equation (3.30) might be more accurately written

$$
X[n, \omega] = \begin{cases} 
    c_k[n] H(k\Omega[n] - \omega) e^{i(k\phi[n] - \omega n)} & \text{for } |w - k\Omega[n]| < \omega_k \\
    \epsilon[n, \omega] & \text{otherwise}
\end{cases}
$$

(3.63)

where $|\epsilon[n, \omega]|$ is negligibly small and $\text{arg}\{\epsilon[n, \omega]\}$ is essentially a random process of some sort.

Since our estimates $\Omega[n, \omega]$ and $\varphi[n, \omega]$ of the instantaneous frequency and phase of $X[n, \omega]$ are based on measurement of $\text{arg}\{X[n, \omega]\}$, those cases when $X[n, \omega] = \epsilon[n, \omega]$ have the potential of introducing significant discontinuities (across $\omega$) in $\varphi[n, \omega]$.

For periodic signals, the values of $\omega$ for which $X[n, \omega] = \epsilon[n, \omega]$ are the same for all $n$. Discontinuities in $\varphi[n, \omega]$ as a result of this effect will always be confined to values of $\omega$ for which the magnitude of $X[n, \omega]$ is negligibly small, and the phase modification operation (equations (3.45) and (3.54)) will never introduce phase discontinuities in the vicinity of any sinusoidal components of $X[n, \omega]$.

For non-periodic signals, the range of $\omega$ for which $X[n, \omega] = \epsilon[n, \omega]$ is not fixed. In particular, values of $\omega$ which fall within this range for some $n = n_0$ may not fall within this range for some $n_1 > n_0$. Phase discontinuities introduced into $\varphi[n, \omega]$ at $n_0$ could possibly be "in the vicinity" of a sinusoidal component (satisfy $|w - k\Omega[n]| < \omega_k$) at $n = n_1$. Phase modification could thus possibly introduce phase discontinuities into $Y[n, \omega]$ which violate the constraint implied by equation (3.34). The effect of such phase discontinuities on waveforms obtained through application of (D)STFT synthesis is largely not understood.

It is straightforward to see that these random phase contributions could also lead to situations where the harmonic phase relationship between sinusoidal components of $Y[n, \omega]$ is not maintained. This type of degradation could explain cases where waveform shape is not preserved after transformation while local periodicity is.
The fact that the algorithm performs slightly better for integer rates of slowdown is not completely understood. Portnoff hints that values of $\beta$ for which $1/\beta$ is an integer are a special case [16], but does not provide further illumination. Holtzman noticed similar results in his simulations [3] but failed to provide an explanation.
Chapter 4

Revised Algorithm

This chapter describes a revised TSM algorithm which attempts to correct one of the possible problems of the Portnoff/Holtzman algorithm pointed out at the end of chapter three. The first section describes the changes that lead to the revised algorithm. The second section provides a subjective comparison of the algorithms' performance. Finally, the third section discusses the effects that varying the frequency resolution (of the revised algorithm) has on processed speech quality.

4.1 How it Works

The development of this revised algorithm was influenced significantly by [9, 17, 18], in which Quatieri and McAulay describe a speech analysis/synthesis system and TSM algorithm based on a summed-sinusoid representation of speech.

One of the possible problems with the Portnoff/Holtzman algorithm is that the modified transform $Y[n, \omega]$ is not necessarily a valid STFT. In particular, we have seen that the phase in the vicinity of a sinusoidal component may be discontinuous and incoherent. One solution to this problem might be to try and detect these situations and then somehow patch them before applying the synthesis algorithm. This would seem ponderous, however, since our only understanding of the structure of the STFT is based on numerous quasi-periodic approximations,
many of which may be less than precise.

Although somewhat ad hoc, a better approach to synthesis is to estimate the amplitude and phase of each underlying sinusoidal component and synthesize these components directly. Synthesis of this type can be accomplished by "simply overlapping and adding time-weighted segments of each of the sinusoidal components," as suggested by McAulay and Quatieri [9]. In this way, the troublesome problem of obtaining a more exact understanding of the structure required for $Y[n, \omega]$ to be a valid STFT can be avoided entirely.\footnote{As suggested in chapter one, one of the goals of this thesis was to obtain an understanding of why the described implementation of the Portnoff/Holtzman algorithm did not provide the expected high level of output quality. This revision is seen more as an intermediate step between the Portnoff/Holtzman algorithm and the McAulay/Quatieri algorithm; it serves more as a means to investigate the hypothesis that phase discontinuities contribute significantly to the observed performance of the Portnoff/Holtzman algorithm than as a proposal for a high-quality TSM system.}

In terms of the framework used to implement the Portnoff/Holtzman algorithm, this can be accomplished with only a few minor changes. First, within each analysis frame, the sinusoidal components can be located by picking the peaks off of the magnitude of $X[sR_A, \omega]$. Consider \{$\omega_1, ..., \omega_l$\} to a set of such peak locations. $Y[sR_S, \omega]$ is then obtained in the following manner:

$$\begin{align*}
|Y[sR_S, \omega]| &= \begin{cases} 
|X[sR_A, \omega]| & \text{for } \omega = \omega_i \ (i = 1, ..., l) \\
0 & \text{otherwise}
\end{cases} 
\end{align*}
$$

(4.1)

For $\omega = \omega_i \ (i = 1, ..., l)$,

$$\mathcal{L}\{Y[sR_S, \omega]\} = \mathcal{L}\{Y[(s - 1)R_S, \omega]\} + \frac{1}{\beta} \tilde{\Omega}[sR_A]R_A$$

(4.2)

where $\tilde{\Omega}[sR_A]R_A$ is obtained as before.

The phase of $Y[sR_S, \omega]$ for $\omega \neq \omega_i$ is determined by the phase of the nearest sinusoid (in that frame) and the rule expressed by equation (3.34). This may seem superfluous since the magnitude of $Y[sR_S, \omega]$ is non-zero only for $\omega = \omega_i$. However, since the phase at a sinusoidal "peak" is obtained by offsetting from the phase of the previous synthesis frame (as per equation (4.2)), this step ensures that the phase of a sinusoid will be coherent and consistent with equation (3.34) as it drifts in $\omega$. (Recall that the phase of $X[n, \omega]$ is subject to discontinuities across
\( \omega \); it is these discontinuities we are attempting to steer clear of.) Empirical evidence indicates that inclusion of this phase extrapolation step provides significant improvements in processed speech quality.

As before, the synthesis stage simply computes an inverse Fourier transform of each synthesis frame \( Y_{sR_S, \omega} \), weights the result by the synthesis window \( f[n] \), and then uses an overlap-add method to combine the results of nearby frames. In the revised algorithm, since each frame \( Y_{sR_S, \omega} \) will be a set of impulses (zero everywhere except at the \( \omega_i \)), computing an inverse Fourier transform effectively computes the sum of the sinusoidal components present in that frame; use of a synthesis window effects the desired time weighting. As suggested in [18], we use a triangular window with a duration of \( 2R_S + 1 \) samples for synthesis. (Note that this too is a \( 1:R_S \) interpolating filter, although somewhat simpler than that used previously.)

The rest of the algorithm framework can be left intact and used as is. The code used to implement the changes described above can be found in appendix A.

Note that this revised algorithm does not attempt to correct any possible problems related to improper harmonic phase relationships between sinusoidal components of processed speech. To extend this algorithm further in that direction would make it very similar to the algorithm described by Quatieri and McAulay in [17].

### 4.2 Output Quality

Figure 4-1 shows the quasi-periodic test signal of figure 3-5 subjected to the same mild slowdown used to obtain figure 3-6, but using the revised algorithm instead of the Portnoff/Holtzman algorithm. Note that, as with the Portnoff/Holtzman algorithm, time-scale modification via the revised algorithm is not a waveform preserving operation. However, somewhat more than with Portnoff/Holtzman, local periodicity is maintained.

The revised TSM algorithm provides processed speech which, in general, sounds as good as or better than that provided by the Portnoff/Holtzman algorithm. However, the processed speech still sounds somewhat "hollow" and quite unnatural. Further, for the special case of integer rates of slowdown (see section 3.5.3), the Portnoff/Holtzman algorithm provides
Figure 4-1: Quasi-periodic test signal processed by revised algorithm, $\beta = \frac{29}{37}$ (200 ms duration) significantly higher output quality. The fact that the output quality of the two algorithms is similar leads us to believe that the primary degradation is not that corresponding to the phase discontinuities discussed in chapter three but instead that corresponding to incorrect harmonic phase relationships between sinusoidal components of the processed speech.

4.3 Varying the Frequency Resolution

As discussed in chapter one, the primary motivation behind developing an implementation of the Portnoff/Holtzman algorithm was a desire to understand how processed speech quality changes as frequency resolution is varied; it was hoped that such an understanding would provide insight into the observed poor performance of Galand’s algorithm [5].

To facilitate this investigation, the revised algorithm of chapter four was used to process several speech passages; each passage was processed using 512-point, 256-point, and 128-point DFTs for both $\beta = \frac{2}{3}$ (one and a half times slowdown) and $\beta = 2$ (two times speedup). (Recall that this thesis utilizes speech sampled at 8 kHz; thus these DFT sizes correspond to sub-band spacings of approximately 16 Hz, 31 Hz, and 63 Hz, respectively. In [5], Galand suggests a
spacing of 100 Hz.) Each of the passages resulting from use of 256-point and 128-point DFTs was compared to the corresponding passage resulting from processing with 512-point DFTs in an informal double-blind A-B comparison test.

When comparing sentences processed with 256-point DFTs against those processed with 512-point DFTs, little or no discernible difference was detected between the sentences. When comparing speech processed with 128-point DFTs to that processed with 512-point DFTs, the difference was usually readily apparent, for speech processed with 128-point DFTs was usually attenuated significantly. Further, the attenuation seemed to slowly vary somewhat randomly with time. Other than the attenuation, however, the sentences processed with 128-point DFTs did not sound too different from those processed with 512-point DFTs.

We saw in equation (3.29) that the STFT $X[n, \omega]$ can be viewed as a sum of shifted and weighted images of $H(\omega)$. We know that computing the corresponding DSTFT $X[n, k]$ is equivalent to computing samples in frequency of that STFT. Reducing the number of frequency samples from 512 to 128 corresponds to performing that sampling over a much coarser grid which in turn increases the likelihood of underestimating sinusoidal peak magnitudes. It would seem reasonable to assume this is a major cause of the observed attenuation when using the revised algorithm with 128-point DFTs.

The Portnoff/Holtzman algorithm implementation behaved similarly for 256-point DFTs; sentences processed with 512-point and 256-point DFTs were difficult to distinguish from one another. When using 128-point DFTs, however, the effects were more drastic: processed speech invariably took on an annoying reverberant quality.

Galand’s algorithm uses relatively low frequency resolution when analyzing input waveforms; the resulting processed speech possesses a reverberant quality. As the frequency resolution used with the Portnoff/Holtzman algorithm is lowered towards that suggested by Galand, it is comforting to note that similar degradation is introduced into the processed speech. By manipulating only sinusoidal peaks, the revised algorithm seems to avoid this difficulty.
Chapter 5

Conclusions

The bulk of this thesis (chapters two and three) is devoted to understanding the time-scale modification algorithms of Portnoff [13] and Holtzman [2]. The speech model and mathematical framework underlying these algorithms are presented in some detail. An implementation that embodies these ideas is described. Although speech processed by this system retains a high degree of intelligibility, it sounds somewhat “hollow” and unnatural and is easily distinguished from the original. Processed speech quality remains somewhat below that demonstrated by Portnoff and Holtzman.

Chapter four describes a revised algorithm which attempts to address one of the observed problems (phase discontinuity) of the Portnoff/Holtzman system. Speech processed with this algorithm sounds similar to that processed with the Portnoff/Holtzman system; possibly a little less scratchy. This similarity leads us to believe that the fundamental problem with output speech quality in the Portnoff/Holtzman system is not the phase discontinuity problem but the fact that proper harmonic phase relationships between sinusoidal components are not maintained.

An examination of processed speech quality from the Portnoff/Holtzman and revised algorithms for various frequency resolutions indicates that speech processed with 256-point DFTs sounds very nearly as good as that processed with 512-point DFTs. When the number of frequency samples is reduced to 128, however, processed speech quality degrades significantly. With such
coarse frequency sampling, speech processed with the Portnoff/Holtzman algorithm takes on a reverberant quality, similar to that observed in speech processed with the algorithm proposed by Galand in [5]. With the revised algorithm, however, degradation due to the use of 128-point DFTs seems to be primarily constrained to attenuation effects.

Work in the field of TSM which followed the works of Portnoff and Holtzman include that of Griffin and Lim [6] and that of Quatieri and McAulay [9, 17, 18]; each of these handles the difficulties surrounding waveform phase differently. In [6], the authors describe an iterative technique for estimating a suitable phase for an STFT given its magnitude. Although it requires a great deal of computation, this algorithm provides very high quality output. Quatieri and McAulay use a less brute-force approach; they develop a summed-sinusoid model of speech and propose an algorithm for TSM based on linearly time-scaling the parameters of that model. This leads to a system which, for the most part, maintains proper harmonic phase relationship between sinusoidal components and preserves waveform shape after time-scale modification. Quatieri and McAulay claim that their system provides very high quality, natural sounding processed speech. (This author has not had any opportunity to listen to any speech processed with their system.)
Appendix A

Simulation Codes

This appendix contains the simulation codes used in this thesis. These codes are written in the C programming language and have been compiled and run on several Unix systems, including a 4.3 BSD variant (Athena Workstation) and a System V variant (IBM AIX).
A.1 Portnoff/Holtzman’s Algorithm

Following is the simulation code for Portnoff/Holtzman’s algorithm used for the results presented in this thesis. A 321-point Hamming window is used for analysis; the synthesis filter is derived by applying Griffin and Lim’s LSEE transformation [6] to that synthesis window.

```c
/** *
  * file: holtz.c
  * auth: kirk johnson
  * date: 16 march 1989
  * what: implements portnoff/holtzman algorithm for tsm
  */

#include "std.h"

/** *
  * this implementation of holtzman’s version of portnoff’s TSM
  * algorithm assumes that both the analysis and synthesis windows have
  * an odd number of taps
  */

int dft_size;            /* DFT size (number of points) */
FILE *ins;               /* input stream */
FILE *outs;              /* output stream */

int ashift;              /* analysis shift */
int awin_size;           /* analysis window size */
int awin_half;           /* (awin_size-1)/2 */
double *awin;            /* analysis window coefs */
int aframeidx;           /* analysis frame index */

int sshift;              /* synthesis shift */
int swin_size;           /* synthesis window size */
int swin_half;           /* (swin_size-1)/2 */
double *swin;            /* synthesis window coefs */
int sframeidx;           /* synthesis frame index */

double beta;             /* tsm factor */
double *inbuf;           /* input buffer */
double *outbuf;          /* output buffer */
double *mag;             /* current frame’s magnitude */
double *phs;             /* current frame’s phase */
double *real;            /* DFT routine i/o buffers */
double *imag;
```

56
double *prvphs;  /* previous frame's phase */
double *outphs;  /* output phase for this frame */

main(argc, argv)
    int argc;
    char *argv[];
{
    /****
    the main body procedure. calls command_line() to decode the
    command line arguments, initialize() to set up windows, buffers,
    etc., and then runs the computation loop.
    ****/

    /* setup */
    command_line(argc, argv);
    initialize();

    /****
    computation loop:

    try to shift in another frame's worth of input samples. if a
    complete frame can not be shifted in, exit computation loop.
    otherwise run the analysis, phase modification and synthesis
    stages for the current frame, then output one frame's worth of
    samples to the output stream and shift the output buffer.
    ****/

    first_frame();
    while (shift_inbuf(ashift))
        other_frame();
}

command_line(argc, argv)
    int argc;
    char *argv[];
{
    /****
    procedure to decode the command line arguments, open the i/o
    streams, etc.
    ****/

    /* set default i/o streams */
    ins = stdin;
    outs = stdout;

    /* decode command line args */
    switch (argc)
    {
    case 6:
/* open input and output files */
ins = fopen(argv[4], "r");
if (ins == NULL)
    fatal("holtz", "problems opening input file");
outs = fopen(argv[5], "w");
if (outs == NULL)
    fatal("holtz", "problems opening output file");

/* fall through to next case */
case 4:
    /* get dft size, shifts */
    sscanf(argv[1], "%d", &dft_size);
    sscanf(argv[2], "%d", &ashift);
    sscanf(argv[3], "%d", &ashift);
    break;

default:
    /* oops */
    fatal("usage", "holtz <dftsize> <ashift> <sshift> [<infile> <outfile>]"");
    break;
}
}

initialize()
{
    /****
    calls procedures to create analysis and synthesis windows.
    allocates space for and initializes assorted buffers. assorted
    variable initialization.
    ****/

    int i;

    setup_analysis_window();
    setup_synthesis_window();

    /* allocate buffers */
inbuf = (double *) grab(sizeof(double) * awin_size);
outbuf = (double *) grab(sizeof(double) * swin_size);
mag = (double *) grab(sizeof(double) * dft_size);
phs = (double *) grab(sizeof(double) * dft_size);
real = (double *) grab(sizeof(double) * dft_size);
imag = (double *) grab(sizeof(double) * dft_size);
prvphs = (double *) grab(sizeof(double) * dft_size);
outphs = (double *) grab(sizeof(double) * dft_size);

    /* zero fill buffers */
    for (i=0; i<awin_size; i++)
inbuf[i] = 0;

for (i=0; i<swin_size; i++)
    outbuf[i] = 0;

for (i=0; i<dft_size; i++)
    prvphs[i] = 0;

for (i=0; i<dft_size; i++)
    outphs[i] = 0;

aframeidx = 0;
sframeidx = 0;

beta = (double) ashift / sshift;
}

first_frame()
{
    int i;

    /* load first frame's worth of input data */
    if (!shift_inbuf(awin_size))
        fatal("holtz", "problems loading first frame");

    analysis();

    for (i=0; i<dft_size; i++)
    {
        prvphs[i] = phs[i];
        outphs[i] = phs[i];
    }

    synthesis();
    shift_outbuf(sshift);
}

other_frame()
{
    analysis();
    phase_modification();
    synthesis();
    shift_outbuf(sshift);
}

setup_analysis_window()
{
    /****
installs values for awin_size and awin_half. allocates space for
and generates analysis window (awin[])
****/

int i;

awin_size = 201;          /* must be an odd number */
awin_half = (awin_size-1) / 2;

awin = (double *) grab(sizeof(double) * awin_size);

/* use a Hamming window */
for (i=0; i<awin_size; i++)
    awin[i] = 0.54 - 0.46 * cos(i*TWOPI/(awin_size-1));
}

setup_synthesis_window()
{
    /***
    installs values for swin_size and swin_half. allocates space for
and generates synthesis window (swin[])
    *****/

int i, j;
double sum;

swin_half = awin_half;
swin_size = swin_half*2 + 1;

swin = (double *) grab(sizeof(double) * swin_size);

/* prototype filter is a Hamming window with the same duration as */
/* the analysis filter */

for (i=0; i<swin_size; i++)
    swin[i] = 0.54 - 0.46 * cos(i*TWOPI/(swin_size-1));

/* transform the synthesis filter according to griffin and lim's */
/* LSEE-MSTFT method (IEEE ASSP, april 1984, p. 237) */

for (i=0; i<sshift; i++)
{
    sum = 0;
    for (j=i; j<swin_size; j+=sshift)
        sum += swin[j]*swin[j];

    for (j=i; j<swin_size; j+=sshift)
        swin[j] /= sum;
}

}
shift_inbuf(cnt)
    int cnt;
{
    /*
    updates inbuf[] by shifting out cnt old samples and shifting in
    cnt new ones. returns 1 if successful; 0 otherwise.
    */
    int i;
    int nread;
    int rslt;

    /* shift old data */
    for (i=cnt; i<awin_size; i++)
        inbuf[i-cnt] = inbuf[i];

    /* load new data */
    nread = fread(inbuf+awin_size-cnt, sizeof(double), cnt, ins);

    if (nread == cnt)
        rslt = 1;
    else
        rslt = 0;

    return rslt;
}

analysis()
{
    /*
    runs one frame of DSTFT analysis on current contents of inbuf[];
    leaves result in mag[] and phs[]. increment aframeidx.
    */
    int i, j;

    /* zero real[] and imag[] */
    for (i=0; i<dft_size; i++)
    {
        real[i] = 0;
        imag[i] = 0;
    }

    /*
    when copying data from inbuf[] to real[], a circular right shift
    of (aframeidx*ashift - awin_half) should be applied to correct for
    the sliding time reference of inbuf[] and the duration of the
    analysis window
    */
****/

j = (aframeidx*ashift - awin_half) % dft_size;
if (j < 0) j += dft_size;

for (i=0; i<awin_size; i++)
{
    real[j++] += inbuf[i] * awin[i];
    if (j >= dft_size) j -= dft_size;
}

/* run the transform; convert result to polar form */
dft(real, imag, dft_size);
for (i=0; i<dft_size; i++)
{
    mag[i] = hypot(real[i], imag[i]);
    if (mag[i] > 0)
        phs[i] = atan2(imag[i], real[i]);
    else
        phs[i] = 0;
}
aframeidx += 1;
}

phase_modification()
{
    /*****
    run phase modification stage on contents of phs[] (also uses
    prvphs[] and outphs[])
    ******/
    int i;
    double tmp;

    for (i=0; i<dft_size; i++)
    {
        /* first backward difference */
        tmp = phs[i];
        phs[i] -= prvphs[i];
        prvphs[i] = tmp;

        /* remove jumps of TWOPi */
        if (phs[i] < -PI)
            phs[i] += TWOPi;
        else if (phs[i] > PI)
            phs[i] -= TWOPi;
    }

62
/* compute the output phase */
outphs[i] += phs[i] / beta;
    phs[i]    = outphs[i];
}
}

synthesis()
{
    /****
    runs one frame of DSTFT synthesis on contents of mag[] and phs[];
    adds result into outbuf[]. increment sframeidx.
    ****/
    int i, j;

    /* convert to rectangular form; run inverse transform */
    for (i=0; i<dft_size; i++)
    {
        real[i] = mag[i] * cos(phs[i]);
        imag[i] = mag[i] * sin(phs[i]);
    }
    idft(real, imag, dft_size);

    /****
    when copying data from real[] to outbuf[], a circular left shift
    of (sframeidx*sshift - swin_half) should be applied to correct for
    the sliding time reference of outbuf[] and the duration of the
    synthesis window
    ****/
    j = (sframeidx*sshift - swin_half) % dft_size;
    if (j < 0) j += dft_size;

    for (i=0; i<swin_size; i++)
    {
        outbuf[i] += real[j++] * swin[i];

        if (j >= dft_size) j -= dft_size;
    }
    sframeidx += 1;
}

shift_outbuf(cnt)
    int cnt;
{
    /****
    shift and output cnt samples from outbuf[]; zero fill new space
    ****/
    63
int i;

    /* output cnt samples */
    fwrite(outbuf, sizeof(double), cnt, outs);

    /* shift old data */
    for (i=cnt; i<swin_size; i++)
        outbuf[i-cnt] = outbuf[i];

    /* zero fill new space */
    for (i=swin_size-cnt; i<swin_size; i++)
        outbuf[i] = 0;

}
A.2 Revised Algorithm

The simulation code for the revised algorithm discussed in chapter four is, for the most part, exactly the same as the code used to simulate the Portnoff/Holtzman algorithm shown in the previous section. Only two procedures are changed:

- `phase_modification()`: This procedure is changed to implement the new phase and amplitude modification stage described in chapter four.

- `setup_synthesis_filter()`: Instead of generating a synthesis filter derived from the analysis filter via the LSEE-MSTFT transformation, this procedure generates a triangular window with a duration which is twice the synthesis interpolation rate.

The new code for these procedures is shown below.

```c
phase_modification()
{
    /****
    * run revised phase modification stage on contents of phs[] (also
    * uses prvpchs[] and outphs[])
    ****/

    int i, done;
    double tmp;
    double slope;
    char *mark;

    for (i=0; i<dft_size; i++)
    {
        /* first backward difference */
        tmp = phs[i];
        phs[i] -= prvpchs[i];
        prvpchs[i] = tmp;

        /* remove jumps of TW0PI */
        if (phs[i] < -PI)
            phs[i] += TW0PI;
        else if (phs[i] > PI)
            phs[i] -= TW0PI;
    }

    /* allocate and initialize space for peak marking */
    mark = (char *) grab(sizeof(char) * dft_size);
```
for (i=0; i<dft_size; i++)
    mark[i] = 0;

/* peak check the low endpoint */
if ((mag[0] >= mag[dft_size-1]) &&
    (mag[0] >= mag[1]))
    mark[0] = 1;

/* peak check the "body" */
for (i=1; i<(dft_size-1); i++)
    if ((mag[i] >= mag[i-1]) &&
        (mag[i] >= mag[i+1]))
        mark[i] = 1;

/* peak check the high endpoint */
if ((mag[dft_size-1] >= mag[dft_size-2]) &&
    (mag[dft_size-1] >= mag[0]))
    mark[dft_size-1] = 1;

/* set mag[] at non-peaks to zero */
for (i=0; i<dft_size; i++)
    if (!mark[i])
        mag[i] = 0.0;

/* compute outphs[] at peaks as normal ... */
for (i=0; i<dft_size; i++)
    if (mark[i])
        outphs[i] += phs[i]/beta;

/* interpolate the phase between peaks */
slope = - (TWOPI/dft_size)*(sframeidx*sshift);
for (i=1; i<(dft_size-1); i++)
    if (!mark[i])
    {
        /* if both adjacent bands are marked ... */
        if (mark[i-1] && mark[i+1])
        {
            /* flip a coin for which to estimate from */
            if (rand()%2)
                outphs[i] = outphs[i-1] + slope;
            else
                outphs[i] = outphs[i+1] - slope;

            /* mark the band */
            mark[i] = 1;
        }
/* if the band below is marked ... */
else if (mark[i-1])
{
    outphs[i] = outphs[i-1] + slope;

    /* mark the band */
    mark[i] = 1;
}

/* if the band above is marked ... */
else if (mark[i+1])
{
    outphs[i] = outphs[i+1] - slope;

    /* mark the band */
    mark[i] = 1;
}

/* see if we're done */
done = 1;
for (i=1; i<(dft_size-1); i++)
    if (!mark[i])
    {
        done = 0;
        break;
    }
}
while (!done);

/* determine phase for endpoints ... */
outphs[0] = outphs[1] - slope;
outphs[dft_size-1] = outphs[dft_size-2] + slope;

/* copy outphs[] to phs[] */
for (i=0; i<dft_size; i++)
    phs[i] = outphs[i];

/* deallocate peak marking space */
ungrab(mark);
}

setup_synthesis_window()
{
    /****
     * installs values for swin_size and swin_half. allocates space for
     * and generates synthesis window (swin[])
     ****/

    int i;
    double fix;
swin_half = sshift;
swin_size = swin_half*2 + 1;

swin = (double *) grab(sizeof(double) * swin_size);

/* generate triangular window */
for (i=0; i<sshift; i++)
{
    swin[i] = (double) i / sshift;
    swin[swin_size-1-i] = swin[i];
}
swin[i] = 1.0;

/* compute compensation factor (to correct for */
/* magnitude problems introduced by peak picking) */
fix = 0;
for (i=0; i<swin_size; i++)
    fix += swin[i];
fix = dft_size / fix;

/* apply compensation to synthesis window */
for (i=0; i<swin_size; i++)
    swin[i] *= fix;
A.3 DFT Routine

The following code implements a multi-radix (2, 3, 5, and 7) decimation-in-time FFT algorithm (see [11]); radix-two components of the decimation-in-time decomposition are computed "in place" for increased performance.

/****
   file: sfft.c
   auth: kirk johnson
   date: 8 october 1988
   what: multi-radix fast dft routine
   ****/

#include "std.h"

static int npts = 0;
static double *wrl;
static double *wim;

dft(rl, im, sz)
   double *rl1;
   double *im1;
   int sz;
{
   setup_dft_coefs(sz);
   sfft(rl1, im1, sz, 1);
}

idft(rl, im, sz)
   double *rl1;
   double *im1;
   int sz;
{
   int i, j;
   double tmp;

   /* time reversal */
   i = 1;
   j = sz - 1;
   while (j > i)
   {
      tmp = rl1[i];
      rl1[i] = rl1[j];
      rl1[j] = tmp;

      tmp = im[i];

   }
im[i] = im[j];
im[j] = tmp;

i += 1;
j -= 1;
}

setup_dft_coefs(sz);
srcft(rl, im, sz, i);

/* scale rl[] and im[] by 1/sz */
tmp = 1.0 / sz;
for (i=0; i<sz; i++)
{
    rl[i] *= tmp;
im[i] *= tmp;
}

setup_dft_coefs(sz)

    int sz;
{
    int i;

    /* see if we're already set up */
    if (sz == npts) return;

    /* check the decomposition */
    radix_check(sz);

    /* de/allocate table space */
    if (npts != 0)
    {
        ungrab(wrl);
        ungrab(wim);
    }
    wrl = (double *) grab(sizeof(double) * sz);
    wim = (double *) grab(sizeof(double) * sz);

    /* fill table */
    for (i=0; i<sz; i++)
    {
        wrl[i] = cos(i*TWOPi/sz);
wim[i] = -sin(i*TWOPi/sz);
    }

    /* update npts */
npts = sz;
}
radix_check(sz)
    int sz;
{
    while (1)
        if (sz == 1) break;
    else if (sz%2 == 0) sz /= 2;
    else if (sz%3 == 0) sz /= 3;
    else if (sz%5 == 0) sz /= 5;
    else if (sz%7 == 0) sz /= 7;
    else
        fatal("srfft", "DFT size not radix (2,3,5,7)" );
}

srfft(rl, im, sz, df)
    double *rl;
    double *im;
    int sz;
    int df;
{
    int i, j;
    int radix, nsz;
    int idx0, idx1;
    double **nrl;
    double **nim;

    /* determine radix for this stage */
    if (sz == 1) return;
    else if (sz%7 == 0) radix = 7;
    else if (sz%5 == 0) radix = 5;
    else if (sz%3 == 0) radix = 3;
    else
    { /* sz must be purely radix 2 */
        radix = 2;
        r2fft(rl, im, sz, df);
        return;
    }

    /* allocate work space */
    nsz = sz / radix;
    nrl = (double **) grab(sizeof(double *) * radix);
    nim = (double **) grab(sizeof(double *) * radix);
    for (i=0; i<radix; i++)
    {
        nrl[i] = (double *) grab(sizeof(double) * nsz);
        nim[i] = (double *) grab(sizeof(double) * nsz);
    }

    /* perform decimation */
for (i=0; i<radix; i++)
    for (j=0; j<nsz; j++)
    {
        nrl[i][j] = rl[j+radix*i];
        nim[i][j] = im[j+radix*i];
    }

/* recurse */
for (i=0; i<radix; i++)
    srfst(nrl[i], nim[i], nsz, df*radix);

/* finish computation */
for (i=0; i<sz; i++)
{
    idx0 = i%nsz;
    rl[i] = nrl[0][idx0];
    im[i] = nim[0][idx0];

    for (j=1; j<radix; j++)
    {
        idx1 = (i*j*df)%npts;
        rl[i] += nrl[j][idx0] * wrl[idx1];
        rl[i] -= nim[j][idx0] * wim[idx1];
        im[i] += nrl[j][idx0] * wim[idx1];
        im[i] += nim[j][idx0] * wrl[idx1];
    }
}

/* clean up work space */
for (i=0; i<radix; i++)
{
    ungrab(nrl[i]);
    ungrab(nim[i]);
}
ungrab(nrl);
ungrab(nim);

r2fft(rl, im, sz, df)
double *rl;
    double *im;
    int sz;
    int df;
{
    int i, j, k;
    int step, wstep;
    int uidx, vidx, widx;
    double rtmp, itmp;

    /* perform bit-reversed sort on the data */
j = 0;
for (i=0; i<(sz-1); i++)
{
    if (i > j)
    {
        /* swap i-th and j-th elements */
        rtmp = rl[i];
        rl[i] = rl[j];
        rl[j] = rtmp;
        itmp = im[i];
        im[i] = im[j];
        im[j] = itmp;
    }

    /* generate next bit-reversed index */
    k = sz >> 1;
    while (j >= k)
    {
        j -= k;
        k >>= 1;
    }
    j += k;
}

/* run the butterflies */
i = 1;
step = 2;
wstep = (df*sz) >> 1;
while (i)
{
    i *= 2;
    if (i > sz) break;

    widx = 0;
    for (j=0; j<(step>>1); j++)
    {
        for (k=0; k<sz; k+=step)
        {
            /* run one butterfly */
            uidx = j + k;
            vidx = j + k + (step>>1);

            rtmp = rl[vidx] * wrl[widx];
            rtmp -= im[vidx] * wim[widx];
            itmp = rl[vidx] * wim[widx];
            itmp += im[vidx] * wrl[widx];

            rl[vidx] = rl[uidx] - rtmp;
            im[vidx] = im[uidx] - itmp;
            rl[uidx] += rtmp;
        }
    }
}
im[uidx] += tmp;
}
        widx += wstp;
}
wstp >>= 1;
step <<= 1;
}
Bibliography


