To the graduate Council:

I am submitting herewith a dissertation written by Miljko Bobrek entitled "POLYPHONIC MUSIC SEGMENTATION USING WAVELET BASED TREE-STRUCTURED FILTER BANKS WITH IMPROVED TIME-FREQUENCY RESOLUTION." I have examined the final copy of this dissertation for form and content and recommended that it be accepted in partial fulfillment of the requirements for the degree of Doctor of Philosophy with a major in Electrical Engineering.

Daniel B. Koch

Daniel B. Koch, Major Professor

We have read this dissertation
and recommended its acceptance:

M. J. Bo

M. A. Abidi

Accepted for the Council:

Lev Minkel

Associate Vice Chancellor and
Dean of The Graduate School

Polyphonic Music Segmentation using Wavelet Based Tree-Structured Filter Banks with Improved Time-Frequency Resolution

A Dissertation
Presented for the
Doctor of Philosophy
Degree
The University of Tennessee, Knoxville

Miljko Bobrek
August 1996

Mojoj supruzi Sonji i sinu Aleksandru.

ACKNOWLEDGMENTS

I would like to thank my advisor Dr. Daniel B. Koch for his valuable help throughout my career as a research assistant, and for his suggestions and guidance in the preparation of this dissertation.

I am also grateful to Dr. Mongi A. Abidi, Dr. Mark E. Boling, and Dr. Michael J. Roberts for serving on my dissertation committee.

Many thanks are due to "Philips Laboratories" at Briarcliff Manor, NY, sponsor of this work, and to Dr. C.A.A.J. Greebe who helped in focusing and shaping the research.

I thank Lesia Shulha who plays the real-life piano recording used in the research.

Finally, I am very grateful to my wife Sonja and son Alex for their constant support and patience during these years.

ABSTRACT

The logarithmic structure of the music signal spectrum requires a constant-Q transform for front-end signal processing in polyphonic music segmentation. The Short-Time Fourier Transform (STFT) offers a linear segmentation of the frequency space, i.e., a variable-Q analysis. On the other hand, the Dyadic Wavelet Transform performs a constant-Q analysis, but with a Q-factor that is much below the required level. Tree-structured filter banks that are related to wavelet packets offer a nearly-constant-Q analysis with an arbitrarily high Q-factor that can be increased using more stages in the tree structure.

To improve selectivity in tree-structured filter banks, a new family of Quadrature Mirror Filters (QMFs) with a narrow transition region and a low reconstruction error was designed and implemented. It is shown that these QMFs significantly reduce the frequency leakage in tree-structured filter banks that occurs when the QMF or perfect reconstruction filter pairs with a wide transition region are used. The total reconstruction error in the tree-structured filter banks when the new QMFs are used is smaller than the round-off error which occurs during the implementation.

For music signal segmentation, an 11-stage tree structure was designed and implemented. The tree-structured filter bank has sufficient frequency resolution over a wide range of frequencies, while the time resolution at high frequencies satisfies the minimum time resolution for the human ear.

The tree-structured filter banks were implemented in two applications. The first one is the transcription of polyphonic piano music where the filter banks were used as the front-end signal processing. In the application, synthesized as well as a real-life polyphonic piano recordings were successfully coded into MIDI streams. In the second application, both the analysis and the synthesis parts of the tree-structure were

implemented in a specially designed editor that was used for voice separation in polyphonic piano music as well as for signal denoising.

TABLE OF CONTENTS

CHAPTER **PAGE**

CHAPTER PAGE

CHAPTER PAGE

LIST OF TABLES

TABLE **PAGE**

LIST OF FIGURES

FIGURE **PAGE**

FIGURE **PAGE**

LIST OF SYMBOLS AND ABBREVIATIONS

$\mathcal{Z}$	set of integer numbers
$\mathcal{R}$	set of real numbers
$\mathcal{N}$	set of natural numbers
$L_2(\mathcal{R})$	space of square-integrable functions
$\|f(t)\|$	norm of $f(t)$ defined as $\sqrt{\int_{t\in\mathcal{R}} \|f(t)\|^2 dt}$
$\langle f(t), g(t)\rangle$	standard inner product for complex-valued functions defined as $\int_{-\infty}^{\infty} f^*(t)g(t)dt$
$l_2(\mathcal{Z})$	space of square-summable sequences
F^{-1}	inverse Fourier Transform
$\lceil x \rceil$	the smallest integer greater than or equal to x
$\lfloor x \rfloor$	the largest integer smaller than or equal to x
$O(x)$	order of x
$\delta(n)$	Kronecker delta function defined as $\delta(n) = \begin{cases} 1, & n = 0 \\ 0, & n \neq 0 \end{cases}$
AM	Amplitude Modulation
CCRMA	Center for Computer Research in Music and Acoustic
CWT	Continuous Wavelet Transform
DC	Direct Current
DSP	Digital Signal Processing
DWT	Discrete Wavelet Transform
ECG	Electrocardiogram
ESPRIT	Estimation of Signal Parameters via Rational Invariance Techniques
FFT	Fast Fourier Transform
FIR	Finite Impulse Response

FM	Frequency Modulation
FT	Fourier Transform
IIR	Infinite Impulse Response
MSTFT	Multiresolution Short-Time Fourier Transform
MIDI	Musical Instruments Digital Interface
MUSICAM	Masking-Pattern Universal Integrated Coding and Multiplexing
QMF	Quadrature Mirror Filter
SMF	Standard MIDI file
STFT	Short-Time Fourier Transform
WT	Wavelet Transform

CHAPTER 1

INTRODUCTION

1.1 Polyphonic Music Transcription Objectives

Polyphonic music analysis, voice recognition, and ECG signal analysis are different cases of the same signal analysis problem: How does one transform a signal, well-localized in both time and frequency, to a space where important time and frequency features become easily separable? Since these signals are nonstationary, a time-varying signal processing technique must be applied. Furthermore, if real-time processing of such signals is required, the problem becomes a challenge in both the signal processing and computing optimization areas.

Music signal analysis can be divided into several areas according to the goals that need to be achieved by the analysis. The most important are music recording restoration and enhancement, music signal compression, music signal representation, music signal separation, and automatic music transcription. Music synthesis and music signal modeling, although very important parts of music signal processing, will not be closely considered in this work. Continuing, a short overview of different music signal analysis methods are given together with the most important articles related to these methods. Afterwards, polyphonic music transcription objectives are described in more detail. The term "transcription" in this work means the coding of music signals into the MIDI stream, while "polyphonic music" denotes a single instrument music piece where more than one note can occur simultaneously. For example, the oboe or clarinet are monophonic instruments while a piano is a polyphonic one.

The most common problem in music signal analysis is music signal enhancement and restoration. Old music recordings are usually degraded by different kind of disturbances such as white noise, scratches, missing data, and impulsive disturbances. So far, many

different signal processing techniques have been used to remove noise or to restore missing data in an old recording. Most of them take advantage of the quasi–periodic nature of the music signal. Algorithms for white noise removal are based on spectral subtraction [1, 2], or different methods that use least mean–square adaptive filtering [3]. It is shown in [1] that scratches can be effectively removed by combining matched filtering and linear adaptive noise subtraction. Recently, some new techniques were used for restoration - so called Short–Time Spectral Attenuation [4], or the Ephraim–Malah Noise Suppresser [5]. In [6] music noise removal is based on specially designed wavelet packets while in [7] short–term spectral analysis was used to remove background noise from music signals. For restoration of missing digital audio data in [8] a spectral extrapolation method was used.

Music signal coding and compression is an important task in the design of audio systems. The main goal is to achieve transparent coding of hi–fidelity music signals at the lowest possible bit rates. These signals are originally sampled at 44.1 kHz and quantized with 16 bits/sample resulting in a high bit rate of 705 Kbits/s. There exist two basic approaches for bit rate reduction. These include several subband coding techniques [9, 10, 11 ,12] and transform coding methods [13, 14]. Recently, the wavelet transform has been used for low-bit-rate transparent audio compression [15]. These methods lead to the development of different standards for low rate, high quality coding of audio signals. The most well–known coding algorithm is MUSICAM (Masking–Pattern Universal Integrated Coding and Multiplexing) [16] that uses a 32–band uniform filter bank designed by modulation of a 512–tap prototype lowpass filter. This algorithm compresses audio signals down to 128 Kbits/s. Two other algorithms [17, 18], reduce the bit rate to 128 Kbits/s as well. However, some of the methods [15] reduce the bit rate down to around 50 Kbits/s with almost perceptually transparent coding. Furthermore, some recent studies [19] show that well–known linear prediction techniques used in speech coding, are also viable for music signal coding.

Music signal representation is predominantly used to transform music signals into a space where all necessary features of the music signal are easier to recognize. Besides the features that can be treated mathematically, including rhythm and harmony, there are many nonmathematical features such as tension, expectancy, and emotions [20]. This means that the representation of music as an art form is not always possible using standard signal processing techniques. More often, music signal representation is performed by different forms of high–resolution spectral analysis. In [21] the matrix pencil method for spectrum analysis of music signals was used. This method relies on the hypothesis that the signal is made up of a sum of exponentially decaying sinusoids, and is used for estimation of frequencies, damping factors, amplitudes, and phases of the signal. There exist other high–resolution methods developed in past 10 years such as the Prony method [22], the ESPRIT method [23], and the Kumaresan–Tufs method [24]. The Discrete Short–Time Fourier Transform was used in [25] for analyzing transitions between notes. In [26] the author transforms music signals into a deterministic and a stochastic component making it possible for a synthesized sound to attain all perceptually significant characteristics of the original sound. Recently, music signals were represented using different forms of time–frequency representations such as the Gabor Transform [27] and the Wavelet Transform [28]. In [29, 30] a new class of so-called harmonic wavelets is suggested. In this work the harmonic wavelets were derived by an orthogonal, nonoverlapping split of frequency space using rectangular windows. By an easy choice of the wavelet parameters, one can split the frequency space into bands that exactly correspond to the note frequencies. However, unlimited support of these wavelets and their slow decay result in a poor time localization.

Music signal separation or voice separation, as it is usually called, is quite a challenging task, especially if a specific voice needs to be extracted from a multi-instrument music piece. Even though voice separation is closely related to music transcription, it goes a step further trying to separate two instruments playing

simultaneously based on their differences in attack and frequency structure. An early work [31] indicated that imposing a common frequency jitter pattern on a set of partials can help in the recognition of a singing voice. In [32] a voice separation algorithm is based on simultaneously tracking frequency jitter patterns for more than one source. Source verification as proposed in [33], relies on three most important discriminatory features, spectral evolution of transients, modulation behavior, and resonance structure. More recently, source separation algorithms try to use time-frequency images to locate patterns that belong to each event [34]; so called Frequency-Warped Signal Processing to separate the AM and FM contributions to the signal bandwidth [35].

Polyphonic music transcription in the general sense means coding of music signals into musical scores or into a MIDI stream, a more precise way. While musical scores give a rough estimate of amplitudes, on–set times, and duration of notes, MIDI allows much finer coding of these parameters. Using musical scores a music piece can be performed in many different styles and nuances. However, a MIDI stream describes a music piece in an exact way allowing one to reproduce music on an electronic instrument with all of the nuances introduced by the original performer. Obviously, fidelity of the reproduced music strongly depends on electronic MIDI instruments and how closely they simulate voices of a natural instrument. As an illustration of the complexity of polyphonic music transcription, a short transcription scenario is quoted: "replay the piano part, this time transposed down a third and with the timbre of a trumpet, while preserving all stylistic features of the original." If the above mentioned piano voice is incorporated into an old symphonic music recording, the task becomes a real challenge.

What makes polyphonic music transcription such a difficult task is the complexity of music signals in both time and frequency. Even though a particular monophonic signal, e.g., a single piano note, is a pseudo–periodic signal with distinguishable spectral components, its amplitude–frequency relationship shows a strong nonlinearity. Moreover, in polyphonic music, a new kind of nonlinearity occurs when two single notes

interfere. Also, the great number of harmonics and large dynamic range of the signal make it difficult to separate amplitude, pitch, on–set time, and duration of a particular note due to possible overlapping in both time and frequency.

More generally, a music transcription procedure as it was used in most of the recent work, can be divided into three steps. As in every signal classification problem, the first step, often called the front-end signal processing, is to transform an original recording to a space where the features of interest are easier to separate. It is obvious that the selection of the transform method has the most crucial role in successful feature extraction and classification that follow in the second step. Most often, one of the forms of the STFT is used to transform polyphonic music signals to a different feature space. In the second step of the procedure the features of interest in the new time–frequency space are extracted and classified. These features in the case of piano music are amplitude, pitch, on–set time, and duration. At the end, these features are used for sheet music or MIDI stream generation.

1.2 Previous Work

Polyphonic music transcription has been in the research focus for at least 20 years. This overview of the most important articles published during these years is divided into several groups. The first one presents the early work in the area of music signal transcription. In his work in this area Moorer [36] used a directed bank of heterodyne filters to track note partials. This method however, requires the key signature of the piece to be determined beforehand in order to direct the filter bank. Furthermore, there are some other limitations concerning the source material that require it to be two-note polyphonic with additional time and frequency limitations. In a later work Moorer [37] used bandpass filtering to extract individual harmonics, but with similar restrictions on the input signal.

Fundamental frequency tracking of monophonic and low complexity polyphonic

music has been extensively studied over the years. However, most of the fundamental frequency tracking methods listed below concentrate solely on note frequency estimation, and do not consider front-end signal processing nor the estimation of other note parameters such as attack, amplitude, or duration. In their work Piszczalski et al. [38] use the Short-Time Fourier Transform to generate three-dimensional images of monophonic flute music and then the computer-generated notation of the piece. Later in a separate paper Piszczalski and Galler [39] described a new method for musical pitch prediction based on estimation of component frequency ratios.

To the same group of work on pitch tracking belong several approaches presented in the last few years. Musical frequency tracking using conventional and "narrowed" autocorrelation was proposed by Brown and Zhang [40]. The proposed algorithm calculates autocorrelation of the input signal which, based on the harmonic structure of the signal, gives a large peak at the fundamental frequency. A disadvantage of the method is that it suffers from poor time resolution. As a result of continuous research on frequency tracking Brown [41, 42] proposed two new methods based on phase changes of the Fourier transform and on a pattern recognition method. Maher and Beauchamp [43] proposed a new method for fundamental frequency estimation using so called two-way mismatch procedure. The method was applied to monophonic as well as to simple duet voices. While relatively effective for monophonic cases, the method for duets suffers from poor frequency resolution at lower frequencies. Furthermore, the method seems to be time consuming and not applicable in the case of a rapid change of the fundamental frequency. Doval and Rodet [44] presented a method for estimation of the fundamental frequency based on a probabilistic model of pseudo-periodic signals. Their method can operate on a large interval of fundamental frequencies (50 to 4000 Hz) and on different sound signals (speech and music). One of the most recent works by Dorken and Nawab [45] improves the algorithm proposed by Piszczalski and Galler [39] in a situation when the signal is corrupted by interference components. Real-time pitch

recognition of the human voice proposed by Khun [46] may also be placed in the group of monophonic music transcription. The method uses a bank of digital filters followed by downsamplers. It works in real time with an accuracy of 1–2 percent, a dynamic range of 20–40 dB and a pitch sample rate of 12–24 Hz.

Music tempo tracking can also be considered as a separate part of polyphonic music transcription. Its main goal is to automatically detect tempo or dynamics of a live music performance allowing a computer to take on the role of an accompanist. Several systems described in [47, 48, 49, 50] detect and respond to the bits in musical performances in real time.

The next group contains work done on the front-end signal processing. The front-end transformation of the music signal is certainly the most important part of polyphonic music transcription. It transforms a music signal to a new feature space allowing easier feature extraction. In the work presented so far, for the front-end signal processing, different forms of the STFT were used. Since music signals are well localized in both time and frequency, the STFT appears to be an obvious way to transform an one-dimensional signal to a two-dimensional time-frequency space. However, the linear frequency scale employed in the STFT performs variable-Q frequency segmentation of harmonic music signals that need constant-Q segmentation. Therefore, most of the proposed front-end signal processing algorithms use some kind of modified STFT. Dolson [51] proposed two alternative ways to use the STFT. In the first one called Spectral Permutation one can simply reorder the frequency bins in each frame of the short-time Fourier transform. The other, so called Time-Varying Filtering, uses a simple gain-scaling of each bin that has an effect of time-varying filtering. Various schemes for the constant-Q spectral analysis not related to music signals have been published [52, 53]. Researchers at the Center for Computer Research in Music and Acoustics (CCRMA) at Stanford have used so-called Bounded-Q Transform [54]. In the Bounded-Q method, the analyzing widow size is selected so as to only resolve partials in the upper octave of the signal. The signal is

repeatedly transformed and down-sampled by an octave. Each octave decimation produces a set of bins with doubling resolution. Brown [55, 56] proposed a constant-Q modification of the STFT. It is a constant-Q transform equivalent to a 1/24-octave filter bank. Kronland-Martinet [57] has employed a relatively new transform, the Wavelet Transform, for music signal analysis. This is a constant-Q transform, but with Q equal to 1.5 which is much below the required selectivity for note frequency estimation. The music transcription method proposed by Pearson [58] and Wilson et. al. [59] uses a new method called the Multiresolution Fourier Transform. In this method many standard STFTs with different time and frequency resolutions are compared and features of interest are extracted. The method is computationally very extensive since several two-dimensional time-frequency transformations are performed on a one-dimensional time signal.

So far in this section, work on different parts of a music transcription system has been presented. However, not much has been done on an overall system for polyphonic music transcription. It is mostly caused by the fact that all the existing methods mentioned above need to be improved and automated to the extent where they can be reliably combined in an automated polyphonic music transcription system. It seems that most of the work on the definition of a complete polyphonic transcription system has been done at the CCRMA at Stanford. In their work Chris Chafe, et. al. [60] presented a complete transcription strategy that employs not only different DSP techniques but also takes musical contents of a piece into consideration. The whole algorithm is divided into five separate technique groups. The first one represents the front-end signal processing where a high resolution in the frequency domain is required to distinguish voices in a polyphonic texture. Then follows the event detection, a time domain technique for identifying points of interest in the signal. In the note modeling, detected events are examined. Chords are broken into notes and separate notes are tracked in time. Early context generation that follows uses the timing events to built a metrical grid. At the end,

in the recursive sharpening, tracked notes are eliminated from the original signal to allow detection of hidden events.

According to the work presented so far on polyphonic music transcription, the following conclusions about the current research status as well as future research directions can be reached.

- Several different front-end processing techniques for music signal analysis have been used so far. They are mostly oriented toward an adaptation of the STFT. However, the connection between wavelets and filter banks presented recently made filter banks good candidates for music signal analysis. Using filter banks one can adjust the frequency resolution during signal analysis so that both the lowest and the highest frequencies of interest are analyzed with same resolution. Furthermore, it is possible to design filter banks that, besides having good selectivity, are orthogonal or almost orthogonal. The orthogonality of the filter banks makes it possible to extract a musical context from a polyphonic music performance.
- Fundamental frequency tracking is evidently the most active research area in polyphonic music transcription. There are many very effective techniques developed for real time frequency estimation. However, they deal with monophonic and low complexity duet signals only. Therefore, there is still a lot of research interest in fundamental frequency tracking of polyphonic music.
- Music tempo tracking relies on on-set time detection, also called event detection. Most of the proposed algorithms use some kind of energy slope detection in the time domain. However, in case of percussive instruments, frequency domain energy distribution seems to be more promising in detection of noisy-like attack signals.
- Sheet music has been considered as the output of music transcription algorithms in most of the presented work. Recently, however, the MIDI stream replaces

sheet music as the algorithm output since it gives a much more precise description of the necessary features. Sheet music is then obtainable from the MIDI stream in a straightforward transcription procedure.

1.3 Overview and Contribution of the Dissertation

This dissertation emphasizes the implementation and improvement of recent DSP techniques that can eventually lead to the automated transcription of polyphonic music made by a percussive instrument such as a piano. The DSP techniques that are the most suitable for the analysis of signals in both time and frequency are different types of the STFT. In this dissertation, Multirate Filter Banks that are closely related to the Wavelet Transform are used for the front-end music signal processing. As a relatively young DSP technique, Multirate Filtering has some advantages over other DSP techniques especially when good frequency resolution is needed over a wide range of signal frequencies. These advantages are exploited in the dissertation by developing a new analysis method that allows multirate filters to be used for pseudo–periodic signal analysis and transcription of polyphonic music. The new method relies on specially designed tree-structured filter banks, and as a general signal processing tool can be used in other areas of audio signal processing such as audio signal compression or enhancement. Similar tree-structured filter banks have been used before in speech and music signal coding and enhancement [15]. However, if these filter banks need to be used in music signal segmentation, they have to have good resolution in both time and frequency. Therefore, in this dissertation, a new class of filter banks with good selectivity is presented and implemented in a tree-structured scheme. The proposed tree-algorithm is compared with the existing front-end signal processing algorithms according to its effectiveness as well as its implementation complexity. Finally, the new algorithm is implemented in two different applications. The first one is a transcription system for polyphonic piano music transcription. In this system, besides the tree-structured filter banks, some other original techniques for attack

location and pitch detection are proposed and implemented. The detected features are transcribed into MIDI streams and then played back on a MIDI sound module. In the second application, a special music signal editor is designed and later used in voice separation of a polyphonic piano piece.

This dissertation is organized as follows:

In Chapter 2, the continuous-time STFT and the CWT as well as their frames are compared with respect to their time-frequency resolutions and their orthogonality conditions. Chapter 3 continues with the comparison for the discrete-time STFT and WT, and describes the connection between the WT and filter banks. At the end of the chapter, tree-structured filter banks and wavelet packets are presented. Chapter 4 introduces tree-structured filter banks for music signal analysis, describes their features and implementation, and compares them to other front-end signal processing techniques. In Chapter 5, the new front-end processing technique is applied in a polyphonic piano music transcription algorithm as well as for voice separation in a polyphonic piano piece.

CHAPTER 2

CONTINUOUS SHORT-TIME FOURIER AND WAVELET TRANSFORMS

2.1 Introduction

Time-frequency representations are used in signal processing when the primary concern is the localization of specific signal features in both time and frequency. The most popular, linear, continuous-time time-frequency representations are the Short-Time Fourier Transform (STFT) and the Continuous Wavelet Transform (CWT). While the STFT transforms a one-dimensional time signal into a two-dimensional space with the time shift and frequency as variables, the CWT does the same with the time shift and scale as variables. In the case of continuous variables, both representations are highly redundant and do not impose any strong restrictions on either the input signal or the analyzing window function.

Both STFT and CWT are defined as the inner product of the input signal and the analyzing function. In case of the STFT, the analyzing function is a bell-shaped function $w(t)$ modulated by the complex sinusoid $e^{j\omega t}$. Admissibility conditions for the bell-shaped function are reduced to finite energy. On the other hand, the CWT of a function is defined as an inner product between shifted and scaled versions of a single function - the *mother wavelet*, and the function itself. The mother wavelet is more restricted than the bell-shaped function in the STFT; it should satisfy the zero-mean condition.

Since both the CWT and STFT are highly redundant, it is possible to discretize the transform variables and retain reconstruction. For the STFT, the (ω,τ) plane is divided into a rectangular grid of the form $(m\omega_0,n\tau_0)$, $m,n \in \mathcal{Z}$, where ω_0 and τ_0 satisfy $\omega_0\tau_0 \leq 2\pi$. In the CWT case however, a hyperbolic grid is used instead. If the scales are powers of two, the grid is then called dyadic. The time-scale plane (s,τ) is discretized

into $(n\tau_0^m s_0, \pm \tau_0^m)$ so that wide basis functions are shifted in large steps, while the narrow basis functions are shifted in small steps. These discretized versions of the continuous transforms are called frames and they in fact represent overcomplete series expansions. Generally, frames require different synthesis functions than analysis functions except for so called tight frames. In that case the frame behaves as an set of orthonormal vectors that are not independent. There is a trade-off between the sampling rate and the restriction on the prototype function, i.e., if sampling is highly redundant, the prototype functions are less restricted. On the other hand, when sampling is critical, then possible prototype functions become very restricted.

This chapter compares the STFT and the CWT, as well as their frames with respect to their time-frequency resolutions and their orthogonality conditions. Also, orthonormal bases for wavelet series expansion are introduced, and their connection with filter banks is presented. Most of the material in Chapters 2 and 3 is based on the pioneer work of I. Daubechies [61, 62, 63], M. Vetterly [64, 65, 66], and P.P. Vaidyanathan [67, 68].

2.2 Resolution and Uncertainty Principle

2.2.1 Scale and Frequency

In any signal expansion, a primary goal is the localization of a given basis function in time and frequency. There are many ways to define the localization of a particular basis function, and they all calculate the spread of the function in time and frequency. One of the ways may be to define intervals I_t and I_ω which contain 90% of the energy of the time- and frequency-domain functions, respectively. This defines a tile in the time-frequency domain that represents the time-frequency support of the function. Time shift of the function by t_0 will be therefore equivalent to the tile shift by t_0. Consequently, modulation by $e^{j\omega_0 t}$ shifts the tile by ω_0 in frequency. Scaling of a function $f(t)$ by s_0 results in a new function $\hat{f}(t) = f(s_0 t)$ with the support defined by a reshaped tile i.e.,

$\hat{I}_t = I_t / s_0$ and $\hat{I}_\omega = s_0 I_\omega$. These basic operations are illustrated in Figure 2.1. As Figure 2.1 (a) shows, the frequency modulation does not affect the time resolution. However, the scaling trades resolution in frequency for resolution in time, Figure 2.1 (b). Scaling is a fundamental operation used in the wavelet transform, and it will be defined in connection with it. The analysis function for the wavelet transform is defined as

$$\psi_{s,\tau}(t) = \frac{1}{\sqrt{s}} \psi\left(\frac{t-\tau}{s}\right), \quad s \in \mathcal{R} \tag{2.1}$$

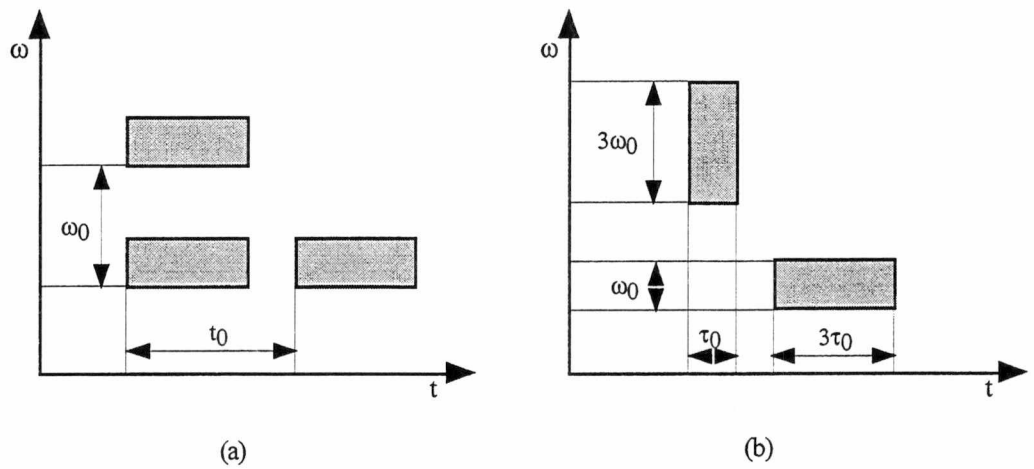


Figure 2.1 Effects of time and frequency shifts on the time-frequency tile. (a) Tile shape does not depend on the shifts in time and frequency. (b) Scale change reshapes the tile.

where the function $\psi(t)$ has the form of the impulse response of a bandpass filter. Large s corresponds to long basis functions that are able to extract long term trends in the signal. Small s makes the basis functions short, and will extract short-term behavior of the signal, i.e., scale is proportional to the duration of the basis function used in the signal expansion. Scale is related to frequency in that signal analysis with large scales (wide basis function) extracts the low frequency contents of the signal, and small-scale analysis (narrow basis function) extracts the high frequency contents. In a qualitative sense, scales are inversely related to frequency.

2.2.2 Uncertainty Principle

As indicated in the previous section, resolution in time can be traded off for resolution in frequency and vice versa. There is no way however, to get arbitrarily fine resolution in both domains simultaneously. This basic property in signal processing is called the uncertainty principle, and is defined by the following theorem.

THEOREM 2.1

If the time width σ_t and the frequency width σ_ω of a unit energy signal $f(t)$ are defined as

$$\sigma_t^2 = \int t^2 |f(t)|^2 dt \tag{2.2}$$

$$\sigma_\omega^2 = \int \omega^2 |F(\omega)|^2 d\omega \tag{2.3}$$

where $F(\omega)$ is the Fourier Transform of the signal, then

$$\sigma_t^2 \sigma_\omega^2 \geq \frac{\pi}{2}. \tag{2.4}$$

Equality in (2.4) holds only for Gaussian signals defined as

$$f(t) = \sqrt{\alpha/\pi} e^{-\alpha t^2}. \tag{2.5}$$

PROOF

Since the Fourier Transform of $f'(t) = df(t)/dt$ is equal to $j\omega F(\omega)$, the following is true

$$\begin{aligned} \sigma_\omega^2 &= \int \omega^2 |F(\omega)|^2 d\omega \\ &= 2\pi \int |f'(t)|^2 dt. \end{aligned} \tag{2.6}$$

Therefore

$$\sigma_t^2 \sigma_\omega^2 = 2\pi \int |tf(t)|^2 dt \int |f'(t)|^2 dt. \tag{2.7}$$

According to the Schwarz inequality

$$\int |tf(t)|^2 dt \int |f'(t)|^2 dt \geq \left| \int tf(t) f'(t) dt \right|^2. \tag{2.8}$$

The integral on right side of (2.8) can be further transformed as follows

$$\int tf(t)f'(t)dt = \frac{1}{2}\int t\frac{\partial f^2(t)}{\partial f}dt$$
$$= \frac{1}{2}tf^2(t)\Big|_{-\infty}^{\infty} - \frac{1}{2}\int f^2(t)dt \qquad (2.9)$$
$$= -\frac{1}{2}$$

since the function is of unit norm. Combining (2.7), (2.8), and (2.9), (2.4) becomes obvious. The Schwarz inequality (2.8) becomes an equality if

$$f'(t) = kf(t). \qquad (2.10)$$

The only function that satisfies (2.10) has the form

$$f(t) = ce^{kt^2/2} \qquad (2.11)$$

and is equal to (2.5) if $k = -2\alpha$ and $c = \sqrt{\alpha/\pi}$. Thus the theorem is proven.

2.3 Continuous Short-Time Fourier Transform

2.3.1 Definition and Properties

The Short-Time Fourier Transform, or short-time spectrum of a function $f(t) \in L_2(\mathcal{R})$ is defined as the inner product of a shifted and modulated window $g_{\omega,\tau}(t) = w(t-\tau)e^{j\omega t}$ and the function itself, i.e.,

$$STFT_f(\omega,\tau) = \left\langle g_{\omega,\tau}(t)f(t)\right\rangle$$
$$= \int_{-\infty}^{\infty} w^*(t-\tau)f(t)e^{-j\omega t}dt \qquad (2.12)$$

where * denotes the complex conjugate. In other words, the STFT at time τ is the standard Fourier Transform of the function multiplied by a window centered around τ. The only constraint on the window $w(t)$ is that it should be of finite energy. It is convenient to use a window that has unit energy, i.e., $\|w(t)\|^2 = 1$.

The reconstruction formula for $f(t)$, or the inverse STFT, is given by:

$$f(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} STFT_f(\omega,\tau)\gamma(t-\tau)e^{j\omega t}d\omega d\tau. \qquad (2.13)$$

The window used in the reconstruction formula should satisfy a relatively weak condition

$$\int w^*(t)\gamma(t)dt = 1. \tag{2.14}$$

There are many synthesis windows that satisfy (2.14), but a natural choice is

$$\gamma(t) = w(t). \tag{2.15}$$

The STFT can also be expressed as the inner product of the signal spectrum, $F(\omega)$, and the spectrum of the analyzing window, $W(\omega)$, i.e.,

$$STFT_f(\omega,\tau) = \frac{1}{2\pi}e^{-j\omega\tau}\int_{-\infty}^{\infty} W^*(\Omega-\omega)F(\Omega)e^{j\Omega\tau}d\Omega. \tag{2.16}$$

According to (2.12) the STFT at time τ is the spectrum of the signal $f(t)$ multiplied beforehand by the window $w(t)$. It means that all signal features located within a local window around time τ show up at time τ in the STFT. Thus, it is clear that good time resolution with the STFT requires a short analyzing window. On the other hand, the STFT at the frequency ω is in fact the result of passing the signal trough the bandpass filter $W^*(\Omega-\omega)$. Good frequency resolution requires a narrow-band filter, i.e. a long analyzing window. As an example, the STFT of $f(t)$ using an infinitely narrow Dirac pulse $w(t) = \delta(t)$ is given by

$$\begin{aligned} STFT_f(\omega,\tau) &= \int_{-\infty}^{\infty}\delta(t-\tau)f(t)e^{-j\omega t}dt \\ &= f(\tau)e^{-j\omega\tau}. \end{aligned} \tag{2.17}$$

In this case, the STFT reduces to the signal itself preserving all time variations, but loosing all frequency information. On the other hand, using an all-constant window $w(t) = 1$, the STFT will be

$$\begin{aligned} STFT_f(\omega,\tau) &= \int_{-\infty}^{\infty} w^*(t-\tau)f(t)e^{-j\omega t}dt \\ &= \int_{-\infty}^{\infty} f(t)e^{-j\omega t}dt \\ &= F(\omega), \end{aligned} \tag{2.18}$$

i.e., the STFT reduces to the FT and does not provide any time information. This time-frequency trade-off results from the uncertainty principle mentioned in the previous section.

For display purposes, it is useful to define a measure for the energy distribution associated with STFT. This measure is called the spectrogram, and is defined as

$$S(\omega,\tau) = |STFT(\omega,\tau)|^2. \tag{2.19}$$

The most important properties of the STFT, the frequency shift, the time shift, and the energy conservation are given below.

* **Frequency shift**

  If $r(t) = f(t)e^{j\omega_0 t}$, then $STFT_r(\omega,\tau) = STFT_f(\omega-\omega_0,\tau)$. (2.20)

* **Time shift**

  If $r(t) = f(t-t_0)$, then $STFT_r(\omega,\tau) = STFT_f(\omega,\tau-t_0)e^{j\omega t_0}$. (2.21)

* **Energy conservation**

  If $STFT_f(\omega,\tau)$ is the STFT of $f(t)$, then the following is true:

$$\|f(t)\|^2 = \frac{1}{2\pi}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\left|STFT_f(\omega,\tau)\right|^2 d\omega d\tau \tag{2.22}$$

  which is equivalent to Parseval's formula for the standard Fourier Transform.

2.3.2 STFT Implementation

The STFT is used whenever both time and frequency localization of a signal are needed. Therefore, for implementation, a time window with good time-frequency localization needs to be used. A rectangular window will have perfect time localization, but poor frequency localization. On the other hand, a sinc window has good frequency localization, but poor time localization. Besides, the sinc window does not have finite support. Therefore, in practical applications, windows between these two extremes are used. The best choice with respect to the time-frequency localization is the Gaussian

window (2.5) where the coefficient α controls the width of the window. The Fourier Transform of the Gaussian window is again a Gaussian window given by:

$$W(\omega) = e^{-\omega^2/4\alpha}. \tag{2.23}$$

If used in the STFT, a Gaussian window produces the Gabor Transform and meets the lower bound of the joint time-frequency localization as mentioned in the previous section. In the following examples [69], the basic limitations regarding the resolution of the STFT are illustrated.

EXAMPLE 2.1

The STFT of a sinusoid defined as $f(t) = e^{j\omega_0 t}$ using the Gaussian window $w(t) = \sqrt{\alpha/\pi}e^{-\alpha t^2}$ is given as

$$\begin{aligned} STFT_f(\omega,\tau) &= \int_{-\infty}^{\infty} \sqrt{\alpha/\pi} e^{-\alpha(t-\tau)^2} e^{j\omega_0 t} e^{j\omega t} dt \\ &= e^{-j(\omega-\omega_0)} e^{-(\omega-\omega_0)^2/4\alpha}. \end{aligned} \tag{2.24}$$

Then the spectrogram is given by

$$S_f(\omega,\tau) = e^{-(\omega-\omega_0)^2/2\alpha}. \tag{2.25}$$

The frequency resolution of the spectrogram depends on α, and can be improved if α is smaller, i.e., a wider analyzing window is used.

EXAMPLE 2.2

If instead of a sinusoid, an impulse $f(t) = \delta(t - t_0)$ is analyzed using the same Gaussian window, then the STFT will be

$$\begin{aligned} STFT_f(\omega,\tau) &= \int_{-\infty}^{\infty} \sqrt{\alpha/\pi} e^{-\alpha(t-\tau)^2} \delta(t-t_0) e^{j\omega t} dt \\ &= \sqrt{\alpha/\pi} e^{j\omega t_0} e^{-\alpha(\tau-t_0)^2} \end{aligned} \tag{2.26}$$

and the spectrogram

$$S_f(\omega,\tau) = (\alpha/\pi) e^{-2\alpha(\tau-t_0)^2}. \tag{2.27}$$

The time resolution increases if α becomes larger, i.e., a narrower analyzing window is used. However, if the input signal contains both fast-time transitions and distinctive

frequency components, no window can give sufficient time and frequency resolutions simultaneously. Once the analyzing window is chosen, all frequencies are analyzed with the same time and frequency resolutions. In other words, the time-frequency resolution is constant over the time-frequency space, unlike the situation where the Wavelet Transform is used in the signal analysis. There the higher frequencies are analyzed with lower frequency resolution and higher time-resolution. The lower frequencies are, in contrast, analyzed with higher frequency resolution and lower time resolution.

2.4 Continuous Wavelet Transform

2.4.1 Definition and Properties

The Continuous Wavelet Transform, $CWT_f(s,\tau)$, of a real valued function $f(t) \in L_2(\mathcal{R})$ is given by

$$\begin{aligned} CWT_f(s,\tau) &= \left\langle \psi_{s,\tau}(t), f(t) \right\rangle \\ &= \int_{-\infty}^{\infty} \psi^*_{s,\tau}(t) f(t) dt, \end{aligned} \tag{2.28}$$

i.e., it is the inner product of the analyzed function $f(t)$ and a family of functions obtained by shifting and scaling a mother wavelet, $\psi(t) \in L_2(\mathcal{R})$, defined as

$$\psi_{s,\tau}(t) = \frac{1}{\sqrt{s}} \psi\left(\frac{t-\tau}{s}\right), \quad s > 0 \tag{2.29}$$

where (s,τ) represent scale and shift respectively. The normalization $1/\sqrt{s}$ ensures constant energy over entire function family, i.e., $\|\psi_{s,\tau}(t)\| = \|\psi(t)\|$. The admissibility condition for $\psi(t)$ is more restrictive than in the case of the STFT, and is given by

$$\int_{-\infty}^{\infty} \frac{|\Psi(\omega)|^2}{|\omega|} d\omega = C_\psi < \infty \tag{2.30}$$

where $\Psi(\omega)$ is the Fourier Transform of $\psi(t)$, and C_ψ is a real constant. There are two conditions imposed by (2.30); the first, $\Psi(\omega)$ has to have sufficient decay, and the second, it has zero DC component, i.e.,

$$\int_{-\infty}^{\infty} \psi(t)dt = \Psi(0) = 0. \tag{2.31}$$

It also means that an admissible mother wavelet, $\psi(t)$, is a bandpass filter impulse response function. There exists the Fourier domain equivalent of (2.28) given by

$$CWT_f(s,\tau) = \frac{\sqrt{s}}{2\pi} \int_{-\infty}^{\infty} \Psi^*(s\omega)F(\omega)e^{j\tau\omega}d\omega \tag{2.32}$$

The inverse CWT, also called the *resolution of identity*, is given by

$$f(t) = \frac{1}{C_\psi} \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} CWT_f(s,\tau)\psi_{s,\tau}(t)\frac{dsd\tau}{s^2} \tag{2.33}$$

Some of the important properties of the CWT are given below.

* **Time shift**

If $CWT_f(s,\tau)$ is the CWT of $f(t)$, then the CWT of $r(t) = f(t-t_0)$ is given by

$$CWT_r(s,\tau) = CWT_f(s,\tau - t_0). \tag{2.34}$$

* **Scale change**

If $CWT_f(s,\tau)$ is the CWT of $f(t)$, then the CWT of $r(t) = \sqrt{s_0}f(s_0t)$ is given by

$$CWT_r(s,\tau) = SWT_f(s_0s, s_0\tau). \tag{2.35}$$

* **Energy conservation**

If $CWT_f(s,\tau)$ is the CWT of $f(t)$, then the following is true

$$\int_{-\infty}^{\infty}|f(t)|^2 dt = \frac{1}{C_\psi}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\left|CWT_f(s,\tau)\right|^2\frac{dsd\tau}{s^2}. \tag{2.36}$$

Equivalently to the spectrogram for the STFT defined in (2.19), for the display of the CWT, a measure called the scalogram is defined as

$$C(s,\tau) = |CWT(s,\tau)|^2. \tag{2.37}$$

2.4.2 Time-Frequency Resolution

To illustrate how the time-frequency resolution is distributed over the time-frequency space in case of the CWT, the following examples are given.

EXAMPLE 2.3

If $f(t) = \delta(t - t_0)$, then the CWT using the Morlet wavelet [70] is given by

$$\begin{aligned} CWT_f(s,\tau) &= \int_{-\infty}^{\infty} \frac{1}{\sqrt{2s\pi}} e^{-j\omega_0\left(\frac{t-\tau}{s}\right)} e^{-(t-\tau)^2/2s^2} \delta(t - t_0)dt \\ &= \frac{1}{\sqrt{2s\pi}} e^{j\omega_0\left(\frac{t_0-\tau}{s}\right)} e^{-(t_0-\tau)^2/2s^2} \end{aligned} \tag{2.38}$$

where the Morlet wavelet is defined as

$$\psi(t) = \frac{1}{\sqrt{2\pi}} e^{j\omega_0 t} e^{-t^2/2}. \tag{2.39}$$

The corresponding scalogram is given by

$$C_f(s,\tau) = \frac{1}{2s\pi} e^{-(\tau - t_0)^2/s^2}. \tag{2.40}$$

According to (2.40), the Dirac pulse is transformed in a ridge-like function with broad support for larger scales and narrow support for smaller scales. In other terms, the CWT gives good time resolution for smaller scales only. Also, the time resolution does not depend on the location of the pulse t_0.

EXAMPLE 2.4

If $f(t) = e^{j\Omega_0 t}$, then the CWT using (2.32) and the same Morlet wavelet is given by

$$\begin{aligned} CWT_f(s,\tau) &= \frac{\sqrt{s}}{2\pi} \int_{-\infty}^{\infty} \Psi^*(s\omega) F(\omega) e^{j\tau\omega} d\omega \\ &= \frac{\sqrt{s}}{2\pi} \int_{-\infty}^{\infty} e^{-(s\omega - \omega_0)^2/2} 2\pi\delta(\omega - \Omega_0) e^{j\omega\tau} d\omega \\ &= \sqrt{s} e^{j\Omega_0\tau} e^{-(s\Omega_0 - \omega_0)^2/2} \end{aligned} \tag{2.41}$$

where $F(\omega) = 2\pi\delta(\omega - \Omega_0)$, and $\Psi(\omega) = e^{-(\omega - \omega_0)^2/2}$. Then the scalogram is given by

$$C_f(s,\tau) = s e^{-(s - \omega_0/\Omega_0)^2 \Omega_0^2}. \tag{2.42}$$

According to the scalogram, the frequency resolution defined as the function width, depends on the signal frequency Ω_0. Knowing that the scale is inversely proportional to frequency, it can be seen from (2.42) that the scalogram is wider in frequency for higher

frequencies and narrower for lower frequencies, and not dependent on the time shift. Figure 2.2 illustrates Examples 2.3 and 2.4.

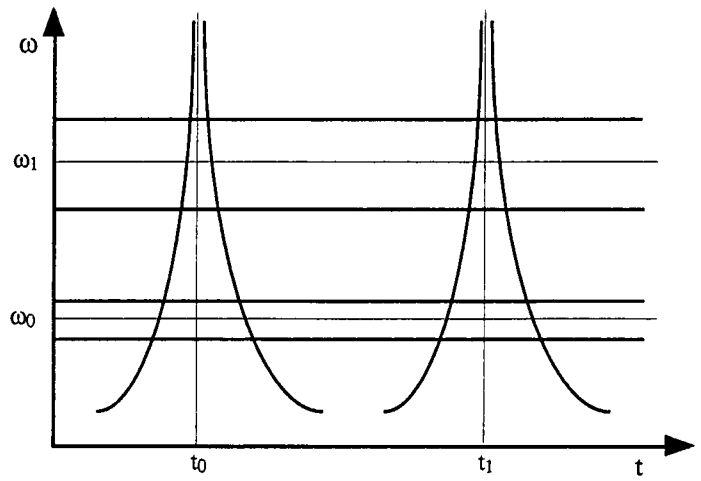


Figure 2.2 Time-frequency resolution for the CWT. Two time-domain Dirac impulses produce two vertical support regions around t_0 and t_1. Two sinusoids are presented by the horizontal support regions around ω_0 and ω_1.

Similarly to the STFT, the CWT can be considered as bandpass filtering with the center frequency at f. While in the STFT case, the bandwidth of the bandpass filter does not depend on the center frequency, in the CWT case, the bandwidth increases with the center frequency thus keeping the Q-factor constant. Therefore, the CWT is sometimes called constant Q-factor analysis.

As far as joint time-frequency resolution is concerned, the CWT suffers from the same time-frequency limitation as the STFT as defined by the principle of uncertainty. However, while the STFT has constant time-frequency resolution over the time-frequency space, the CWT analyzes higher frequencies with lower frequency resolution and higher time resolution, and lower frequencies with lower time and higher frequency resolution as shown in Figure 2.2.

2.5 Frames of Short-Time Fourier and Wavelet Transforms

It is obvious that the STFT and CWT are highly redundant since they transform an one-dimensional, continuous-time signal into a two-dimensional, continuous time-frequency or time-scale space. As a benefit, the analyzing function in both transforms needs to satisfy relatively weak admissibility conditions. However, being continuous-time, these transforms are not directly suitable for implementation with DSP. A question arises; can the time-frequency and time-scale spaces be discretized and still retain reconstruction? The answer is yes, i.e., there exist overcomplete continuous-time expansions called *frames* that are sets of nonindependent vectors obtained by discretizing the continuous-time transforms. Larger sampling rates for the transform variables allow more freedom in selecting the analyzing functions that have better filtering characteristics. Going down with the sampling rate, the admissibility conditions for the analyzing functions become more restrictive. When the sampling is critical, according to the Balian-Low-Coifman-Semmes theorem [62], it is not possible to get frames with good time and frequency resolutions in case of the STFT bases. On the other hand, wavelet frames are less restricted and that is why they have become more popular for expansion of continuous signals.

2.5.1 Discretization of the Time-Frequency and Time-Scale Spaces

Instead of using continuous variables in the STFT and the CWT, it is possible to discretize the analyzing functions $g_{\omega,\tau}(t)$, and $\psi_{s,\tau}(t)$ as follows

$$\begin{aligned} g_{m,n}(t) &= e^{jm\omega_0 t} w(t - n\tau_0), \quad m,n \in \mathcal{Z},\ \tau_o, \omega_0 > 0 \\ \psi_{m,n}(t) &= s_0^{-m/2} \psi\left(s_0^{-m} t - n\tau_0\right), \quad m,n \in \mathcal{Z},\ \tau_o > 0,\ s_0 > 1. \end{aligned} \tag{2.43}$$

Thus, instead of the continuous support for the transforms, a discrete grid is used as illustrated in Figure 2.3. If $s_0 = 2$ and $\tau_0 = 1$, then the *Dyadic Wavelet Transform* is obtained.

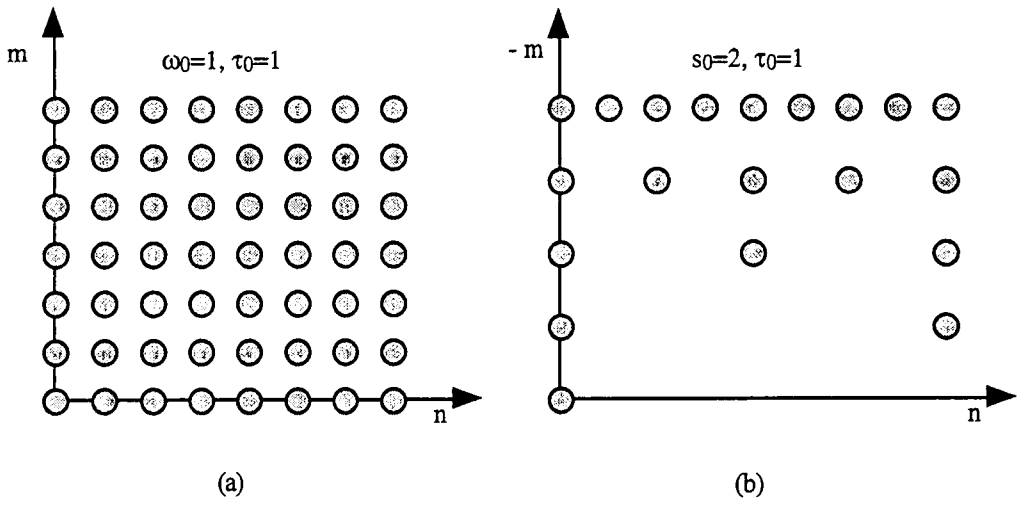


Figure 2.3 Discrete grid support. (a) The STFT. (b) The CWT.

The reconstruction formulas for the continuous-time transforms are given in (2.13) and (2.33). The question is: do numerically stable reconstruction formulas in the discrete-time case exist? The answer is yes, and the general reconstruction formula for the wavelet case is given by

$$f(t) = \sum_m \sum_n \left\langle \psi_{m,n}(t), f(t) \right\rangle \tilde{\psi}_{m,n}(t) \tag{2.44}$$

where $\psi(t)$, $\tilde{\psi}(t)$, s_0, and τ_0 are appropriately chosen. $\{\tilde{\psi}_{m,n}(t)\}$ denotes so-called *dual frames*. To have a numerically stable reconstruction, according to [61], the following needs to be satisfied

$$A\|f(t)\|^2 \le \sum_m \sum_n \left\langle \psi_{m,n}(t), f(t) \right\rangle \le B\|f(t)\|^2 \tag{2.45}$$

where $A \le B$, $A, B \in \mathcal{R}$. If this condition is satisfied, then the function family $\{\psi_{m,n}(t)\}$ constitutes a frame. When $A = B \neq 1$ then the functions constitute so-called *tight frames*, i.e., an orthogonal basis with linearly dependent vectors. Finally, if $A = B = 1$, and $\|\psi_{m,n}(t)\| = 1$, then the family of wavelets is an orthonormal basis.

2.5.2 Orthonormal Frames of Wavelets and STFT

It was mentioned at the beginning of this section that the admissibility conditions for wavelets to constitute a frame are weaker than those for the STFT. For the wavelet case, it is shown in [61] that $\psi_{m,n}(t) = s_0^{-m/2}\psi\left(s_0^{-m}t - n\tau_0\right)$, $m,n \in \mathbb{Z}$ constitutes a frame with the frame bounds A and B if

$$\begin{aligned}\frac{\tau_0 \ln s_0}{2\pi} A \le \int_0^{\infty} \frac{|\Psi(\omega)|^2}{\omega} d\omega \le \frac{\tau_0 \ln s_0}{2\pi} B \\ \frac{\tau_0 \ln s_0}{2\pi} A \le \int_{-\infty}^{0} \frac{|\Psi(\omega)|^2}{|\omega|} d\omega \le \frac{\tau_0 \ln s_0}{2\pi} B.\end{aligned} \tag{2.46}$$

For the tight frame $A=B$, i.e.,

$$\begin{aligned}A &= B \\ &= \frac{2\pi}{\tau_0 \ln s_0} \int_0^{\infty} \frac{|\Psi(\omega)|^2}{\omega} d\omega \\ &= \frac{2\pi}{\tau_0 \ln s_0} \int_{-\infty}^{0} \frac{|\Psi(\omega)|^2}{|\omega|} d\omega.\end{aligned} \tag{2.47}$$

If finally, $A=B=1$ then, for the dyadic case, a wavelet family constitute an orthonormal set if

$$\begin{aligned}\int_0^{\infty} \frac{|\Psi(\omega)|^2}{\omega} d\omega &= \int_{-\infty}^{0} \frac{|\Psi(\omega)|^2}{|\omega|} d\omega \\ &= \frac{\ln 2}{2\pi}.\end{aligned} \tag{2.48}$$

As shown in [61], the condition (2.46) is not as restrictive, and if $\psi(t)$ satisfies the general admissibility condition given in (2.30), there exist many combinations of s_0 and τ_0 such that $\{\psi_{m,n}(t)\}$ constitutes a frame. In the STFT case however, the situation is more complicated in the sense that there are more restrictions for the analyzing function. These restrictions are defined by the Balian-Law-Coifman-Semmes theorem given below. The proof for the theorem can be found in [62].

THEOREM 2.2

If the analyzing functions $g_{m,n}(t) = e^{j2\pi mt} w(t-n)$, $m,n \in \mathcal{Z}$ constitute a frame for $L_2(\mathcal{R})$, then either $\int t^2 |w(t)|^2 dt = \infty$, or $\int \omega^2 |W(\omega)|^2 d\omega = \infty$.

The theorem states that no orthonormal bases for the STFT exist that have good time-frequency resolution. Further analysis of the theorem can be done using the conditions for the frame bounds for the STFT [62] given as

$$A \le \frac{2\pi}{\omega_0 \tau_0} \|g(t)\|^2 \le B. \tag{2.49}$$

Assuming $\|g(t)\| = 1$ and $\omega_0 \tau_0 = 2\pi$, the function family $\{g_{m,n}(t)\}$ constitutes a tight frame with $A=B=1$. However, it does not necessarily mean that $\{g_{m,n}(t)\}$ constitutes an orthonormal basis. It only means that the STFT is critically sampled [64]. If $\omega_0 \tau_0 > 2\pi$, according to (2.49), no frames exist. Frames with good time-frequency resolution exist only if $\omega_0 \tau_0 < 2\pi$, but the STFT then becomes redundant. In the wavelet case however, there exist orthonormal frames with good time-frequency resolution, as shown in the next section.

2.6 Continuous-Time Orthogonal Bases of Wavelets

Below are given some examples of the continuous-time orthogonal bases of wavelets. Also, an alternative approach to wavelet theory, multiresolution analysis [71], is presented, and its use for wavelet construction is explained. It is also, shown that there exists a close connection between wavelets and filter banks.

2.6.1 Haar and Sinc Wavelets

Haar expansion [72] has been known for long time, but only recently its connection with wavelets was established. In fact, it represents an orthonormal basis for wavelets. The Haar wavelet is defined as

$$\psi(t) = \begin{cases} 1, & 0 \le t < 1/2 \\ -1, & 1/2 \le t < 1 \\ 0, & otherwise \end{cases} \tag{2.50}$$

Then the corresponding wavelet family, using the dyadic grid, is defined as $\psi_{m,n}(t) = 2^{-m/2}\psi(2^{-m}t - n)$, $m,n \in \mathbb{Z}$. To be orthogonal, the family needs to satisfy the following condition

$$\left\langle \psi_{m,n}(t), \psi_{m',n'}(t) \right\rangle = \delta(m - m')\delta(n - n'), \tag{2.51}$$

i.e., it needs to be orthogonal in both dimensions. Figure 2.4 illustrates the Haar wavelet family as well as its orthogonality in both dimensions.

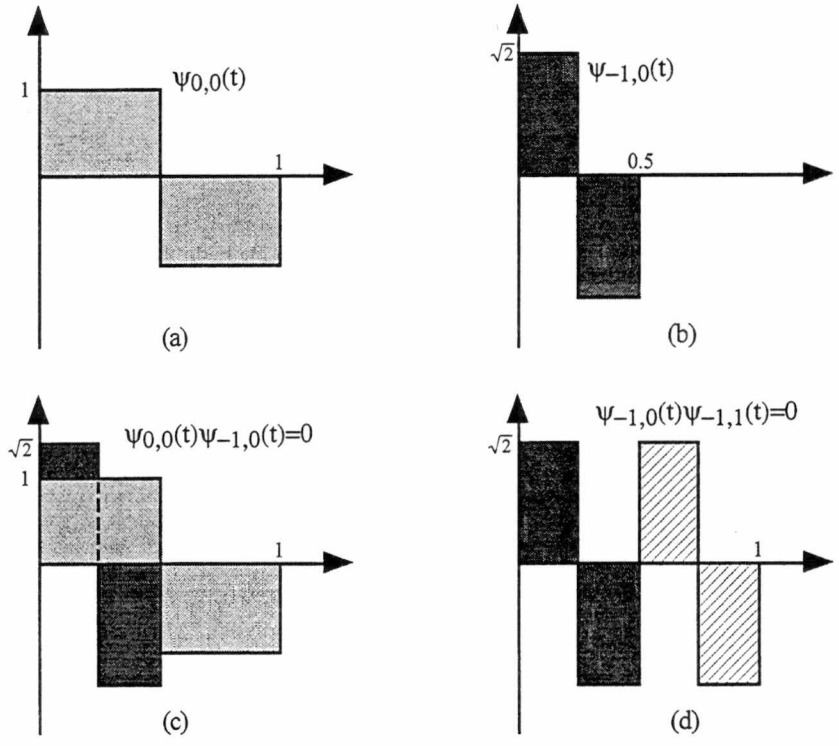


Figure 2.4 The Haar wavelet family. (a) The mother wavelet. (b) Scaled-by-2 wavelet. (c) Orthogonality with respect to the scale. (d) Orthogonality with respect to the time shift.

The Haar wavelets are well localized in time, and, when $m \to -\infty$, they have arbitrarily good time resolution. However, the frequency resolution is relatively poor since the FT of (2.50) is a sinc function with $1/\omega$ decay in frequency having also an unlimited support. For the sinc wavelet, the situation is just the opposite; good

frequency resolution is possible because of the rectangular spectrum of the sinc function. Time resolution however, is limited since the sinc function has infinite support and $1/t$ decay. The sinc wavelet is defined as

$$\psi(t) = \frac{\sin(\pi t/2)}{\pi t/2}\cos(3\pi t/2). \tag{2.52}$$

It is orthogonal to its translates by integers as well as with respect to the dyadic scaling. In practical applications, wavelets between these two extreme cases, the Haar wavelet and the sinc wavelet, are used.

2.6.2 Multiresolution Analysis

Expansion of time signals using an orthonormal basis can be considered as a successive approximation where, in each step of the approximation, the signal is analyzed in more detail. This approach to signal decomposition is called multiresolution analysis [71], and is defined by the next set of axioms.

DEFINITION 2.1

Multiresolution analysis is represented by a sequence of embedded closed subspaces

$$\ldots V_2 \subset V_1 \subset V_0 \subset V_{-1} \subset V_{-2} \ldots \tag{2.53}$$

such that

(i) $$\overline{\bigcup_{m\in(\mathcal{Z})} V_m} = L_2(\mathcal{R}) \tag{2.54}$$

(ii) $$\bigcap_{m\in(\mathbf{Z})} V_m = \{0\} \tag{2.55}$$

(iii) $$f(t) \in V_m \Leftrightarrow f\left(2^m t\right) \in V_0 \tag{2.56}$$

(iv) $$f(t) \in V_0 \Rightarrow f(t-n) \in V_0, \quad n \in \mathcal{Z} \tag{2.57}$$

(v) There exists $\varphi \in V_0$, so that

$$\{\varphi(t-n), n \in \mathcal{Z}\} \tag{2.58}$$

constitutes an orthonormal basis.

The function $\varphi(t)$ is called the *scaling function.* The orthogonality condition (2.58) has its equivalent in the Fourier domain [64] given as

$$\sum_{k=-\infty}^{\infty}|\Phi(\omega+2k\pi)|^2=1. \tag{2.59}$$

Some important conclusions can be drawn from the above axioms; If $\{\varphi(t-n), n\in\mathcal{Z}\}$, as per (2.58), is the orthonormal basis for V_0, then the orthonormal basis for V_{-m} space is $\{2^{m/2}\varphi(2^m t-n), n\in\mathcal{Z}\}$. Furthermore, since $\varphi(t)$ is in both V_0 and V_{-1}, it can be expressed as a linear combination of basis functions from V_{-1}, i.e.,

$$\varphi(t)=\sqrt{2}\sum_{n=-\infty}^{\infty}h_0(n)\varphi(2t-n), \qquad n\in\mathcal{Z}. \tag{2.60}$$

In the Fourier domain, (2.60) has the form

$$\Phi(\omega)=\frac{1}{\sqrt{2}}H_0(e^{j\omega/2})\Phi(\omega/2), \tag{2.61}$$

where

$$H_0(e^{j\omega})=\sum_n h_0(n)e^{-j\omega n}. \tag{2.62}$$

The above function, $H_0(e^{j\omega})$, characterizes a multiresolution analysis, and, as will be shown in continuation, it can be used for construction of wavelets. An important property of $H_0(e^{j\omega})$ that follows from (2.59) and (2.61) is given by

$$\left|H_0(e^{j\omega})\right|^2+\left|H_0\left(e^{j(\omega+\pi)}\right)\right|^2=2. \tag{2.63}$$

The following theorem [61] is crucial in wavelet construction using the multiresolution approach.

THEOREM 2.3

If a sequence of spaces satisfies (2.53 - 2.58), then an orthonormal basis in $L_2(\mathcal{R})$ exists, and it is given by:

$$\psi_{m,n}(t)=2^{-m/2}\psi\left(2^{-m}t-n\right), \qquad m,n\in\mathcal{Z}. \tag{2.64}$$

The basis is orthonormal for W_m, where W_m is the orthogonal complement of V_m in V_{m-1}.

The proof for the above theorem can be found in [61], and here are given only some useful results of the theorem. Similarly to (2.60), there exists an equivalent relation for the wavelet

$$\psi(t) = \sqrt{2}\sum_{n=-\infty}^{\infty} h_1(n)\varphi(2t-n), \qquad n \in \mathbb{Z} \tag{2.65}$$

and its Fourier Transform

$$\Psi(\omega) = \frac{1}{\sqrt{2}} H_1(e^{j\omega/2})\Phi(\omega/2), \tag{2.66}$$

where

$$\begin{aligned} H_1(e^{j\omega}) &= -e^{-j\omega} H_0^*\left(e^{j(\omega+\pi)}\right) \\ h_1(n) &= (-1)^n h_0(-n+1). \end{aligned} \tag{2.67}$$

Finally, the wavelet can be obtained using

$$\Psi(\omega) = -\frac{1}{\sqrt{2}} e^{-j\omega/2} H_0^*\left(e^{j(\omega/2+\pi)}\right)\Phi(\omega/2), \tag{2.68}$$

or in the time domain

$$\psi(t) = \sqrt{2}\sum_{n}(-1)^n h_0(-n+1)\varphi(2t-n) \tag{2.69}$$

Using (2.59 - 2.63), additional important relations between the discrete filters $H_0(e^{j\omega})$ and $H_1(e^{j\omega})$ can be obtained

$$\begin{aligned} H_0(e^{j\omega})H_1\left(e^{j(\omega+\pi)}\right) - H_1(e^{j\omega})H_0\left(e^{j(\omega+\pi)}\right) &= 2e^{-j\omega} \\ H_0^*(e^{j\omega})H_1(e^{j\omega}) + H_0^*\left(e^{j(\omega+\pi)}\right)H_1\left(e^{j(\omega+\pi)}\right) &= 0 \end{aligned} \tag{2.70}$$

where $H_0(e^{j\omega})$ and $H_1(e^{j\omega})$ represent lowpass and highpass filters respectively.

2.6.3 Construction of Wavelets

Design of orthogonal wavelet bases using multiresolution analysis relies on selecting a feasible scaling function, or calculating it using both (2.60) and the beforehand determined discrete-time filter $H_0(e^{j\omega})$. Many different wavelets that are othonormal or

almost orthonormal, and have acceptable time-frequency resolution were proposed recently [63, 73, 74].

According to (2.65), the wavelet $\psi(t)$ can be considered as the output of a filter, $H_1(e^{j\omega})$, where the input is $\varphi(2t)$. Since $\varphi(2t)$, according to (2.60), is the output of the filter $H_0(e^{j\omega})$ with $\varphi(4t)$ at the input, an infinite filter structure can be used to represent the above relations, as given in Figure 2.5.

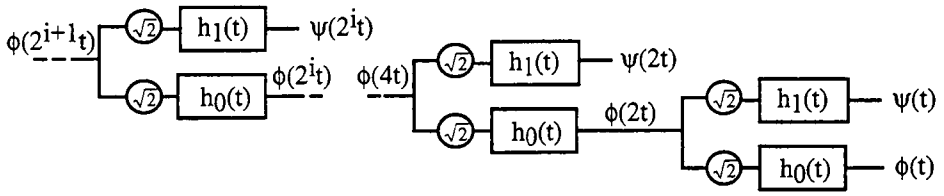


Figure 2.5 Filter structure that leads to wavelet construction.

The filter structure also illustrates how the wavelets and the filter banks are related. According to (2.61), (2.66), and Figure 2.5

$$\Phi(\omega) = \prod_{i=1}^{\infty} 2^{-i/2} H_0(e^{j\omega/2^i}) \tag{2.71}$$

$$\Psi(\omega) = 2^{-1/2} H_1(e^{j\omega/2}) \prod_{i=2}^{\infty} 2^{-i/2} H_0(e^{j\omega/2^i}). \tag{2.72}$$

The above equations can be used to construct a wavelet, assuming the infinite products converge.

Special attention in this chapter was placed on relations between resolution and orthogonality for the continuous-time STFT and CWT. It was shown that a trade-off between the orthogonality and the resolution exists, and that the CWT offers better time-frequency resolution retaining the orthogonality. Comparison of these two transforms with respect to the resolution and orthogonality for the discrete-time case is a subject of the next chapter.

CHAPTER 3

DISCRETE-TIME BASES AND FILTER BANKS

3.1 Introduction

This chapter continues to elaborate on the most important relationships between orthogonality and resolution that are, for continuous-time bases, established in Chapter 2. Here, the discrete-time orthogonal bases are introduced, and their time-frequency resolution characteristics are analyzed. Again, as in Chapter 2, the emphasis is on the comparison of the STFT and the WT, two of the most important linear time-frequency representations.

The discrete-time STFT, similar to its continuous-time counterpart, has very limited time-frequency resolution if it becomes orthogonal. For overcomplete STFT expansions, there is more freedom in the selection of the synthesizing window that has good time-frequency resolution. The Discrete-Time Wavelet Transform (DWT) however, allows one to construct a discrete wavelet basis being both orthogonal and selective at the same time.

At the end of Chapter 2, the connection between the orthogonal bases and filter banks was established. This connection for discrete-time bases is even more important. It turns out that discrete-time orthogonal bases have direct theoretical support in discrete-time filter banks. Moreover, in practical application, the discrete-time orthogonal expansions are performed using the discrete-time filters.

The theory of orthogonal filter banks goes back to 1976 when Quadrature Mirror Filters (QMF) were discovered [75]. This was the first split of a signal into two subbands where alias-free reconstruction was possible. Soon after, so-called perfect reconstruction filter banks were introduced. Besides the basic two-channel case, many other orthogonal and biorthogonal structures of multichannel filter banks have been

presented recently. In those filter banks, all different kinds of filters, FIR, IIR, linear-phase, nonlinear-phase, one- or two-dimensional were used. However, the primary concern in this chapter are orthogonal structures with linear-phase FIR filters since they have many advantages, especially in practical applications.

3.2 Discrete-Time Signal Expansions

The orthonormal expansion of the signal $f(n) \in l_2(\mathbb{Z})$ is given by

$$f(n) = \sum_k F(k)\xi_k(n) \tag{3.1}$$

where

$$F(k) = \sum_n \xi_k^*(n)f(n) \tag{3.2}$$

is the transform of $f(n)$ with respect to the basis sequences $\xi_k(n)$. For (3.1) to be orthonormal, the following needs to be satisfied

$$\sum_n \xi_k^*(n)\xi_l(n) = \delta(k-l). \tag{3.3}$$

The biorthogonal expansion of the signal $f(n)$ with respect to the dual basis sequences $\xi_k(n)$ and $\tilde{\xi}_k(n)$ is given by

$$f(n) = \sum_k \tilde{F}(k)\xi_k(n) \tag{3.4}$$

where

$$\tilde{F}(k) = \sum_n \tilde{\xi}_k^*(n)f(n). \tag{3.5}$$

The biorthogonality constraint is now

$$\sum_n \xi_k^*(n)\tilde{\xi}_l(n) = \delta(k-l). \tag{3.6}$$

The discrete-time STFT and DWT have the same general form as (3.1) with the basis sequences in the form of modulated windows in the STFT case, and discrete wavelets in the DWT case.

3.2.1 Discrete-Time STFT

The discrete-time STFT of a sequence $f(n)$ is defined as a two dimensional sequence given by

$$STFT_f(k,m) = \sum_{p=mM-N+1}^{mM} h(mM-p)f(p)e^{-j2\pi pk/N}, \quad m \in \mathbb{Z}, k = 0,1,...,N-1 \qquad (3.7)$$

where $h(n)$ is an N-tap analyzing window, and M is the window shift. The input signal $f(n)$ can be recovered from its discrete-time STFT using the following formula [76]

$$f(n) = \frac{1}{N}\sum_{m=-\infty}^{\infty}\sum_{k=0}^{N-1}\gamma(n-mM)STFT_f(k,m)e^{j2\pi nk/N} \qquad (3.8)$$

where $\gamma(n)$ is an N-tap synthesizing window. The pair of equations (3.7, 3.8) can be described as an N-channel analysis/synthesis filter bank as shown in Figure 3.1.

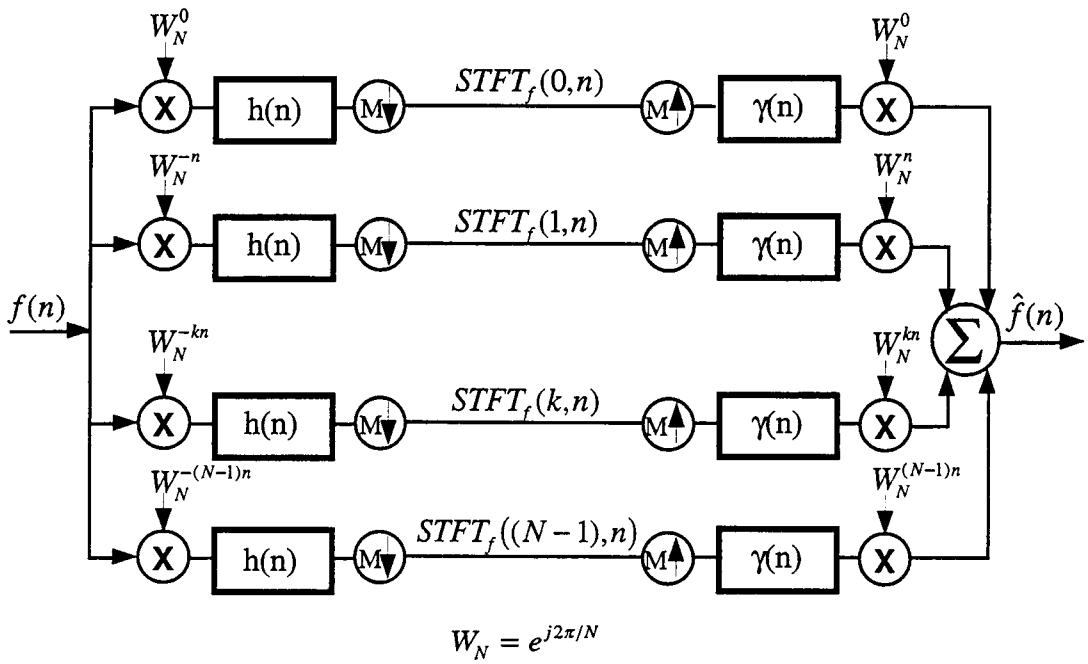


Figure 3.1 N-channel analysis/synthesis filter bank that performs the discrete-time STFT.

If $N > M$, the system from Figure 3.1 represents an overcomplete expansion, while for $N = M$ the system becomes orthogonal. Obviously, reconstruction is impossible if $N < M$. To be orthogonal, besides the condition $N = M$, the filter bank should satisfy the following biorthogonality condition [76]

$$\sum_{m=-\infty}^{\infty}\gamma(n-mN)h[(p+m)N-n]=\delta(p) \quad for \quad all \quad n. \tag{3.9}$$

Since (3.9) is periodic in n with period N, it represents an $N\times N$ linear system where $\gamma(n)$, $n=0,1,...,N-1$ can be calculated given $h(n)$, $n=0,1,...,N-1$. It is shown in [76] that the system (3.9) has solutions under the following conditions

$$\Gamma(\omega,m)=\begin{cases}N, & \omega=0,m=0\\ 0, & m=\pm M,\pm 2M,...\\ 0, & \omega=2\pi/N,4\pi/N,...,2\pi(N-1)/N\end{cases} \tag{3.10}$$

where

$$\Gamma(\omega,m)=\sum_{n=-\infty}^{\infty}h(m-n)\gamma(n)e^{-j\omega n}. \tag{3.11}$$

The basic constraint of an orthogonal discrete-time STFT regarding time-frequency resolution is given in a theorem [64] equivalent to THEOREM 2.2. It says that there are no finite support bases for an orthogonal STFT. However, in the wavelet case, it is possible to have an orthogonal expansion with good time and frequency resolution.

3.2.2 Discrete-Time Wavelet Transform

The Discrete-Time Wavelet Transform of a time sequence $f(n)$ defined on the dyadic grid, is given by

$$DWT_f(r,m)=\sum_k \psi^*_{r,m}(k)f(k) \tag{3.12}$$

where $\{\psi_{r,m}(n)\}$, $r\in\mathcal{N}$, $m,n\in\mathcal{Z}$ represents a discrete wavelet family defined as

$$\psi_{r,m}(n)=\psi_{r,0}(n-2^r m). \tag{3.13}$$

The scale index r represents scaling of the wavelet so that $\psi_{r+1,m}(n)$ is the upsampled-by-2 version of $\psi_{r,m}(n)$ and downsampled-by-2 version of $\psi_{r+2,m}(n)$. Unlike the continuous-time case, the scaling operation is not shape-invariant. Therefore, the wavelet family cannot be obtained by shifting and resampling of a single mother wavelet. Discrete wavelets and the DWT are instead represented by a perfect reconstruction filter bank. It

is shown in [62] that the mother wavelets, $\psi_{r,0}(n)$, from (3.13) can be obtained from a lowpass analyzing filter according to

$$\psi_{r,0}(n) = F^{-1}\left\{\prod_{k=o}^{r-1} H_0(e^{j2^k\omega})\right\}, \quad r \in \mathcal{N} \tag{3.14}$$

where F^{-1} denotes the Inverse Fourier Transform.

The original sequence $f(n)$ can be recovered using the following formula

$$f(n) = \sum_r \sum_m DWT_f(r,m)\tilde{\psi}_{r,m}(n) \tag{3.15}$$

where $\{\tilde{\psi}_{r,m}(n)\}$ represents the synthesizing wavelet family. The analyzing and the synthesizing wavelets need to satisfy the biorthogonality condition

$$\sum_k \tilde{\psi}^*_{r,m}(k)\psi_{s,p}(k) = \delta(r-s)\delta(m-p). \tag{3.16}$$

This is equivalent to the existence of the perfect reconstruction filter bank [77] defined by the analysis filters, $H_0(z)$ and $H_1(z)$, and the synthesis filters, $G_0(z)$ and $G_1(z)$. The DWT can be implemented using those filters as shown in Figure 3.2.

To perform a biorthogonal wavelet expansion, the filters from the perfect reconstruction filter bank need to satisfy the following conditions [64] equivalent to (3.16)

$$\begin{aligned} G_0(z)H_0(z) + G_1(z)H_1(z) &= 2 \\ G_0(z)H_0(-z) + G_1(z)H_1(-z) &= 0. \end{aligned} \tag{3.17}$$

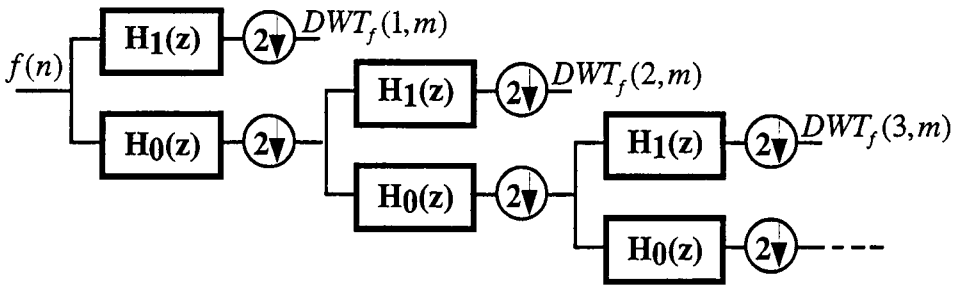
In the case of the orthogonal DWT, the following relations between the analyzing and the synthesizing filters exist

$$\begin{aligned} H_0(z) &= G_0(-1/z) \\ H_1(z) &= G_1(-1/z). \end{aligned} \tag{3.18}$$

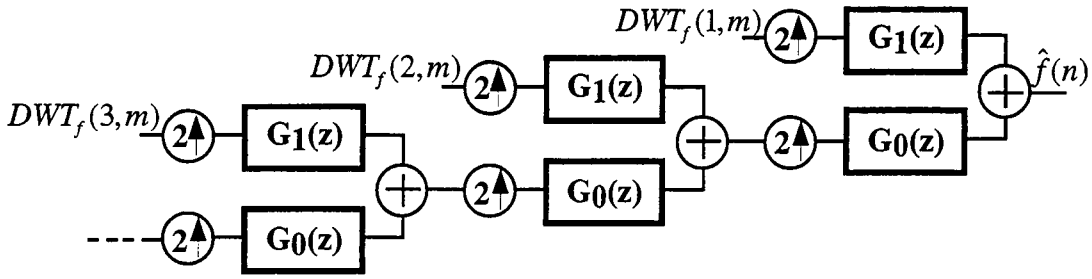
Equivalently, the biorthogonality condition (3.16) becomes the orthogonality condition

$$\sum_k \psi^*_{r,m}(k)\psi_{s,p}(k) = \delta(r-s)\delta(m-p). \tag{3.19}$$

If the perfect reconstruction FIR filter bank performs the orthogonal DWT, then all the filters are defined by only one filter, $H_0(z)$, according to the following relations [64]



(a)



(b)

Figure 3.2 Discrete Wavelet Transform. (a) The analysis structure. (b) The synthesis structure.

$$
\begin{aligned}
&H_0(z)H_0(1/z) + H_0(-z)H_0(-1/z) = 2 \\
&H_1(z) = z^{2k-1}H_0(-1/z), k \in \mathbb{Z} \\
&G_0(z) = H_0(-1/z) \\
&G_1(z) = -z^{-2k+1}H_0(z)
\end{aligned}
\tag{3.20}
$$

Selecting $H_0(z)$ as a good lowpass filter satisfying the first condition in (3.20), one can perform an orthonormal wavelet expansion with good time-frequency resolution.

3.3 Two-Channel Filter Banks

3.3.1 General Form Filter Banks

The two-channel filter bank is presented in Figure 3.3. Its output is given by

$$\hat{X}(z) = \frac{1}{2}\mathbf{G}(z)\mathbf{H}(z)\mathbf{X}(z) \tag{3.21}$$

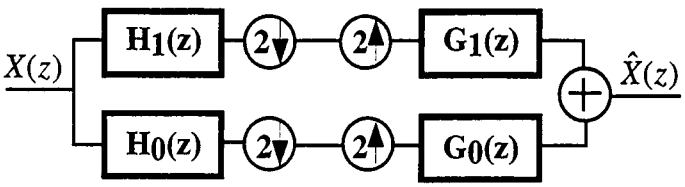


Figure 3.3 Two-channel filter bank.

where

$$\mathbf{G}(z) = \begin{bmatrix} G_0(z) & G_1(z) \end{bmatrix}$$
$$\mathbf{H}(z) = \begin{bmatrix} H_0(z) & H_0(-z) \\ H_1(z) & H_1(-z) \end{bmatrix} \tag{3.22}$$
$$\mathbf{X}(z) = \begin{bmatrix} X(z) \\ X(-z) \end{bmatrix}.$$

For perfect reconstruction, allowing a time delay only, $\hat{X}(z) = z^{-2k+1}X(z)$, the following condition should be satisfied [78]

$$\mathbf{G}(z)\mathbf{H}(z) = \begin{bmatrix} 2z^{-2k+1} & 0 \end{bmatrix}. \tag{3.23}$$

The above condition can be rewritten as

$$\begin{bmatrix} G_0(z) \\ G_1(z) \end{bmatrix} = \frac{2z^{-2k+1}}{\det(\mathbf{H}(z))} \begin{bmatrix} H_1(-z) \\ -H_0(-z) \end{bmatrix}. \tag{3.24}$$

A solution for (3.24) exists only if $\mathbf{H}(z)$ is nonsingular. Then a simple choice

$$\begin{aligned} G_0(z) &= H_1(-z) \\ G_1(z) &= -H_0(-z) \end{aligned} \tag{3.25}$$

leads to

$$\begin{aligned} \det(\mathbf{H}(z)) &= H_0(z)H_1(-z) - H_0(-z)H_1(z) \\ &= 2z^{-2k+1}. \end{aligned} \tag{3.26}$$

Most of the design methods reported recently use relations (3.25, 3.26) to design perfect reconstruction filter banks. The first solutions were proposed in [62, 79, 80] followed by biorthogonal solutions [65]. These solutions use so-called spectral factorization for the filter bank design. Linear phase FIR filters [81, 82, 83] are designed using different nonlinear programming techniques. A different approach that uses lattice factorization [84, 85] has shown some advantages in filter bank design being more stable

than some of the previous techniques.

3.3.2 Polyphase Representation

The two-channel structure from Figure 3.3 can be optimized since in (3.26) both $H_i(z)$ and $H_i(-z)$ are involved. Using the polyphase representation [86], filters $H_i(z)$ and $H_i(-z)$ can be represented in terms of their polyphase components using the forward polyphase representation as

$$\begin{aligned} H_i(z) &= H_{i,0}(z^2) + zH_{i,1}(z^2) \\ H_i(-z) &= H_{i,0}(z^2) + z^{-1}H_{i,1}(z^2). \end{aligned} \tag{3.27}$$

Similarly, the synthesizing filters can be represented using the inverse polyphase representation as

$$\begin{aligned} G_i(z) &= G_{i,0}(z^2) + z^{-1}G_{i,1}(z^2) \\ G_i(-z) &= G_{i,0}(z^2) + zG_{i,1}(z^2). \end{aligned} \tag{3.28}$$

Thus, four filters $H_i(z), H_i(-z), i = 0,1$ are represented by four filters $H_{i,j}(z^2), i,j = 0,1$ that are half their length. This brings a significant reduction in the number of operations when the polyphase representation is implemented in a multilevel filter structure.

The output of the two-channel filter bank using polyphase representation is given by

$$\hat{X}(z) = [1 \quad z^{-1}]\mathbf{G}_p(z^2)\mathbf{H}_p(z^2)\mathbf{X}_p(z^2) \tag{3.29}$$

where

$$\begin{aligned} \mathbf{G}_p(z^2) &= \begin{bmatrix} G_{00}(z^2) & G_{10}(z^2) \\ G_{01}(z^2) & G_{11}(z^2) \end{bmatrix} \\ \mathbf{H}_p(z^2) &= \begin{bmatrix} H_{00}(z^2) & H_{10}(z^2) \\ H_{01}(z^2) & H_{11}(z^2) \end{bmatrix} \\ \mathbf{X}_p(z^2) &= \begin{bmatrix} X_0(z^2) \\ X_1(z^2) \end{bmatrix}. \end{aligned} \tag{3.30}$$

The corresponding filter bank structure is illustrated in Figure 3.4. $\mathbf{H}_p(z)$ represents the

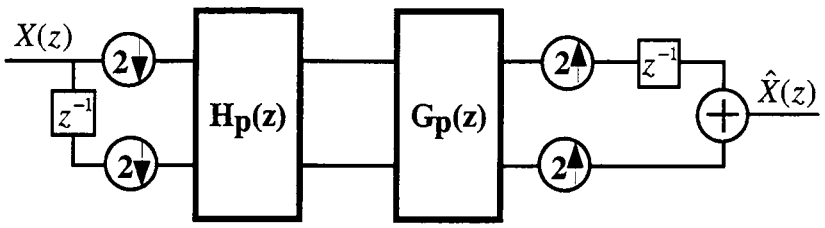


Figure 3.4 Two-channel filter bank represented by a polyphase structure.

analysis polyphase matrix, while $\mathbf{G}_p(z)$ represents the synthesis polyphase matrix. It is shown in [64] that the computational complexity of polyphase analysis is lower compared to the general form filter bank. The complexity is equal to $L/2$ multiplications per input sample for polyphase filter banks, compared to L multiplications per input sample for the general structure, where L represents the order of the filters in the bank.

3.3.3 Design of Linear-Phase FIR Filter Banks

Of a special interest are FIR filters with linear phase since they do not require phase correction after each step when used in a multilevel structure as given in Figure 3.2. However, there are no orthogonal filter banks using linear-phase FIR filters except for some trivial cases [66]. Many biorthogonal filter banks with linear-phase FIR filters have been proposed recently. Design methods used in these filter banks use different nonlinear optimization techniques [81, 82, 83], spectral factorization [62, 65, 66], and lattice factorization [67, 68, 85]. The QMF filter banks, as nearly-perfect reconstruction filter banks, are used very often since they are represented by a single prototype filter, i.e., the highpass filter is defined in terms of the lowpass filter, $H_1(z) = H_0(-z)$. The QMF filters are designed using different forms of nonlinear optimization. The above mentioned design methods are described briefly in the following. Also a new design, originally presented in this dissertation, is elaborated in more detail.

A. Nonlinear Optimization

For a linear-phase FIR filter, the following relation exists

$$H(z) = \pm z^{-L+1} H(1/z) \tag{3.31}$$

where ± means symmetric/antisymmetric, and L is the order of the filter. It is shown in [68] that only two types of filter banks yield nontrivial filters:

1. Both filters have even length and opposite symmetry
2. Both filters have odd length and are symmetric

Furthermore, the sum of the lengths of the analyzing filters must be a multiple of 4.

In a method proposed in [83], the Lagrange Multiplier Approach is used to design filters from the above groups. The method uses the biorthogonality constraint (3.26) in the time domain. The objective function to be minimized is defined as the out-of-passband power for the analyzing filters $H_0(z)$ and $H_1(z)$. As shown in [83], it is possible to get the closed form expressions for the impulse responses of the filters. Figure 3.5 shows the frequency response for a filter pair designed by this method.

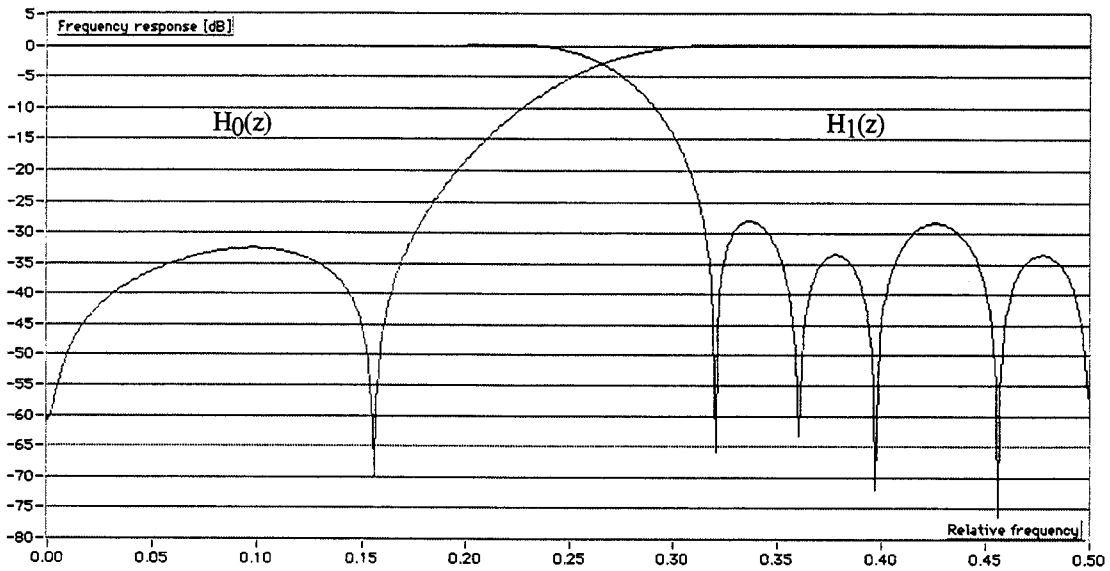


Figure 3.5 Frequency response of a perfect reconstruction filter pair designed using nonlinear optimization [83]. Both filters have order equal to 22.

B. Spectral Factorization

This method uses the biorthogonality constraint in the form

$$H_0(z)H_1(-z) - H_0(-z)H_1(z) = P(z) - P(-z) = 2z^{-2k+1} \tag{3.32}$$

where the polynomial $P(z) = H_0(z)H_1(-z)$ is designed first, and then factored into two parts representing $H_0(z)$ and $H_1(-z)$. A similar method, called the complementary filter method, starts with a known filter $H_0(z)$ and then solves a system of linear equations to find a complementary filter leading to a valid polynomial $P(z)$. Figure 3.6 shows a perfect reconstruction filter pair from [66] designed by this method.

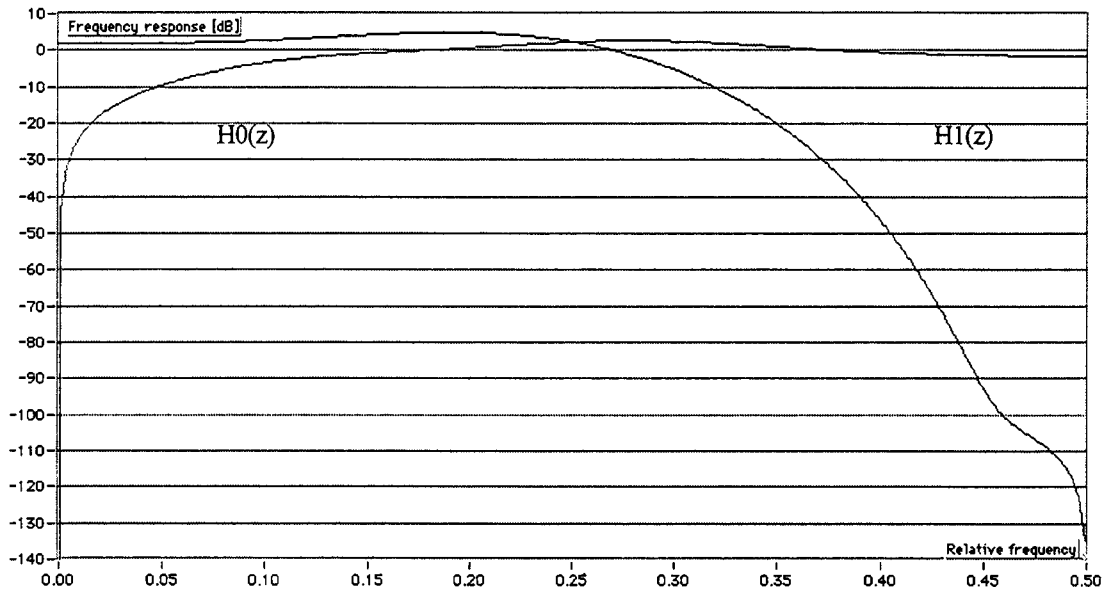


Figure 3.6 Frequency response of a perfect reconstruction filter pair designed using spectral factorization [66]. Both filters have order equal to 18.

C. Lattice Factorization

Every linear phase FIR filter can be represented in a lattice structure as shown in [68]. In the case of even length filters these structures are similar to the well-known linear prediction lattice structures as shown in Figure 3.7. The optimization procedure starts with a known filter pair $H_0(z), H_1(z)$ and then optimizes the set of lattice coefficients $\{k_m\}$ so that the following objective function is minimized

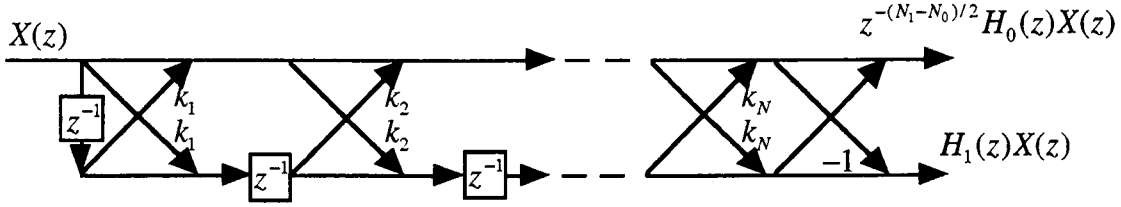


Figure 3.7 A 2x2 lattice structure implemented as a perfect reconstruction filter pair.

$$\Phi = \int_0^{\omega_p} \left[1 - \left|H_0(e^{j\omega})\right|\right]^2 d\omega + \int_{\omega_s}^{\pi} \left[H_0(e^{j\omega})\right]^2 d\omega$$
$$+ \int_{\omega_s}^{\pi} \left[1 - \left|H_1(e^{j\omega})\right|\right]^2 d\omega + \int_0^{\omega_p} \left[H_1(e^{j\omega})\right]^2 d\omega \tag{3.33}$$

where ω_p and ω_s are the passband and the stopband frequencies respectively.

In the case of odd-length filters, the lattice has the structure shown in Figure 3.8.

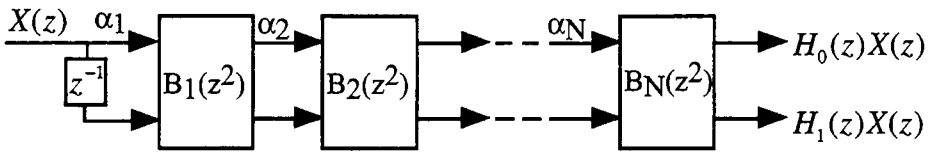


Figure 3.8 Lattice structure for implementation of odd length filters.

Each of the blocks in the structure is represented by a matrix

$$\mathbf{B}_m(z) = \begin{bmatrix} 1+z^{-1} & 1 \\ 1+a_m z^{-1}+z^{-2} & 1+z^{-1} \end{bmatrix}. \tag{3.34}$$

The optimization procedure optimizes $\{\alpha_m\}$ for arbitrarily chosen $\{a_m\}$ using the same objective function (3.33). Figure 3.9 shows the frequency response of the perfect reconstruction filter pair designed by lattice factorization [68].

Lattice structures offer some advantages over other perfect reconstruction filters. Their implementation complexity is lower, and they are less sensitive to the quantization error [89]. For example, an L samples long QMF filter requires $L/2$ multiplications and $L/2$ additions per input sample. The same length lattice filter requires $(L/2+2)/2$ multiplications and $(3L/4+1)$ additions per input sample.

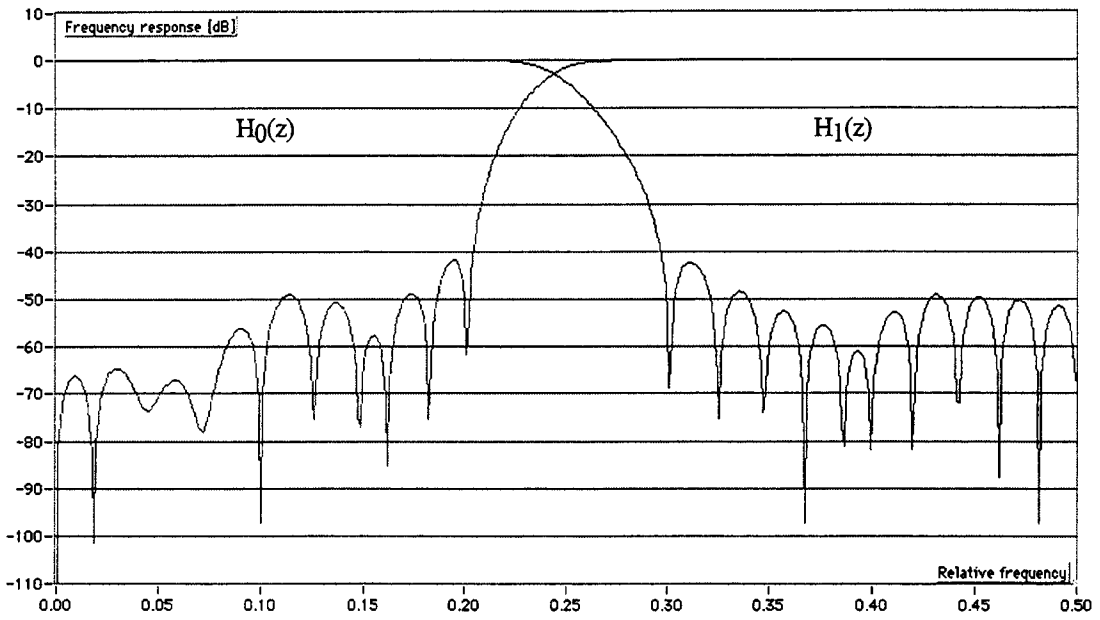


Figure 3.9 Frequency response of a perfect reconstruction filter pair designed using lattice factorization [68]. Both filters have order equal to 64.

D. QMF Filter Bank Design

By giving up the perfect reconstruction constraint one can design filter banks that offer some useful properties. In the case of the QMF filter bank, the whole structure is defined by a single prototype FIR filter while a small reconstruction error is involved. Since for the QMF filter bank $H_1(z) = H_0(-z)$, the biorthogonality constraint (3.26) in the case of even-order FIR filters with linear phase, reduces to [86]

$$\left|H_0(e^{j\omega})\right|^2 + \left|H_0\left(e^{j(\omega+\pi)}\right)\right|^2 = 1. \tag{3.35}$$

As mentioned earlier, there are no FIR filters with linear phase that satisfy the above constraint. Only approximate solutions can be found using nonlinear programming techniques. The QMF filter design in [87] uses the following objective function

$$\Phi = \alpha \int_0^{\pi} \left[1 - \left|H_0(e^{j\omega})\right|^2 - \left|H_0\left(e^{j(\omega+\pi)}\right)\right|^2\right]^2 d\omega + \int_{\omega_s}^{\pi} \left|H_0(e^{j\omega})\right|^2 d\omega \tag{3.36}$$

were the factor α determines the relative importance of the two terms: the reconstruction error, and the stopband attenuation error. Figure 3.10 illustrates a 64-tap analyzing filter

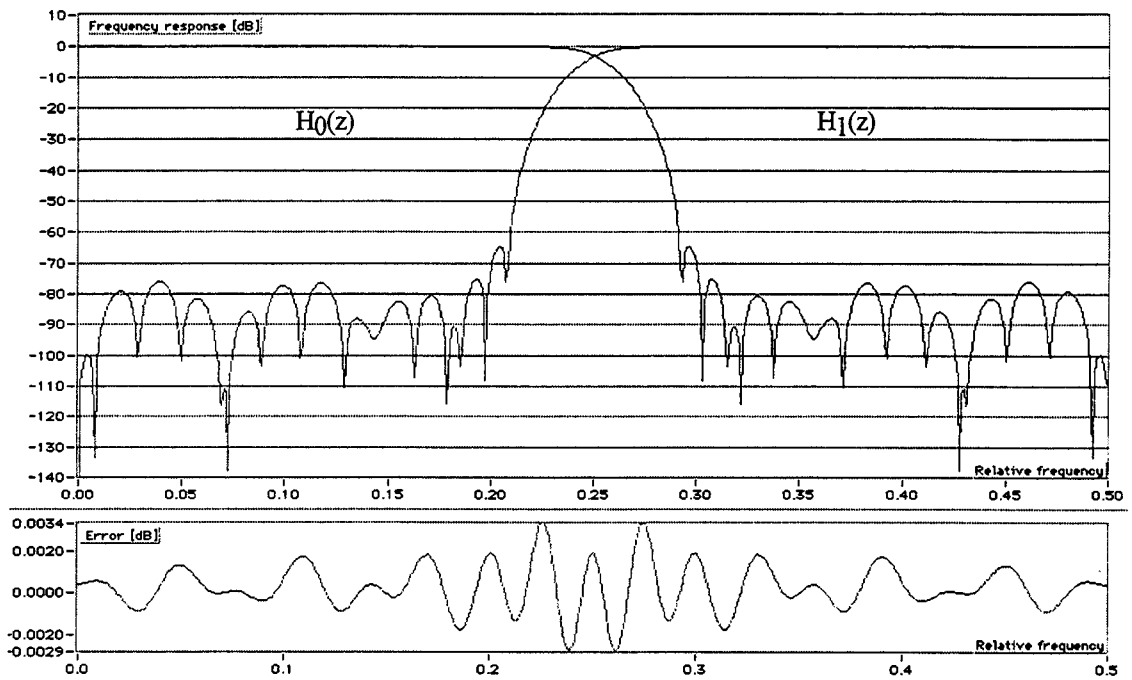
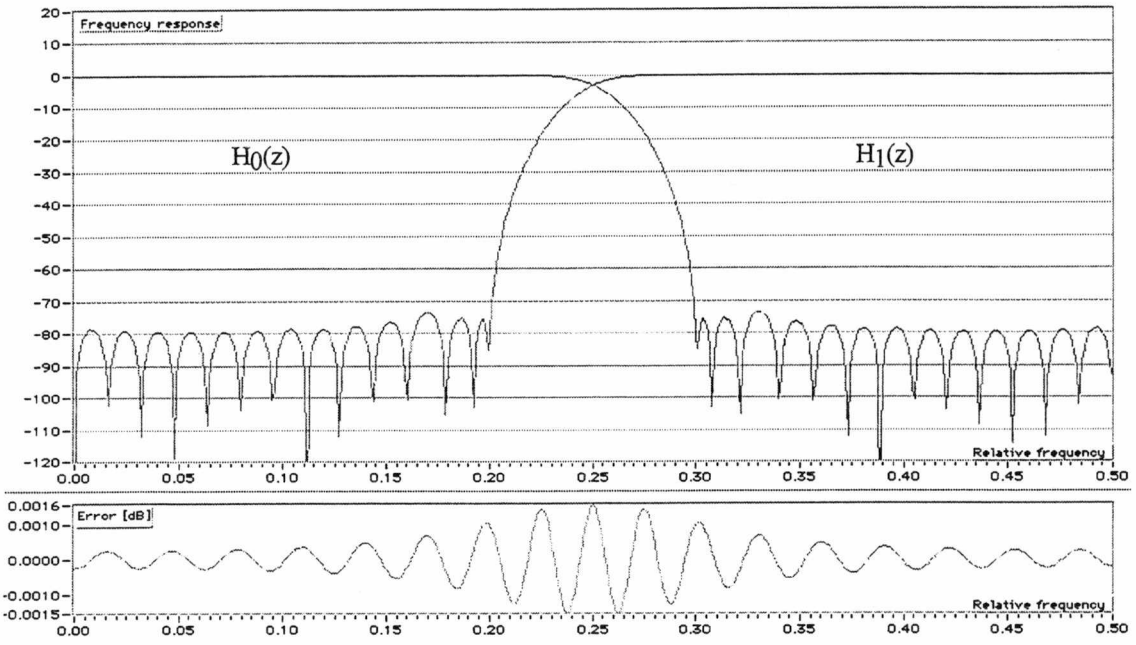


Figure 3.10 Frequency response of a 64-tap near-perfect reconstruction filter pair designed by Johnston's method [87].

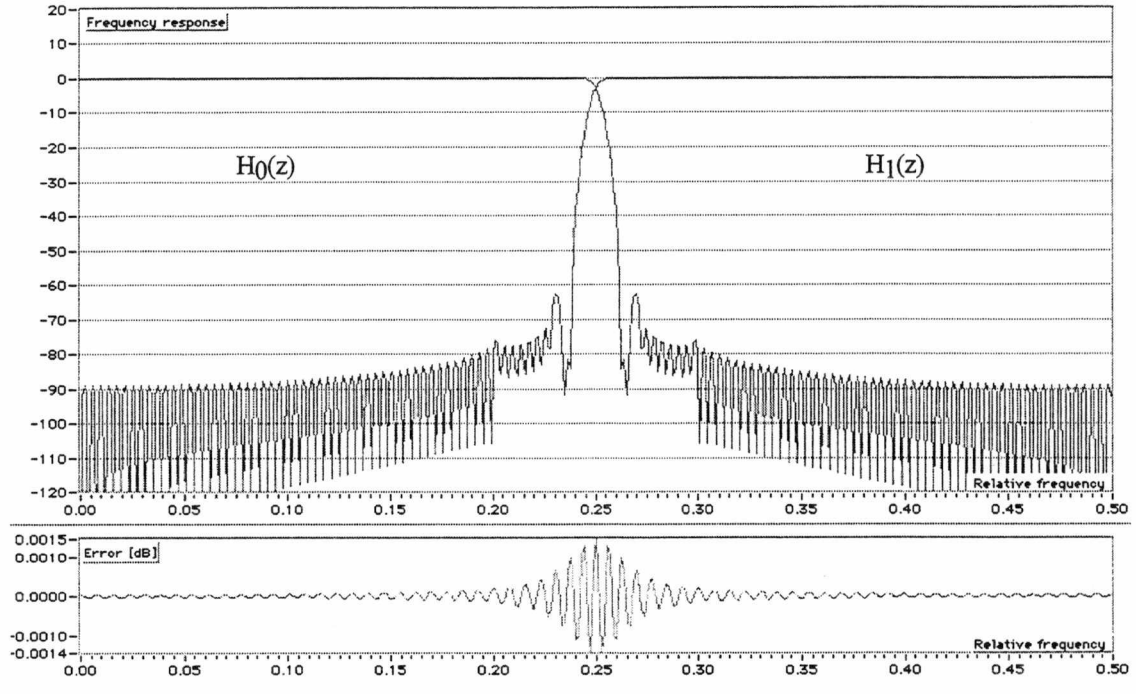
pair designed by Johnston's method [87] together with the reconstruction error defined as

$$\varepsilon(\omega) = 10\log\left(\left|H_0(e^{j\omega})\right|^2 + \left|H_0(e^{j(\omega+\pi)})\right|^2\right) \tag{3.37}$$

All the methods that use some kind of nonlinear optimization technique suffer from divergence if longer windows need to be designed. However, the longer windows are necessary in applications where good selectivity in the spectral domain is needed. In this dissertation a new class of high-quality FIR filter banks with linear phase was designed. These filter banks are later used in tree-structured filter banks for music signal segmentation. The new design, described in Appendix A in more detail, relies on an iterative procedure where the reconstruction error (3.37) is reduced in each step of the procedure. As a starting point, a lowpass FIR filter designed by a standard FIR filter design is used. The method converges fast and designed filters may have an order in the thousands thus having good selectivity. As an example, Figure 3.11 shows two filters



(a)



(b)

Figure 3.11 Frequency responses and the reconstruction errors for filters designed by the new method. Order of the filters are 64 (a) and 256 (b).

designed by this original method. Comparing these filters with all the filters illustrated in this section, it can be seen that they offer much better selectivity paid for by an acceptably small error. This error is analyzed in more detail in the next chapter where these filters are used in a multilevel structure for music signal processing.

3.4 Tree-Structured Filter Banks and Wavelets

Tree-structured filter banks are obtained by combining a two-channel filter bank in a multilevel structure. There are three major groups of tree-structured filter banks.

1. The full-tree filter banks that yield a linear division of the spectrum similar to the STFT.
2. The octave-band filter banks that perform discrete-time wavelet transform.
3. General structure filter banks that have arbitrarily connected two-channel filter banks. They lead to a family of orthonormal bases known as wavelet packets.

A. Full-Tree Filter Banks

As mentioned above, full-tree filter banks split the signal spectrum in constant-bandwidth bands thus performing a variable-Q spectral analysis. The Q-factor depends linearly on the analyzing frequency, the same way as in the STFT. Figure 3.12 (a) illustrates a three-stage, eight-channel full-tree filter bank. Using the multirate identities [66], the filter bank can be redrawn as shown in Figure 3.12 (b). Each analysis channel can now be considered as a bandpass filter followed by a downsampler of factor 8. The bandpass filters in an M-stage full-tree bank are defined by

$$H^{(k)}(z) = \prod_{i=1}^{M} H_l(z^{2^i}), \quad k = 0,1,...,2^M - 1, \; l = 0 \; or \; 1. \tag{3.38}$$

If both $H_0(z)$ and $H_1(z)$ are of equal length L, then the length of each of the bandpass filters (3.38) is equal to

$$L_M = (2^M - 1)(L-1) + 1. \tag{3.39}$$

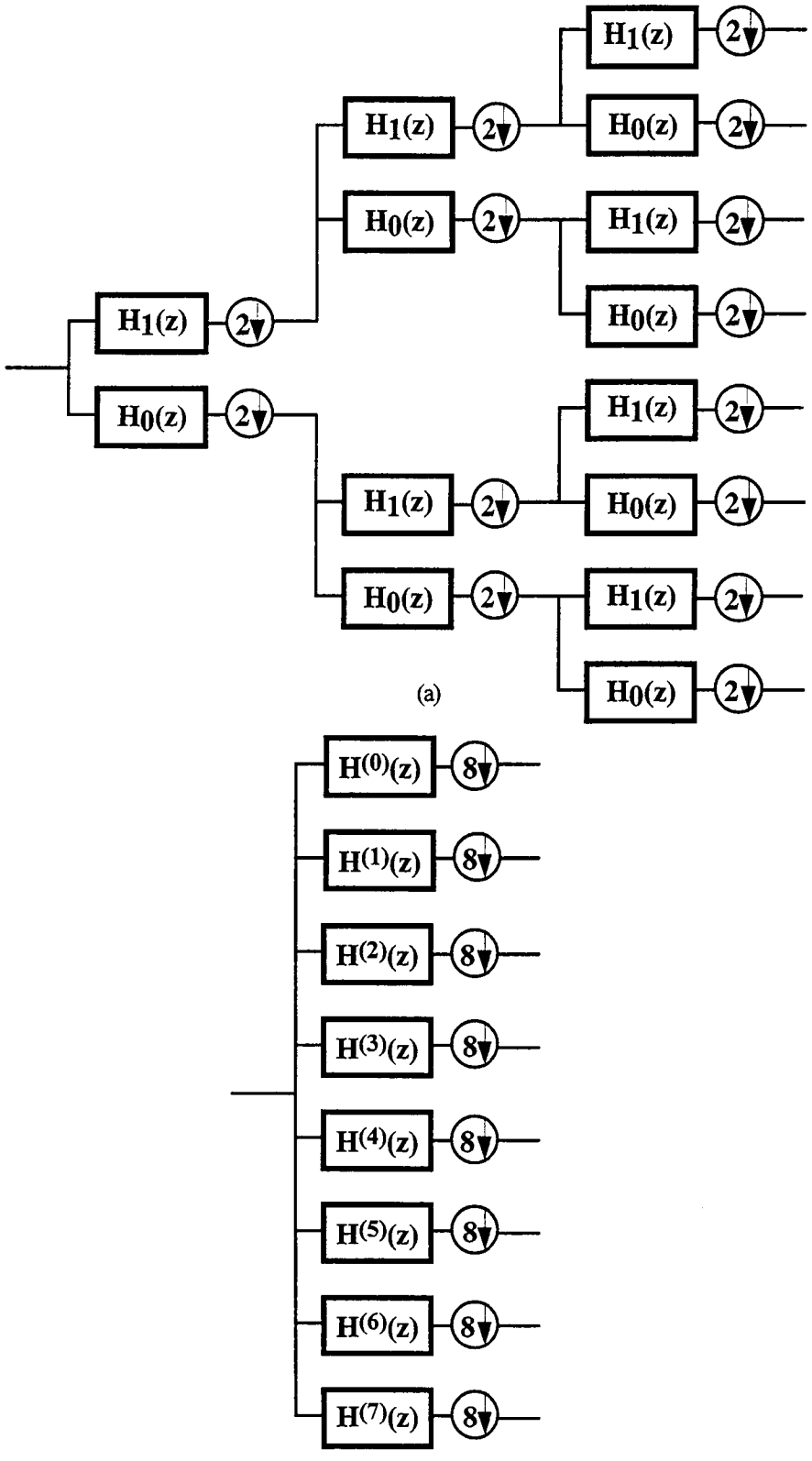


Figure 3.12 Eight-channel full-tree filter bank (a) and the corresponding eight-channel structure (b).

If the filter pair $\{H_0(z), H_1(z)\}$ is a perfect reconstruction filter pair, then the whole full-tree structure has the perfect reconstruction property. If QMFs are used instead, then the analysis subbands can have good selectivity and linear phase, but there will be some reconstruction error involved.

B. Octave-Band Filter Banks

While the full-tree filter banks correspond to the discrete-time STFT, the octave-band filter banks lead to the discrete-time WT. This time, the signal spectrum is divided into octave bands thus keeping the Q-factor of the analysis constant. Note however, that the Q-factor in this case is equal to 1.5 leading to a low resolution signal analysis. A three-level WT can be performed using a three-stage octave-band filter bank shown in Figure 3.2 (a). Again, using the multiresolution identities, the above structure can be redrawn as shown in Figure 3.13.

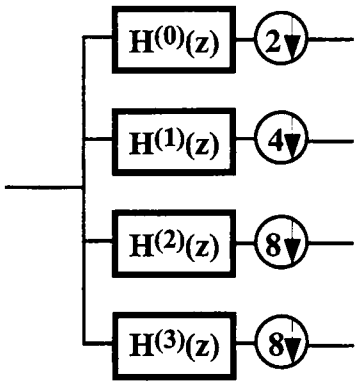


Figure 3.13 Four-channel filter bank equivalent to the filter bank in Figure 3.2 (a).

Each channel in the structure has bandwidth half the bandwidth of its upper neighbor, except for the lowest channel. In an M-level WT, each of M+1 subbands can be represented by

$$\begin{aligned} H^{(0)}(z) &= H_1(z) \\ H^{(k)}(z) &= H_1(z^{2^k}) \prod_{i=0}^{k-1} H_0(z^{2^i}),\ k = 1,\ldots,M-1 \\ H^{(M)}(z) &= \prod_{i=0}^{M-1} H_0(z^{2^i}). \end{aligned} \tag{3.40}$$

Note that the length of each of the bandpass filters from Figure 3.13 is different and is given by

$$L^{(k)} = (2^{k+1} - 1)(L-1) + 1, \; k = 0,...,M-1$$
$$L^{(M)} = L^{(M-1)}. \tag{3.41}$$

Also, as the consequence of different subsampling after each of the subband filters, the sampling rate is different in each of the channels.

C. General Tree-Structured Filter Banks

Using an arbitrary branching of the tree-structured filter banks one can get different splits of the signal spectrum. If the same branching is used in the synthesis part, then an orthogonal structure can be obtained. An example in Figure 3.14 shows all possible two-stage general structures.

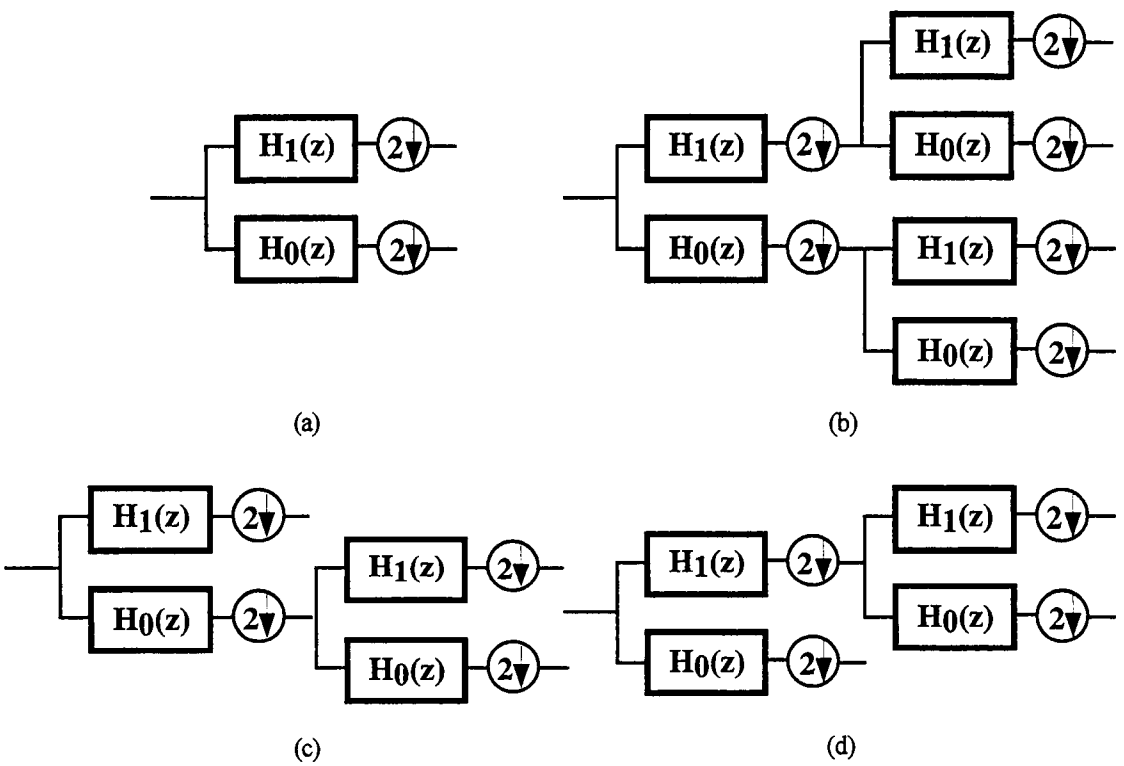


Figure 3.14 All possible general tree-structured filter banks in a two-stage structure. (a) Two-band split. (b) Full-tree bank (STFT like). (c) Octave-band bank (Wavelets). (d) Upper-split octave-band bank.

As Figure 3.14 shows, the STFT and the wavelet trees are only two of all possible forms of the general-tree structures. As mentioned earlier, the general tree-structured filter banks lead to a set of orthogonal bases called wavelet packets [90]. The potential of wavelet packets lies in a possibility to select the best basis according to a particular criterion. In the general case, the number N_M of all possible bases for an M-stage structure including the no-split case, is given by the recursive formula [64]

$$N_M = N_{M-1}^2 + 1, \quad M \in \mathcal{N}, N_0 = 1. \tag{3.42}$$

For example, a 5-stage tree has 458,330 different bases that are all orthogonal if the filters used in the tree are perfect reconstruction filters. This huge number of bases offers the possibility to select a basis that best fits the particular application. This leads to an idea to use the general tree-structured filter banks for music signal segmentation. One can select a tree that has constant or nearly-constant Q-factor over the entire signal spectrum. The Q-factor can be arbitrarily increased using "longer" trees, i.e., trees with more stages. Construction, properties, and implementation of such filter banks are subjects of the next chapter.

CHAPTER 4

FILTER BANKS FOR MUSIC SIGNAL SEGMENTATION

4.1 Introduction

This chapter presents tree-structured filter banks designed for use in polyphonic music segmentation. In general, these filter banks transform a polyphonic music signal into a two-dimensional space where all necessary features of the signal are easier to extract.

Tree-structured filter banks allow selection among a large number of possible structures thus achieving the best segmentation in both time and frequency. The best segmentation means obtaining sufficient time-frequency resolution over the entire time and frequency space while possibly retaining the orthogonality of the transform. As shown in Chapter 2 and Chapter 3 there is a trade-off between time-frequency resolution and the orthogonality of the transform. In the case of the STFT there is no good time-frequency analysis with orthogonal bases. Better time-frequency resolution can be achieved using the DWT. However, the resolution in this case is limited by a relatively low Q-factor. General tree-structured filter banks offer an arbitrary split of the time-frequency space and are good candidates for music signal segmentation.

At the beginning of the chapter, basic physical characteristics of music signals are presented. Then, the design and characteristics of tree-structured filter banks for music signal analysis are described. Orthogonality and the time-frequency resolution of these filter banks are analyzed in more detail. Afterwards, implementation issues of general tree-structured filter banks are addressed concentrating on the computational complexity of the structure. At the end, tree-structured filter banks are compared with other front-end processing methods.

4.2 Physical Characteristics of Music Signals

Music sounds can be considered as the human perception of a change in the air pressure produced by a vibrating object [91]. The vibrating object may be a string on a violin, a column of air set in motion by a reed or mouthpiece, or human vocal cords. It can also be acoustic equipment used for the playback of recorded music.

There are many characteristics of music signals used for the description of a particular music piece. Some of them are nonmathematical in their nature as mentioned in Chapter 1, and cannot be quantitatively measured. Here are considered only measurable characteristics needed to describe a music piece by MIDI parameters.

A. Pitch

Music signals are in most cases periodic signals with a fundamental frequency and number of different harmonics called overtones. The pitch of a music sound quantitatively describes the fundamental frequency of a single music note. In musical terms, if two instruments play at the same frequency, they sound at the same pitch or in unison. However, pitch is a perceptual category and cannot be directly related to the fundamental frequency. For example, a set of frequencies {200 Hz, 300 Hz, 400 Hz} will be perceived as a 100 Hz-pitch sound even though the fundamental frequency is missing. This leads to the conclusion that the human ear senses pitch as the spacing between overtones rather than as the fundamental frequency.

The difference between two pitches is called an interval. If the upper pitch is exactly twice that of the lower pitch, the interval is then called an octave. In Western music, the octave is divided logarithmically into twelve intervals. Rather than specifying frequencies, musicians use letter names (notes) to indicate pitches within an octave. The fundamental frequency of a pitch or a note can be determined by the following formula

$$f_i = f_A 2^{i/12}, \quad i \in \mathbb{Z} \tag{4.1}$$

where $f_A = 440Hz$ represents the reference frequency, and i denotes the distance in intervals between the particular note and the note middle A. Therefore any system for

music signal analysis needs to have enough resolution to separate two adjacent notes, i.e., the minimal Q-factor of the analysis needs to be

$$\begin{aligned} Q_{\min} &= \frac{f_i}{f_{i+1} - f_i} \\ &= \frac{1}{2^{1/12} - 1} \\ &= 16.72. \end{aligned} \tag{4.2}$$

However, the above defined Q-factor is sufficient only for analysis within an octave. If the analysis needs to cover more than one octave, then the resolution needs to be even better. In a polyphonic music piece that covers more than one octave there is a possibility of collision between the overtones from different octaves. As an illustration, Table 4.1 (a) shows the fundamental frequencies of music notes starting from C_1 and covering five octaves. On the other hand, Table 4.1 (b) shows one octave starting from A_3 together with four overtones for each of the notes.

Table 4.1 Note frequencies. (a) Fundamental frequencies starting from C_1 covering five octaves. (b) Fundamental frequency and four overtones covering an octave starting with A_3.

C	C#	D	D#	E	F	F#	G	G#	A	A#	B	
65.41	69.30	73.42	77.78	82.41	87.31	92.50	98.00	103.83	110.00	116.54	123.47	1
130.81	138.59	146.83	155.56	164.81	174.61	185.00	196.00	207.65	220.00	233.08	246.94	2
261.62	277.18	293.66	311.13	329.63	349.23	369.99	391.99	415.30	440.00	466.16	493.88	3
523.25	554.36	587.33	622.25	659.25	698.45	739.99	783.99	830.61	880.00	932.33	987.76	4
1046.50	1108.73	1174.66	1244.51	1318.51	1396.91	1479.97	1567.98	1661.22	1760.00	1864.65	1975.53	5

(a)

A3	A#3	B3	C4	C#4	D4	D#4	E4	F4	F#4	G4	G#4	
440.00	466.16	493.88	523.25	554.37	587.33	622.25	659.26	698.46	739.99	783.99	830.61	F0
880.00	932.33	987.77	1046.50	1108.73	1174.66	1244.51	1318.51	1396.91	1479.98	1567.98	1661.22	01
1320.00	1398.49	1481.65	1569.75	1663.10	1761.99	1866.76	1977.77	2095.37	2219.97	2351.97	2491.83	02
1760.00	1864.66	1975.53	2093.00	2217.46	2349.32	2489.02	2637.02	2793.83	2959.96	3135.96	3322.44	03
2200.00	2330.82	2469.42	2616.26	2771.83	2936.65	3111.27	3296.28	3492.28	3699.94	3919.95	4153.05	04

(b)

As the table shows, the first overtone of A_3 overlaps with the fundamental of A_4, and the fundamental of E_5 is very close to the second overtone of A_3. In practice, whenever a collision occurs, note detection may be made using the remaining nonoverlaping overtones.

B. Timbre

Timbre determines the *color* of a note. It allows one to distinguish between two sustained sounds of the same pitch, i.e., between two instruments playing the same note. Even the timbre of a single instrument may change depending on how the instrument is played. For example, the same pitch played on two different strings of the same violin will have a slightly different timbre. In the physical sense, timbre is determined by the distribution of amplitudes of the fundamental frequency and the overtones of a particular note.

C. Amplitude and Loudness

The term amplitude in music is used in the same way as in signal processing. Loudness is related to the perception of a sound amplitude. It depends on both the amplitude and the frequency of a note. The human ear is most sensitive to frequencies in the range 2,500 Hz to 5,000 Hz. The lower and the higher frequencies having the same amplitudes, have lower loudness. While relatively insensitive to the phase changes in music sounds, the human ear is very sensitive to the amplitude change. It is a logarithmic receiver with dynamic range of about 120 dB.

During the production of a particular note, the note amplitude varies with time. This amplitude change is also called the envelope of the note. Obviously, each of the overtones has its own envelope that is, in general, different from the envelopes of other overtones. The note amplitude can be broken into several parts along the time axis. The attack is the first part where the signal envelope starts from zero and reaches its maximum level. What follows is the sustained part where the envelope stays nearly constant. Some instruments do not have sustained part, i.e., the envelope decays immediately after

the attack. If the sound is stopped before it has died away completely, it enters the release part until it eventually stops sounding. For example, the amplitude envelope of a piano sound has a fast attack, no sustained part, relatively slow decay, and a fast release. The organ sound however, has a slightly longer attack, a sustained part that lasts as long as the performer holds down the key, and a slightly longer release.

4.3 Tree-Structured Filter Banks for Music Signal Analysis

Presented below is a procedure for the design of tree-structured filter banks. The objective is to design a structure, possibly an orthogonal one, that has constant or nearly-constant frequency resolution over several octaves. The frequency resolution needs to have a lower bound defined by the minimal Q-factor as in (4.2). Also, the structure needs to be based on a two-channel filter bank that has a minimum number of taps for a given selectivity. Besides other characteristics, special attention in this section is made on the selectivity and the orthogonality of the proposed filter bank. At the end, the implementation issues are discussed, and the proposed structure is compared with other front-end signal processing methods.

4.3.1 Selection of the Tree-Structure

If a full-tree structure with M stages as in Figure 3.12 is used for music signal analysis, then only $R_M < 2^M$ upper subbands will have a Q-factor better than $Q_{\min}$ defined in (4.2). R_M is given by

$$\begin{aligned} R_M &= 2^M - \lceil Q_{\min} + 0.5 \rceil \\ &= 2^M - 18 \end{aligned} \tag{4.3}$$

where $\lceil x \rceil$ denotes the smallest integer greater than or equal to x. From (4.3), the number of stages is $M \geq 5$. If $M = 5$, then $R_M = 14$, i.e., only 14 upper of a total 32 bands will have sufficient resolution. In other words, only frequencies above

$$f_R = f_s \frac{\lceil Q_{\min} + 0.5 \rceil}{2^{M+1}} = 0.28125 f_s \tag{4.4}$$

where f_s is the sampling frequency, will be analyzed with sufficient resolution. Note that the topmost subband in the above case has a Q-factor equal to $2^M - 0.5 = 31.5$ which is much above $Q_{\min}$ defined in (4.2). To make f_R smaller, i.e., to improve the resolution in the lower bands, one needs to increase M according to (4.4). However, the highest frequencies will be then analyzed with too much resolution making the algorithm inefficient.

The above discussion shows that the full-tree structures cannot be used for music signal analysis in an efficient way. As a reasonable alternative, a modified full-tree structure can be used instead. The full-tree structure is used only to give sufficient resolution in the upper half of the signal spectrum. The minimum number of full-tree stages that give sufficient resolution in the upper half of the signal spectrum is determined by the inequality

$$R_M = 2^M - \lceil Q_{\min} + 0.5 \rceil \geq 2^M / 2 \tag{4.5}$$

and is equal to

$$M_{\min} = \lceil \log_2 \lceil Q_{\min} + 0.5 \rceil \rceil + 1 = 6. \tag{4.6}$$

Note that the upper half of the signal spectrum is then analyzed with a Q-factor of at least 31.5 providing some safety margin. After the upper half of the signal spectrum is analyzed with sufficient resolution, only the lower half is processed further splitting $2^{M_{\min}-1}$ subbands into new $2^{M_{\min}}$ subbands. That way, the new upper $2^{M_{\min}-1}$ subbands will have sufficient resolution, i.e., Q-factor is at least 31.5. The lower $2^{M_{\min}-1}$ subbands can be further split in their halfbands. The process of splitting the lower group of subbands is performed until sufficient resolution is reached for the lowest frequency in the signal spectrum.

If the number of the half-band splitting stages is denoted by I, then the minimum

frequency analyzed with at least $Q_{\min}$, is given by

$$f_R = f_s \frac{\lceil Q_{\min} + 0.5 \rceil}{2^{M_{\min}+I+1}}. \tag{4.7}$$

If the minimum frequency, f_R, is given, then the minimum number of additional stages, $I_{\min}$, is given by

$$I_{\min} = \left\lceil \log_2 \left(\frac{f_s}{f_R} \lceil Q_{\min} + 0.5 \rceil \right) \right\rceil - M_{\min} - 1. \tag{4.8}$$

EXAMPLE 4.1

If a music signal is sampled at 24,000 Hz, and sufficient resolution defined by $Q_{\min} = 16.72$ needs to be achieved above the note C_2 ($f_{C_2} = 130.81 Hz$), then the minimum number of the additional stages according to (4.8), is $I_{\min} = 5$. Note that the signal frequency band from 130.81 Hz to 12,000 Hz covers more than six octaves.

There exists a phenomenon inherent to the tree-structured filter banks called *frequency shuffling* that needs to be taken into consideration. Frequency shuffling occurs as a result of downsampling the upper band signal in a two-channel structure of Figure 3.3. After downsampling, not only will the upper band be shifted into the lower frequency band, but it will also be flipped over. As a result, the frequency axis is inverted. Consequently, in a full-tree filter bank, the output subbands will not be ordered in increasing order with respect to frequency, but rather shuffled. The entries in Table 4.2 show the actual frequency indices for each of the subbands in a 6-stage, full-tree filter bank where lower indices correspond to lower frequencies. To place the subbands in increasing order with respect to frequency, a reshuffling procedure that uses Table 4.2 needs to follow the full-tree filter bank.

Frequency shuffling also occurs after each stage of the half-band splitting that follows the full-tree structure in the tree-structured filter bank described above. Table 4.3 shows the actual frequency indices of the 64-subband output after each half-band

Table 4.2 Actual frequency indices of the outputs of the 6-stage full-tree filter bank.

OUTPUT INDICES	ACTUAL INDICES							
0-7	**0**	**1**	**3**	**2**	**6**	**7**	**5**	**4**
8-15	**12**	**13**	**15**	**14**	**10**	**11**	**9**	**8**
16-23	**24**	**25**	**27**	**26**	**30**	**31**	**29**	**28**
24-31	**20**	**21**	**23**	**22**	**18**	**19**	**17**	**16**
32-39	**48**	**49**	**51**	**50**	**54**	**55**	**53**	**52**
40-47	**60**	**61**	**63**	**62**	**58**	**59**	**57**	**56**
48-55	**40**	**41**	**43**	**42**	**46**	**47**	**45**	**44**
56-63	**36**	**37**	**39**	**38**	**34**	**35**	**33**	**32**

Table 4.3 Actual frequency indices of the outputs of the half-band splitting stage.

OUTPUT INDICES	ACTUAL INDICES							
0-7	**0**	**1**	**3**	**2**	**4**	**5**	**7**	**6**
8-15	**8**	**9**	**11**	**10**	**12**	**13**	**15**	**14**
16-23	**16**	**17**	**19**	**18**	**20**	**21**	**23**	**22**
24-31	**24**	**25**	**27**	**26**	**28**	**29**	**31**	**30**
32-39	**32**	**33**	**35**	**34**	**36**	**37**	**39**	**38**
40-47	**40**	**41**	**43**	**42**	**44**	**45**	**47**	**46**
48-55	**48**	**49**	**51**	**50**	**52**	**53**	**55**	**54**
56-63	**56**	**57**	**59**	**58**	**60**	**61**	**63**	**62**

splitting stage. Therefore, a similar reshuffling procedure needs to be applied after each of the half-band splitting stage.

Based on the above discussion, a complete tree-structured filter bank with $M = 6$ full-tree and $I = 6$ half-band stages is illustrated in Figure 4.1. The number of output channels in the analysis part of the filter bank in Figure 4.1 (a), is given by

$$N_{SB} = 2^{M-1}(I+2) = 224. \tag{4.9}$$

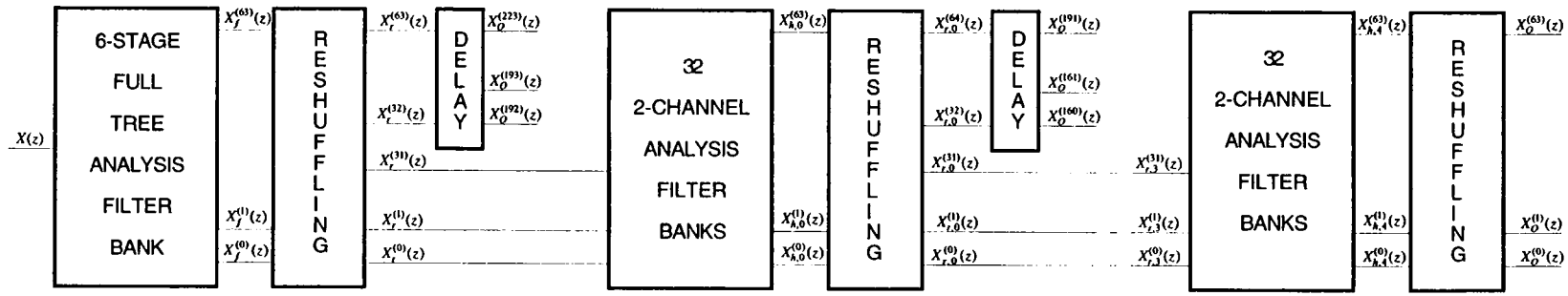
An important issue related to the tree-structured filter banks is the signal delay through the structure. The single analysis filter pair causes a delay of (L-1)/2 taps, so that the input signal processed by the full-tree part of the structure in Figure 4.1 is delayed by $M_{\min}(L-1)/2$ taps. After each half-band splitting stage, an $(L-1)/2$-tap delay is added. Thus, the total delay in the lowest 64 bands is $(M_{\min}+I_{\min})(L-1)/2$ taps, or 58.4 ms if $L = 256$ and the sampling frequency is 24 kHz.

Special care needs to be taken to line up in time all 224 outputs of the structure in Figure 4.1. Note that every group of 32 outputs lasts an equal amount of time even though they have different sampling rates. Therefore, if the output data from different output groups needs to be analyzed in time, the following needs to be kept in mind: each group of 32 outputs is delayed by (L-1)/2 with respect to the next group above, and every sample from a group corresponds to two samples of the next group above.

So far, the selection of the best tree-structure was made solely on the minimum resolution criteria. The described filter bank has an almost constant Q-factor over the entire signal frequency band. What follows is an analysis of how the filters used in the structure affect its selectivity and orthogonality.

4.3.2 Orthogonality

Equivalently to Figure 3.12 (b), the analysis part of the tree-structured filter bank from Figure 4.1 can be transformed using the multiresolution identities as given in Figure 4.2. Equivalenly to (3.38), each of 224 bandpass filters from the figure has the



$X(z)$ - Input

$X_f^{(j)}(z), j = 0,...,63$ - Output of the full-tree analysis filter bank

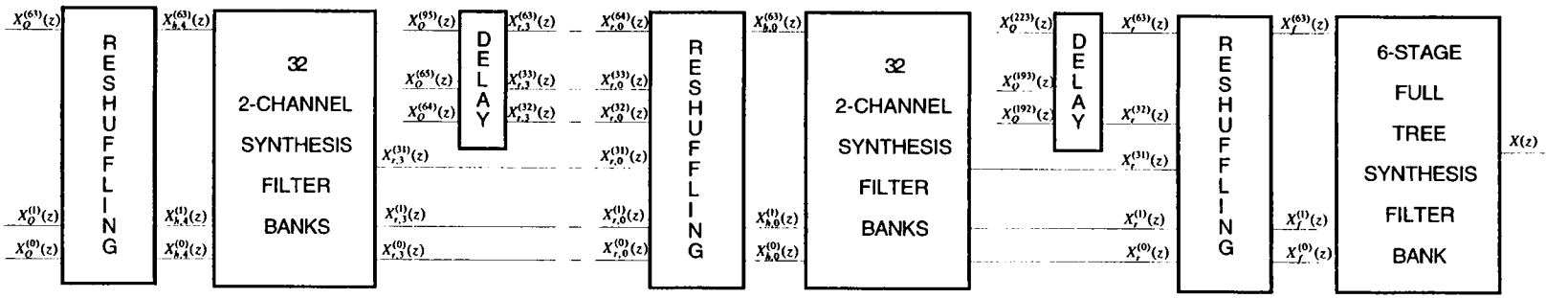
$X_r^{(j)}(z), j = 0,...,63$ - Reshuffled output of the full-tree analysis filter bank

$X_{r,i}^{(j)}(z), j = 0,...,63, i = 0,1,2,3$ - Reshuffled output of the half-band analysis filter bank

$X_{h,i}^{(j)}(z), j = 0,...,63, i = 0,1,2,3,4$ - Output of the half-band analysis filter bank

$X_Q^{(j)}(z), j = 0,...,223$ - Output of the tree-structured analysis filter bank

(a)



$X_Q^{(j)}(z), j = 0,...,223$ - Input of the tree-structured analysis filter bank

$X_{h,i}^{(j)}(z), j = 0,...,63, i = 0,1,2,3,4$ - Input of the half-band analysis filter bank

$X_{r,i}^{(j)}(z), j = 0,...,63, i = 0,1,2,3$ - Shuffled input of the half-band analysis filter bank

$X_r^{(j)}(z), j = 0,...,63$ - Shuffled input of the full-tree analysis filter bank

$X_f^{(j)}(z), j = 0,...,63$ - Input of the full-tree analysis filter bank

$X(z)$ - Output

(b)

Figure 4.1 11-stage tree-structured filter bank with 224 subbands. (a) Analysis structure. (b) Synthesis structure.

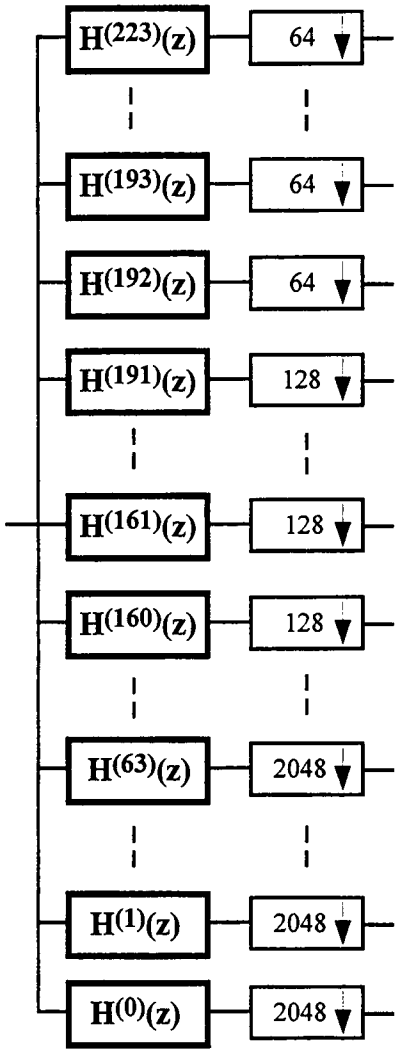


Figure 4.2 Equivalent representation of the tree-structured filter bank of Figure 4.1. There are 224 bandpass filters followed by downsamplers.

response defined as

$$H^{(k)}(z) = \prod_{i=1}^{M_{\min}+I} H_l(z^{2^i}),\ l = 0,1,\ k \in [192-32I,\ 223-32I],\ I = 0,1,\ldots,I_{\min}$$
$$H^{(k)}(z) = \prod_{i=1}^{M_{\min}+I_{\min}} H_l(z^{2^i}),\ l = 0,1,\ k \in [0,\ 31]. \tag{4.10}$$

The filter lengths are given by

$$L^{(k)} = (2^{M_{\min}+I} - 1)(L-1)+1,\ k \in [192-32I,\ 223-32I],\ I = 0,1,\ldots,I_{\min}$$
$$L^{(k)} = (2^{M_{\min}+I_{\min}} - 1)(L-1)+1,\ k \in [0,\ 31] \tag{4.11}$$

where L is the length of the filters used in the structure. For example, if

$M_{\min} = 6$, $I_{\min} = 5$, and $L = 64$, the lowest 64 bandpass filters from Figure 4.2 have an order of 128,960, while the topmost 32 filters have an order of 3,968. Obviously, the longer filters have very narrow frequency response, but very poor time localization.

If N is the number of the input samples, then the total number of the output samples for the structure in Figure 4.2, assuming that circular convolution is used for filtering, is given by

$$\begin{aligned} N_{tot} &= 2^{M_{\min}-1} \sum_{i=0}^{I_{\min}-1} \frac{N}{2^{M_{\min}+i}} + 2^{M_{\min}} \frac{N}{2^{M_{\min}+I_{\min}}} \\ &= N\left(2^{-I_{\min}} + \sum_{i=0}^{I_{\min}-1} 2^{-i-1}\right) \\ &= N, \end{aligned} \tag{4.12}$$

i.e., it is equal to the number of input samples. It means that the tree-structured filter bank from Figure 4.1 is critically sampled thus satisfying a necessary condition for orthogonality for the whole structure. The orthogonality of the tree-structured filter bank considered as a whole in Figure 4.1, is defined below.

PROPOSITION 4.1

If the filter bank $\{H_0(z), H_1(z), G_0(z), G_1(z)\}$ used in a tree-structured filter bank, constitutes a perfect reconstruction two-channel filter bank, then the tree-structured filter bank has the perfect reconstruction property as well, i.e., $\hat{X}(z) = z^{-(M_{\min}+I_{\min})(L-1)} X(z)$, where $X(z)$ denotes the input, and L is the order of the filters.

PROOF

According to Figure 4.1, the last 64 outputs come from 32 analysis filter pairs $\{H_0(z), H_1(z)\}$. Combined with the corresponding 32 synthesis filter pairs $\{G_0(z), G_1(z)\}$, they cancel each other up to a delay equal to L-1 taps. Now, 32 analysis filter pairs from the previous analysis stage cancel 32 corresponding synthesis filter pairs from the next synthesis stage increasing the total delay to $2(L\text{-}1)$. Repeating the same procedure $M_{\min} + I_{\min}$ times, the whole structure reduces to a delay $(M_{\min} + I_{\min})(L-1)$

taps long. The output of the structure is then $\hat{X}(z) = z^{-(M_{\min}+I_{\min})(L-1)}X(z)$.

Obviously, if instead of the perfect reconstruction two-channel filter bank, a QMF filter bank is used, then besides the delay, an amplitude distortion will occur. The question is, how big is the distortion, and can it be reduced to a level acceptable in practical applications. To investigate this, the structure from Figure 4.1 was applied on a test signal, and two kind of error measures were calculated. The first one is the normalized short-time error energy defined as

$$STEE_x(n) = \frac{\sum_{i=Tn}^{Tn+P} \left| x^2(i) - \hat{x}^2(i) \right|}{\sum_{i=Tn}^{Tn+P} x^2(i)}, \quad n = 0,1,\ldots,\left\lceil \frac{N}{T} \right\rceil - 1 \tag{4.13}$$

where $x(n)$ is the N-sample long input signal, $\hat{x}(n)$ is the output signal, T is the window hop, and P is the window width. The second error measure is the normalized total error energy defined as

$$TEE_x = \frac{\sum_{i=0}^{N-1} \left| x^2(i) - \hat{x}^2(i) \right|}{\sum_{i=0}^{N-1} x^2(i)}. \tag{4.14}$$

The input signal in the example was a piano recording 240,000 samples long sampled at 24,000 Hz. The window length was P = 1000 samples, and the hop was T = 500 samples.

Figure 4.3 shows the $STEE_x(n)$ and TEE_x for three different QMF filter pairs given in Figure 3.10 and Figure 3.11, as well as for the perfect reconstruction filter pair from Figure 3.9. As Figure 4.3 (a) shows, the short-time error for the filter pair from Figure 3.10 is less than 55 dB while the TEE_x is equal to 1.337E-6. However, both filters designed by the method proposed in this dissertation have better reconstruction characteristics. For the 64-tap filter, Figure 3.11 (a), the short-time error is more than 61 dB down and the TEE_x is 2.695E-7 (Figure 4.3 (b)). For the 256-tap filter, the short-time

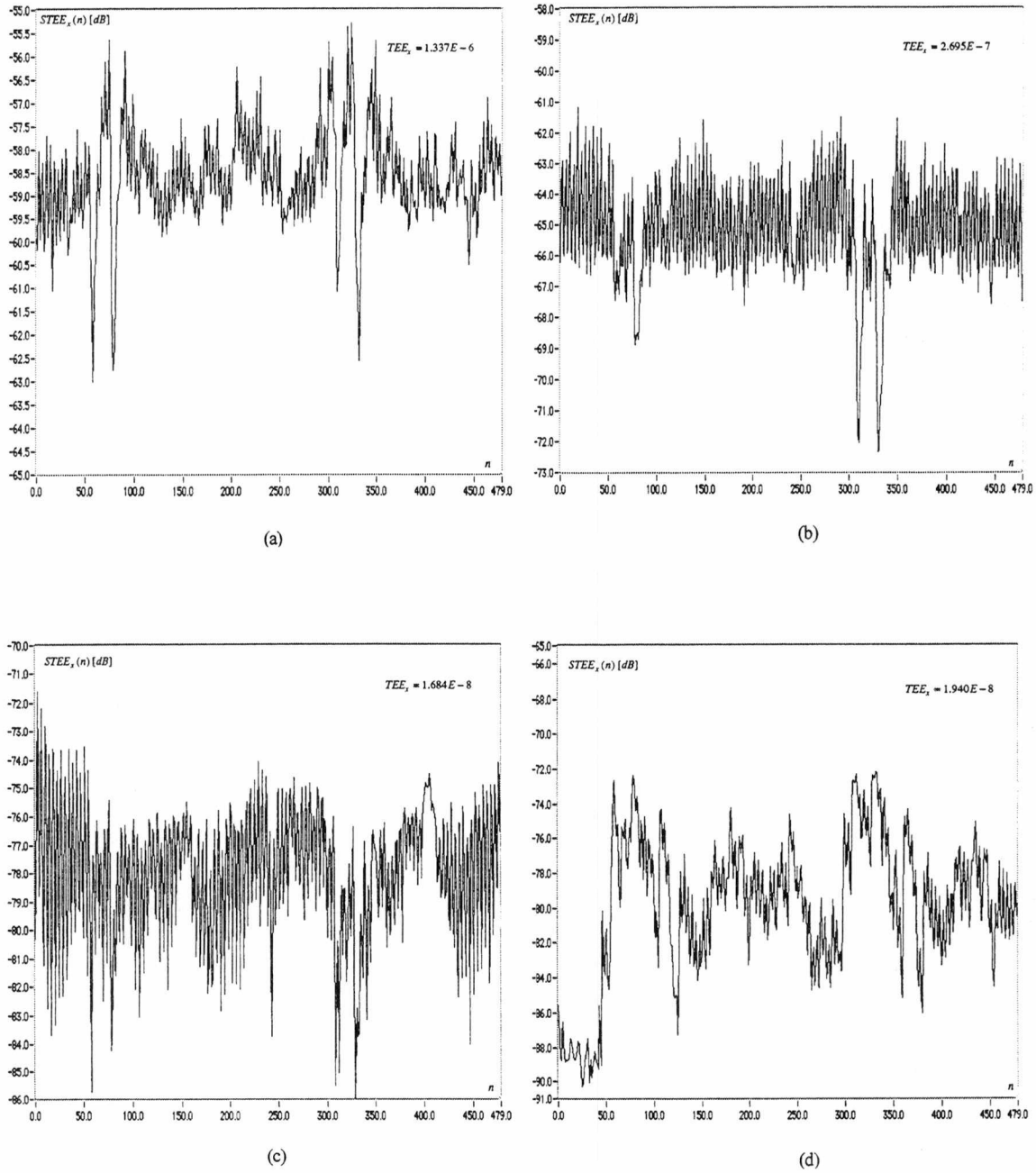


Figure 4.3 The reconstruction error analysis. (a) Johnston's filters. (b) The new design filters with order 64. (c) The new design filters with order 256. (d) Vaidyanathan's perfect reconstruction filters.

error is even smaller due to better stopband attenuation, i.e., it is more than 71 dB down (Figure 4.3 (c)). The total error energy, TEE_x, is 1.684E-8. The short-time error for the perfect reconstruction filter pair from Figure 3.9 is given in Figure 4.3 (d). The error is caused by the finite precision used for the filter taps as well as by the finite arithmetic. It is remarkable that this error is not smaller than the error for the QMF pair from Figure 3.10 (b) (compare Figure 4.3 (c) and Figure 4.3 (d)). This is because the round-off error becomes larger than the reconstruction error for the filter in Figure 3.11 (b).

The above error analysis shows that the tree-structured filter banks that use the QMF pair proposed by the author, have a small reconstruction error that is acceptable in most applications. The reconstruction error of the tree-structured filter banks depends on the reconstruction error of the QMF pair, as well as on the stopband attenuation of those filters.

4.3.3 Time-Frequency Resolution

Time-frequency resolution in the tree-structured filter bank of Figure 4.1 is different in each of the 6 blocks of the bandpass filters. As Figure 4.1 shows, 224 bandpass filters are grouped in 5 blocks each containing 32 equal-length filters. The lowest block has 64 bandpass filters. Therefore, the time-frequency resolution is constant within a particular block and is determined by the total length of the bandpass filters as well as by the time-frequency characteristics of the filter bank $\{H_0(z), H_1(z), G_0(z), G_1(z)\}$. As a qualitative comparison, the time resolution in the topmost block of 32 channels is 32 times better than the time resolution in the lowest 64 channels. At the same time, the frequency resolution in the lowest channels is 32 times better then at the topmost frequencies which is consistent with the requirements for constant Q-factor analysis.

The above discussion suggests that the tree-structured filter bank of Figure 4.1 gives the necessary frequency resolution over the entire frequency band of interest. However, the time resolution is better at higher frequency bands thus suggesting that time events in

music signals can be detected using the output of the topmost channels. As an illustration, Figure 4.4 shows the time-frequency representation of a test signal using the tree-structured filter bank of Figure 4.1. The test signal contains two Dirac pulses placed 30 ms apart as well as two pairs of pure sinusoids with the frequencies 250 Hz, 265 Hz, 4,000 Hz, and 4,240 Hz. As Figure 4.4 shows, the sinusoids are well separated regardless of their location in the frequency space. However, the two Dirac pulses are clearly separated only at the topmost channels.

There is a specific phenomenon that occurs when the tree-structured filter banks are used in frequency segmentation. The phenomenon represents a kind of *frequency leakage* caused by the finite selectivity of the analysis filter pair $\{H_0(z), H_1(z)\}$. As Figures 3.5, 3.6, 3.9, 3.10, and 3.11 show, there is an overlapping region in the frequency responses of the filter pair $\{H_0(z), H_1(z)\}$ placed around $f_s/4$. This causes a leakage of some lowpass

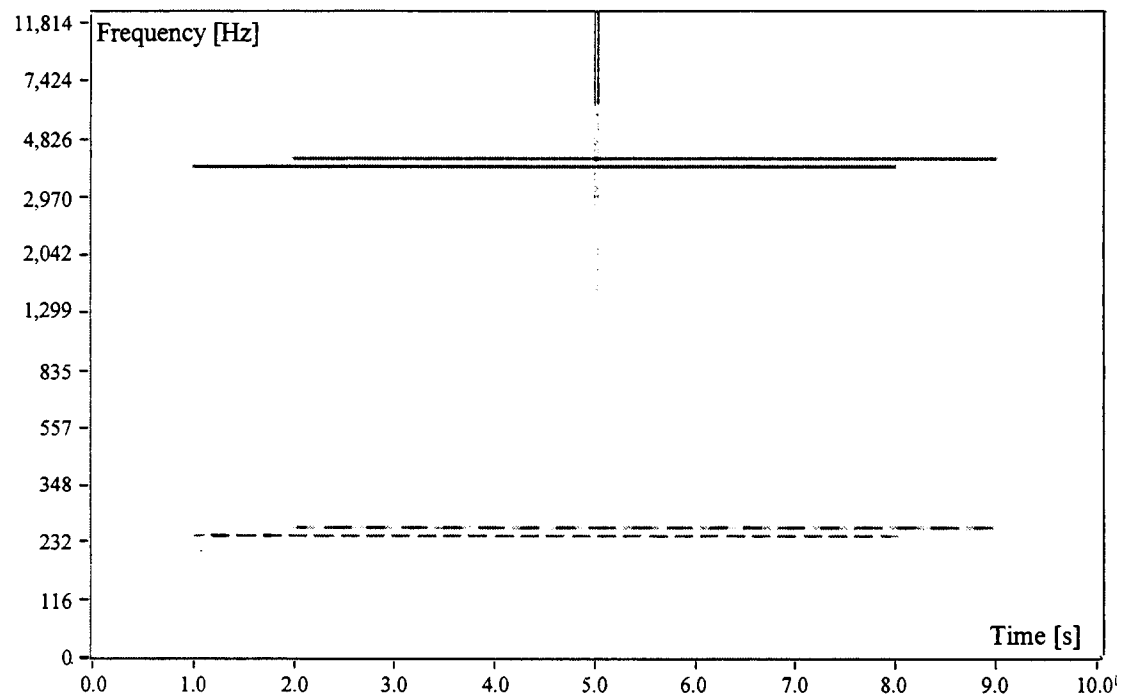


Figure 4.4 Time-frequency representation of four sinusoids with frequencies 250 Hz, 265 Hz, 4,000 Hz, and 4,240 Hz, and two Dirac pulses placed at 5.000 s and 5.030 s, using the tree-structured filter bank of Figure 4.1. The sampling frequency is 24,000 Hz and the total number of samples is 241,664.

energy into the highpass band and vice versa. The leakage repeats in the next stages of the tree-structured filter bank causing a multiple response of a single input frequency. Figure 4.5 illustrates the frequency leakage when the filter pair from Figure 3.5 is used in the tree-structured filter bank. The input signal is a scale of ten different sinusoids with their frequencies in the range from 1000 Hz to 5,500 Hz with increments of 500 Hz.

To reduce the negative effect of the frequency leakage one needs to select a sharp analysis filter pair which can be accomplished using longer filters. Unfortunately, there exist significant difficulties if one needs to design a perfect reconstruction filter pair using the methods proposed so far. All the methods use different kinds of nonlinear optimization techniques that generally suffer from divergence if longer filters (more than 100 taps) need to be designed. Therefore, in this dissertation, a new class of high quality filter pairs is presented and used in the tree-structured filter banks. These filters may easily have an order in the thousands thus having a very narrow transition region and

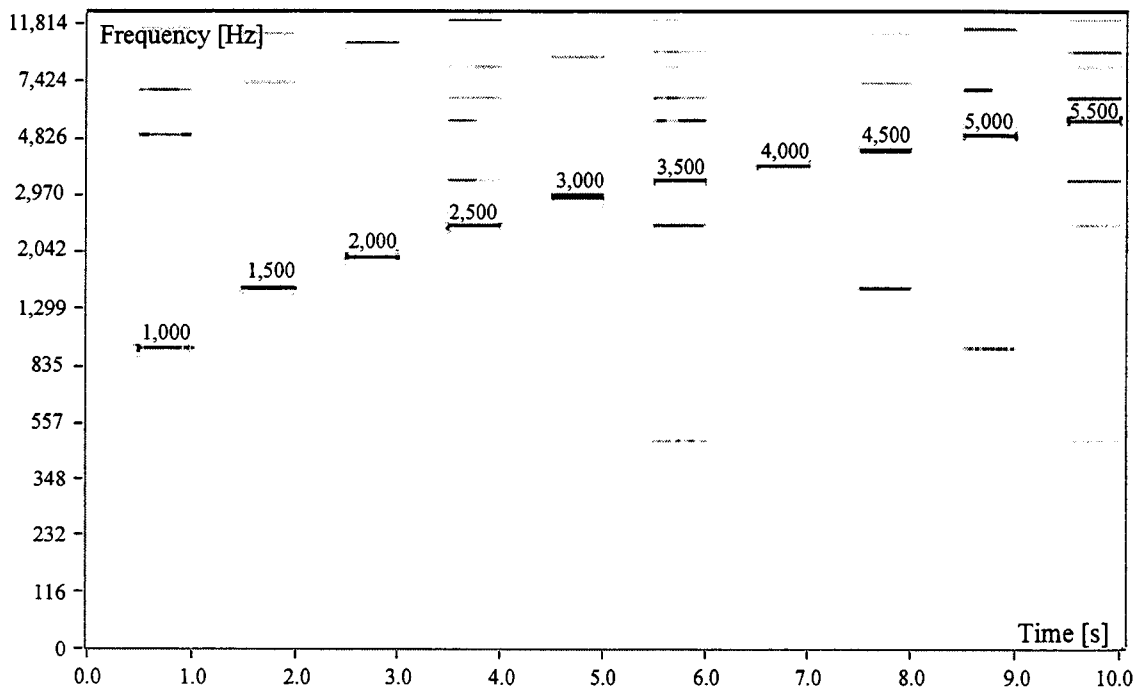


Figure 4.5 Frequency leakage for the filter pair of Figure 3.5. The input signal consists of ten sinusoids with frequencies from 1,000 Hz to 5,500 Hz.

causing a very small frequency leakage. Even though these filters are not perfect reconstruction filters, the error caused by their use is acceptable as shown in the previous subsection. For comparison, Figure 4.6 shows the time-frequency representation of the same input signal from Figure 4.5, but using the filter pair from Figure 3.11 (b) designed by the new method. As the figure shows, the frequency leakage is reduced significantly.

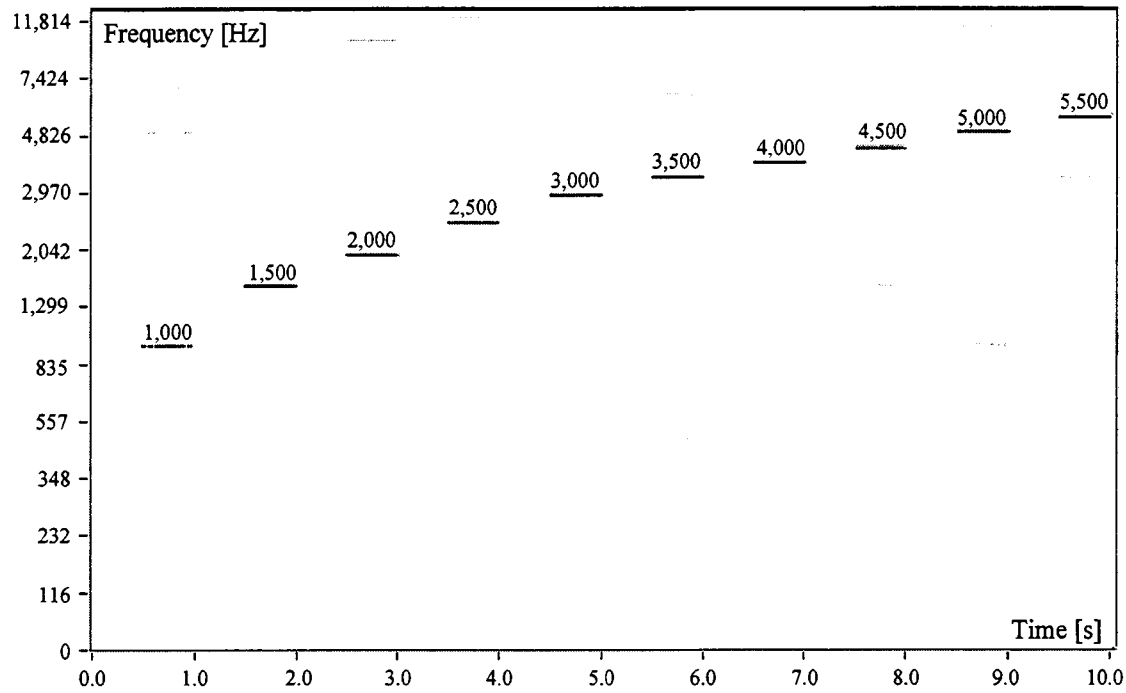


Figure 4.6 Frequency leakage for the new design filter pair of Figure 3.11 (b).

4.3.4 Computational Complexity

Signal processing using tree-structured filter banks seems to be computationally intensive since a structure with $M+I$ stages contains $2^{M-1}(I+5)-2$ filters. However, there exist fast algorithms that can be applied to these filter banks thereby significantly reducing the computational complexity. The discussion that follows concentrates only on the analysis part of tree-structured filter banks since the same procedure can be applied to the synthesis part.

The analysis part of tree-structured filter banks consists of the analysis cells, the filters $H_0(z)$ and $H_1(z)$, followed by downsamplers. In the literature, computational complexity is usually defined as the number of multiplications and additions per input sample. Using time convolution, the analysis cell has complexity

$$C = 2L \; mults/sample + 2(L-1) \; adds/sample \tag{4.15}$$

where L is the order of the filters assumed here to be the same for both filters. The M-stage DWT using the octave-band filter bank of Figure 3.2 (a), then has the complexity [92]

$$C_M = 4L(1-2^{-M}) \; mults/sample + 4(L-1)(1-2^{-M}) \; adds/sample. \tag{4.16}$$

A remarkable fact is that the complexity is bounded as the number of stages increases, i.e., it never exceeds the complexity of a $4L$-order filter.

For the M-stage full-tree filter bank as one in Figure 3.12, the complexity is given by

$$C_M = 2ML \; mults/sample + 2M(L-1) \; adds/sample, \tag{4.17}$$

i.e., it linearly depends on the number of stages, even though the number of filters in the structure increases exponentially.

It is obvious that a direct application of the cell from Figure 3.3 is ineffective since the convolution in both filters is performed for every input sample, and then, the outputs are downsampled. Using the polyphase representation from Figure 3.4, the downsampling is done first, and then four convolutions with $L/2$-order filters are performed on every two input samples as illustrated in Figure 4.7.

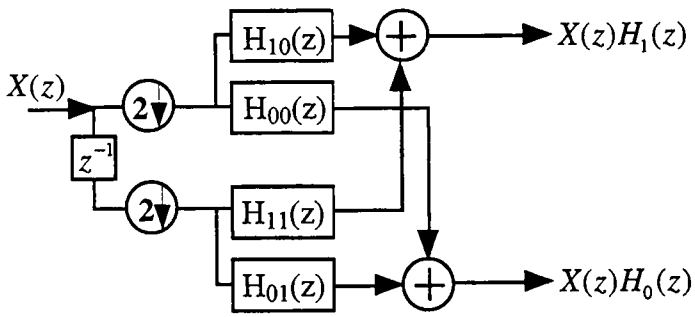


Figure 4.7 Polyphase representation of the analysis filter pair.

The polyphase implementation thus halves the complexity of the cell given by (4.15), and consequently, of the octave-band filter banks, (4.16), and the full-tree filter banks, (4.17).

Further reduction of the computational complexity can be achieved using special relations between filters $H_0(z)$ and $H_1(z)$. For the QMF filters, $H_1(z) = H_0(-z)$ so that the analysis cell can be redrawn as illustrated in the Figure 4.8.

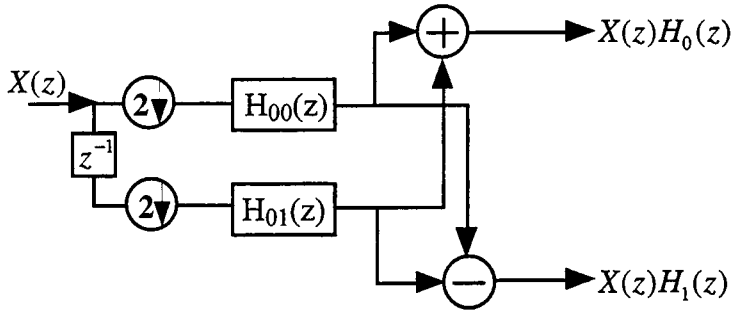


Figure 4.8 Polyphase representation of the QMF analysis pair.

Filters $H_{00}(z)$ and $H_{01}(z)$ are $L/2$-order polyphase components of the filter $H_0(z)$. The cell complexity is now

$$C = (L/2)\ mults/sample + (L-1)/2\ adds/sample, \tag{4.18}$$

i.e., four times lower than the complexity of the general case (4.15).

It is well known that convolution can be calculated much faster if done in the frequency domain using the FFT. The equivalent structure of the cell in Figure 4.8 using FFT-based convolution, is given in Figure 4.9.

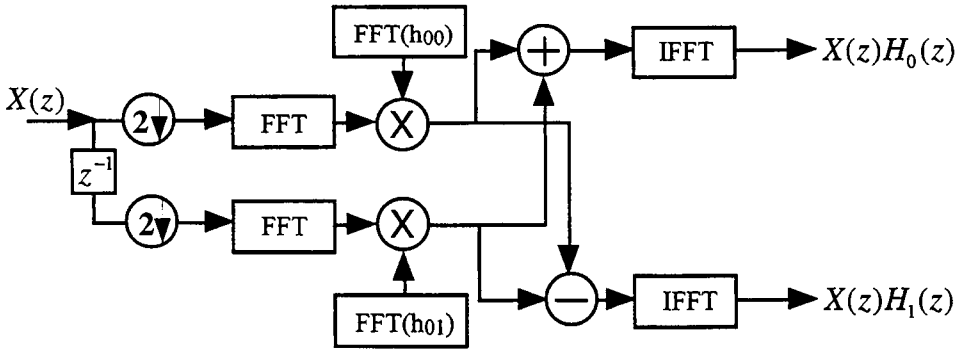


Figure 4.9 FFT-based implementation of the QMF analysis pair.

There is a problem involved in the FFT-based convolution related to the wrap-around effect. Therefore, the FFT needs to be performed in blocks that are at least L-1 samples longer than the input blocks. There exist two well-known methods for fast convolution that eliminate the wrap-around effect. These are the overlap-add and the overlap-save methods [93]. To apply either of these two methods for the structure in Figure 4.9, the input block length needs to be

$$B = 2N - (L-2) \tag{4.19}$$

where N is the size of the FFT and is chosen to be of the form $N = 2^n, n \in \mathcal{N}$.

To compute the output of the structure in Figure 4.9 two N-size FFTs need to be performed first. Then, two frequency domain convolutions are performed by multiplying the Hermitian symmetric FFT of the input by the Hermitian symmetric FFT of the filter. This requires $2N/2$ complex multiplications for the two filters. Finally, two blocks are added requiring $2N/2$ additions, and two N-size IFFTs are applied at the end. It is shown in [93] that a complex multiplication can be performed with three real multiplications and three real additions. Then, the total number of operations (multiplications + additions) per input sample is given by

$$\begin{aligned} C(N) &= \frac{N(2\log_2 N - 3) + 8}{2N - (L-2)}\, mults/input + \frac{6N(\log_2 N - 1) + 16}{2N - (L-2)}\, adds/input \\ &= \frac{N(8\log_2 N - 9) + 24}{2N - (L-2)}\, ops/input \end{aligned} \tag{4.20}$$

where the N-size split-radix FFT is used. Its complexity is given by

$$C = \left[(N/2)(\log_2 N - 3) + 2\right] mults + \left[(N/2)(3\log_2 N - 5) + 4\right] adds. \tag{4.21}$$

The optimal FFT size that minimizes $C(N)$ given in (4.20) satisfies

$$N_{opt} = (L/2 - 1)(\ln N_{opt} + 1 - \frac{9}{8}\ln 2) + 3\ln 2. \tag{4.22}$$

It is shown in [92] that, for large L, N_{opt} is given by

$$\ln N_{opt} = \ln L + O(\ln(\ln L)). \tag{4.23}$$

Then, the minimal value for $C(N)$ is given by

$$C_{\min}(N) = 4\log_2 L + O(\log(\log L)). \tag{4.24}$$

If this value is compared to (4.18) where the total number of operations (mults+adds) is equal to L, then the FFT method becomes more effective for longer filters ($L > 16$) [92].

For shorter filters, one of the *fast running FIR algorithms* [94, 95] may be used. These algorithms will not be considered closely here since the filters used in practice in the tree-structured filter banks for music signal analysis have length $L \geq 64$.

According to the above discussion, the following conclusions regarding the complexity of the tree-structured filter banks of Figure 4.1 can be made. If the cells used in the tree-structured filter banks have complexity C_{cell}, determined by (4.18) or (4.20), then the complexity of the whole structure is given by

$$C_{tot} = C_{cell}\left(M_{\min} + 1 - 2^{-I_{\min}}\right) \tag{4.25}$$

where $M_{\min}$ and $I_{\min}$ are given by (4.6) and (4.8) respectively. For example, if the filter length is $L = 256$, $M_{\min} = 6$, and $L_{\min} = 5$, then the total number of operations, if standard time convolution is used, is equal to 1784. That means that the whole structure has the complexity as an equivalent filter with length 892 taps. It is remarkable that the whole structure containing 318 filters of length 256 has the complexity of an equivalent filter only 3.5 times longer. This is due the fact that the filters in the tree-structured filter banks are QMF filters ($H_1(z) = H_0(-z)$), and that every stage of the structure contains downsampling by 2.

If instead of time convolution, the FFT method is used for the cell calculation, then, for the same number of filter taps and stages in the structure, the total complexity is only 278.75 operations. Here, (4.20) is used to calculate the complexity of a single cell, and the size of the FFT is 2048. The complexity of the tree-structured filter bank is now 6.4 times lower than that with time convolution. This leads to the conclusion that the tree-structured filter banks, even though they seem to be complicated, can be very effectively implemented, especially if the FFT method is used.

Hardware implementation of the tree-structured filter banks will not be discussed here since there exist many implementation solutions for the STFT and DWT proposed recently. They include implementations using standard DSP chips [96, 97], as well as special VLSI architectures [98, 99, 100]. These solutions can be applied to tree-structured filter banks in a straightforward way.

4.3.5 Tree-Structured Filter Banks vs. Other Methods

As mentioned in Chapter 1, methods for front-end music signal processing used so far are based on the STFT and its derivatives. Throughout Chapter 2 and Chapter 3 the STFT was compared to the WT and filter banks, and its inferiority regarding time-frequency resolution and orthogonality was clearly shown. Here, the most referenced methods that use the STFT for front-end music signal processing are compared with the proposed tree-structured filter banks with respect to their complexity as well.

The STFT in different forms has been widely used for the front-end signal processing of music signals [25, 27, 38, 45, 51, 55, 59]. As shown in Chapter 2 and Chapter 3 it has constant time-frequency resolution over the entire time-frequency space. For example, if the analysis needs to separate two semitones at 130 Hz and the input is sampled at 24 kHz, then the discrete spectrum needs to have aa spacing of at least 7.8 Hz (Q-factor 16.72), or 24,000 Hz/7.8 Hz = 3,077 frequency samples. That means that a 4,096-point FFT needs to be performed, i.e., a 4,096-tap analysis window used. However, a window that long will cause extremely high resolution at the high-frequency end of the signal spectrum (Q-factor = 12,000 Hz/7.8 Hz = 1,538). Further, if the time shift in the STFT needs to be the same as in the example with the tree-structured filter bank (64/24,000 Hz = 2.67 ms), then the computational complexity is determined by the 4,096-size FFT performed every 64 input samples. Using (4.21), the complexity is C = 1,280 operations per input sample which is much greater than the complexity of the tree-structured filter bank. Besides, the time resolution is very poor over the entire frequency

band due to the long window used in the procedure.

The Multiresolution Short-Time Fourier Transform [MSTFT] proposed in [59] improves the time-frequency resolution of the STFT performing several standard STFTs with different analyzing windows. Thus, the output of the STFT with a narrow window is used for localization in time, while the STFT with a wider window gives good frequency resolution. Obviously, the complexity of the MSTFT is greater than the complexity of a single STFT.

The constant-Q STFT proposed by J. Brown [55] improves time-frequency resolution using a variable-size analyzing window within a single STFT. The analyzing window is wider for lower frequencies and narrower for higher frequencies thus performing a constant-Q analysis. The fast implementation of the algorithm [56] uses the frequency domain to perform the STFT. An example from [56] shows that the complexity of the algorithm when the lowest frequency is 175 Hz and the sampling frequency is 11,025 Hz, corresponds to a 1024-point FFT. The FFT is calculated every 25 ms (275 input samples) thus giving a low complexity of 223.4 operations per input sample. The tree-structured filter bank still has lower complexity since it covers more than 6 octaves having a complexity of 278.75 operations per input sample while in the example from [56], less than 5 octaves are covered. Another drawback of the constant-Q STFT is that it does not represent an orthogonal signal expansion and even more, it does not have an inverse [55]. Therefore, the method is limited in its use to the signal analysis only.

It is shown in this chapter that tree-structured filter banks are good candidates for music signal analysis. They have good frequency resolution over a very broad band of frequencies. As EXAMPLE 4.1 shows, using 6 full-tree stages and 5 half-band stages it is possible to cover almost 7 octaves with the frequency resolution necessary to separate two adjacent semitones. The time resolution achieved by the structure is 2.67 ms at the high frequencies which is much below the minimum time resolution needed for the event

detection in music signals. Also, the computational complexity of the tree-structured filter banks is relatively low if fast filtering algorithms are used. Use of the QMF banks in those structures allows even more effective computational solutions. Even though the presented tree-structured filter banks are near-perfect reconstruction filter banks, the reconstruction error is very small and acceptable in practical applications.

In the next chapter, the analysis part of the tree-structured filter bank from Figure 4.1 is applied in an automated polyphonic music transcription system. In an another application, the whole analysis/synthesis structure is used for voice separation and noise reduction in a polyphonic piano piece.

CHAPTER 5

IMPLEMENTATION OF TREE-STRUCTURED FILTER BANKS IN POLYPHONIC PIANO MUSIC SEGMENTATION

5.1 Introduction

In the previous chapter, tree-structured filter banks for music signal processing were developed. The structure of the filter bank can be selected so that sufficient frequency resolution is achieved over the entire signal spectrum. It was shown that an 11-stage structure covers almost seven octaves. Furthermore, if the two-channel filter bank used in the structure is a perfect reconstruction bank, then the whole structure has the perfect reconstruction property as well. To minimize the frequency leakage in the structure, the two-channel filter bank used in the structure needs to have a very narrow transition band. However, perfect reconstruction and QMF filter banks designed so far, have relatively low selectivity. Therefore, in Chapter 4, a high-quality near-perfect reconstruction filter pair was developed and used in the tree-structured filter banks. It was shown that the reconstruction error in these filters is not greater than the round-off error.

This chapter presents two applications of tree-structured filter banks in polyphonic piano music segmentation. In the first one, both synthesized and real-life piano pieces are transcribed into a MIDI stream. In the first step, the transcription algorithm processes the input signal with the tree-structured filter bank thus obtaining the time-frequency representation of the input signal. The time-frequency coefficients are then used for the attack detection in a piano piece. An original method for the attack detection based on the energy spread in the frequency domain is developed and implemented. Then, the detected attack locations are used in the pitch detection algorithm based on template matching. Next, the duration of each of the detected notes is determined. At the end, the extracted

note parameters are transformed into a MIDI file. The MIDI file is then played back on a piano sound module so that the reproduced music can be compared with the input signal.

Generated MIDI files offer many new possibilities for analysis and implementation of the original input music. First of all, a MIDI file represents a compressed version of the input signal with a compression ratio of about 500:1. Also, it can be used to play back the input piano piece with different speeds, different voices, and different transpositions. Furthermore, using a MIDI representation of a piano piece, one can incorporate it into orchestral MIDI music.

All of the separate algorithms for the feature extraction mentioned above are later combined in an automated transcription algorithm. It is shown that a fully automated transcription system performs well on the synthesized music signals. However, for real-life music recordings, human intervention during the transcription process is necessary. Therefore, an interactive algorithm for the transcription of real-life polyphonic music is developed and implemented.

In the second application, a tree-structured filter bank is used for voice separation in a piano piece. Here, both the analysis and the synthesis part of the tree-structured filter banks are implemented. To be able to effectively separate particular notes, the whole structure needs to have both good time-frequency resolution and low reconstruction error. An interactive algorithm that uses tree-structured filter banks is developed and implemented for the separation of two voices in a real-life piano piece. Also, the algorithm is used to reduce the noise level in the original recording and to remove some other disturbances.

The above mentioned algorithms were developed and implemented on an Apple Macintosh PowerPC 8100/100 AV using LabVIEW, a graphically oriented programming language from National Instruments. The synthesized input signals were generated by PROformance Plus, a piano sound module from E-mu Systems, while the real piano recording represents the first 10 seconds of Beethoven's piano piece *Für Elise* played by

Lesia Shulha, piano music performer and teacher. The piano piece was recorded onto audio tape, and then played back and acquired for further analysis.

The MIDI files generated by the transcription algorithm were captured and fed into a sound module using *TurboTrax,* a sequencer software [101]. For the input signal acquisition, an Apple Macintosh IIx equipped with a plug-in acquisition board National Instruments NB-A2150 was used. The acquisition board performs 16-bit acquisition with up to a 51.2 kHz sampling rate. The input signals used in the applications were acquired with a 16-bit resolution and 24 kHz sampling rate, and were 240,000 samples long.

5.2 Polyphonic Piano Music Transcription

The polyphonic piano music transcription algorithm has the general structure presented in Figure 5.1. As Figure 5.1 shows, the input signal is first transformed into its

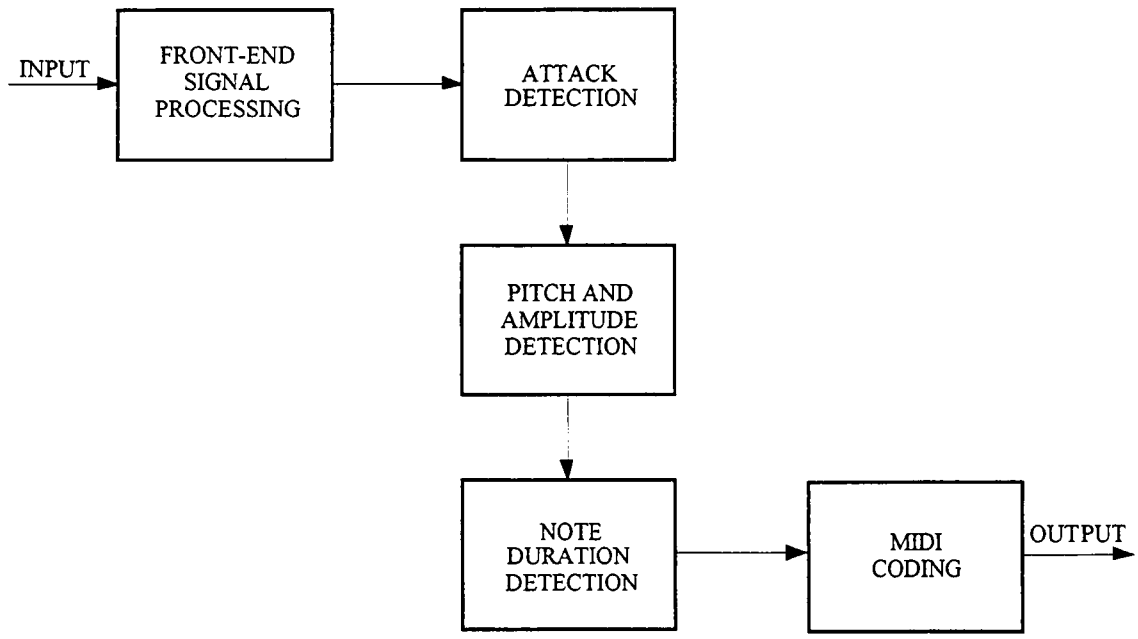


Figure 5.1 Polyphonic music transcription algorithm.

time-frequency representation using the tree-structured filter bank. Then, the time-frequency coefficients are used to extract all necessary signal features such as the attack time, pitch, duration and amplitude of each of the notes in the piece. At the end, the extracted features are transformed into a MIDI stream. Below, particular parts of the transcription algorithm are described.

5.2.1 Front-End Signal Processing

For the front-end signal processing, the tree-structured filter bank of Figure 4.1 is used. It contains a 6-stage full-tree filter bank and a 5-stage half-band bank. The filters used in the structure are the 256-tap filters of Figure 3.11 (b). Since the input files were 240,000 samples long, the convolution was performed on 10,000-sample-long input blocks using the overlap-save method [93]. For the implementation of the tree-structured filter banks of Figure 4.1, a LabVIEW program was developed. The front panel and block diagrams of the program are given in Appendix B.

Figure 5.2 shows the time-frequency representation of the input signal containing a 12-note scale starting from C_2 (130.81 Hz). The frequency axis in Figure 5.2 is given in the subband indices from 0 to 223. The starting frequencies, $f_{SB}(i)$, and the bandwidths, $BW_{SB}(i)$, of the subbands are calculated according to

$$\begin{aligned} f_{SB}(i) &= \begin{cases} iBW_{SB}(i), & i = 0,...,63 \\ \left(32 + i - \lfloor i/32 \rfloor 32\right) BW_{SB}(i), & i = 64,...,223 \end{cases} \\ BW_{SB}(i) &= \begin{cases} f_s / 2^{M_{\min} + I_{\min} + 1}, & i = 0,...,63 \\ f_s / 2^{M_{\min} + I_{\min} - \lfloor i/32 \rfloor + 2}, & i = 64,...,223 \end{cases} \end{aligned} \tag{5.1}$$

The time axis in Figure 5.2 is given in seconds. The sampling frequency, $f_{SSB}(i)$,is different in each of 6 subband blocks, and is given by

$$f_{SSB}(i) = 2BW_{SB}(i), \quad i = 0,...,223. \tag{5.2}$$

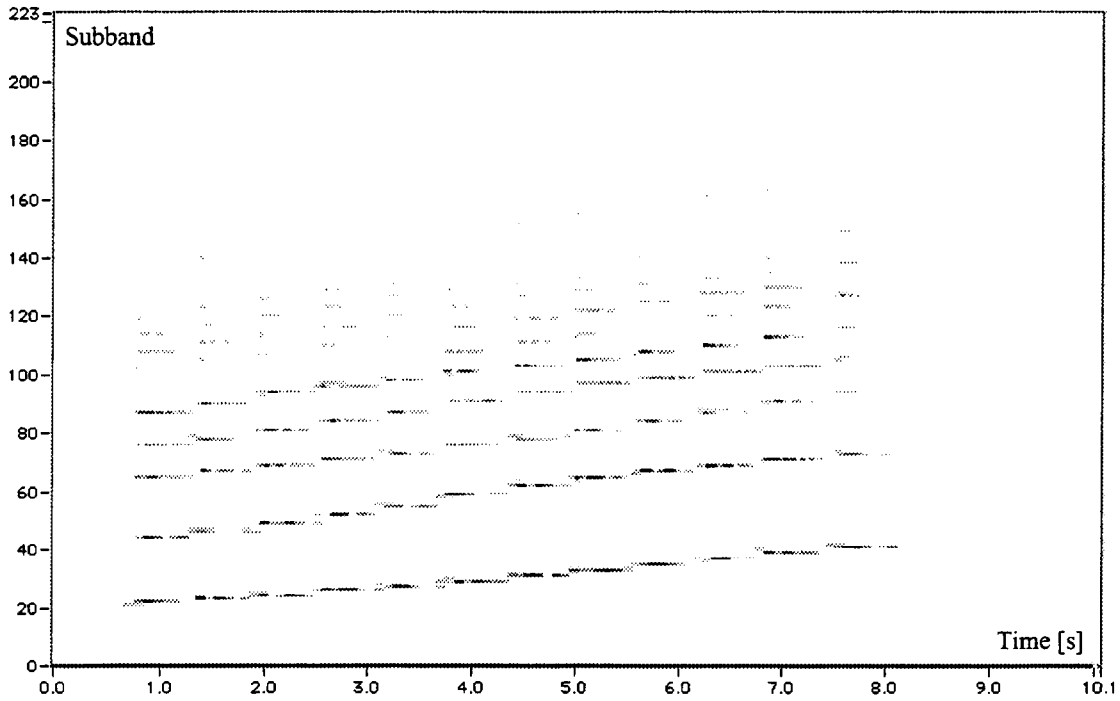


Figure 5.2 Time-frequency representation of the 12-note scale starting from C_2.

Table 5.1 shows the starting frequencies and the bandwidths for the subbands indicated on the vertical axis, while Table 5.2 shows the sampling frequencies and the corresponding time resolutions, $1/f_{SSB}(i)$, in each of the subband blocks.

As Figure 5.2 illustrates, the time-frequency representation clearly shows the frequency as well as the time structure of the input signal. Rich frequency structure indicates a large number of overtones for low-frequency piano notes. Along the time axis, the attack times and the duration of each of the notes are clearly visible.

Figure 5.3 represents a 12-note scale starting from C_4 (523.25 Hz). The number of overtones in this case is lower than for the notes in Figure 5.2. Figure 5.4 illustrates the time frequency representation of a 12-note scale starting from C_6 (2093 Hz). As Figure 5.4 indicates, the high piano notes have almost all of their energy concentrated in the fundamental frequency. Also, their energies decay very quickly after the attack.

Table 5.1 Subband frequencies and bandwidths.

Subband	0	20	40	60	80	100
Frequency [Hz]	**0**	**117.2**	**234.4**	**351.6**	**562.6**	**843.5**
Bandwidth [Hz]	**5.86**	**5.86**	**5.86**	**5.86**	**11.72**	**23.43**
Subband	120	140	160	180	200	223
Frequency [Hz]	**1,313**	**2,062**	**3,000**	**4,875**	**7,500**	**11,813**
Bandwidth [Hz]	**23.43**	**46.88**	**93.75**	**93.75**	**187.5**	**187.5**

Table 5.2 Sampling frequencies and time resolutions in subbands.

Subbands	0-63	64-91	92-127	128-159	160-191	192-223
Sampling frequency [Hz]	**11.72**	**23.44**	**46.88**	**93.75**	**187.5**	**375**
Time resolution [ms]	**85.44**	**42.72**	**21.36**	**10.68**	**5.34**	**2.67**

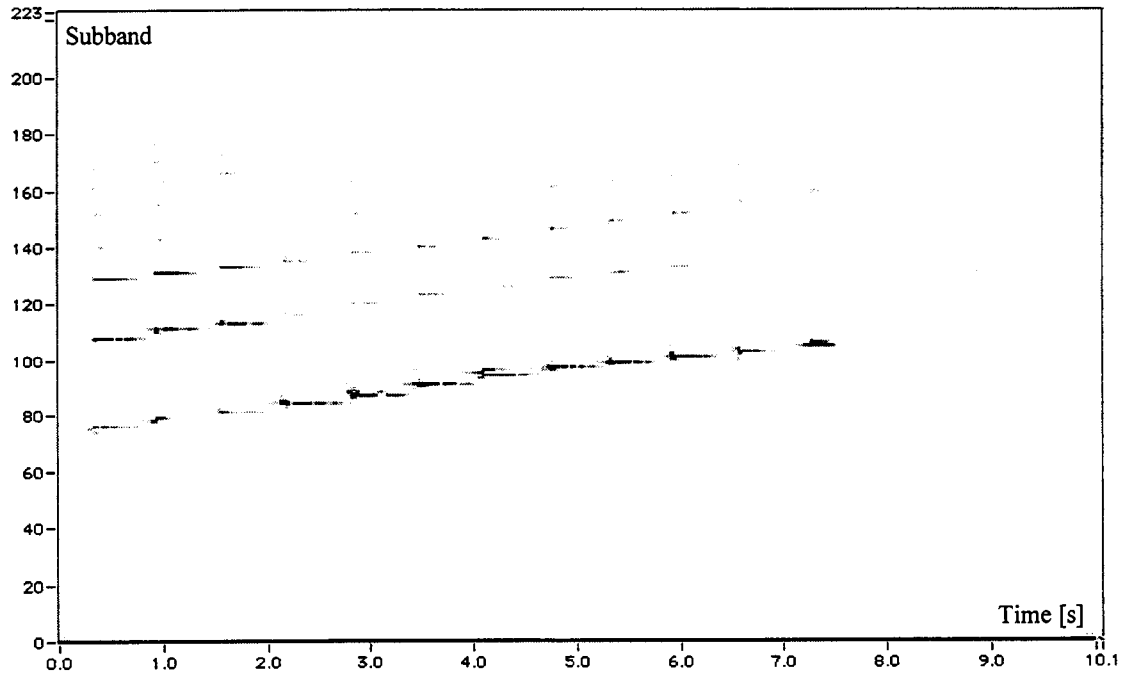


Figure 5.3 Time-frequency representation of the 12-note scale starting from C_4.

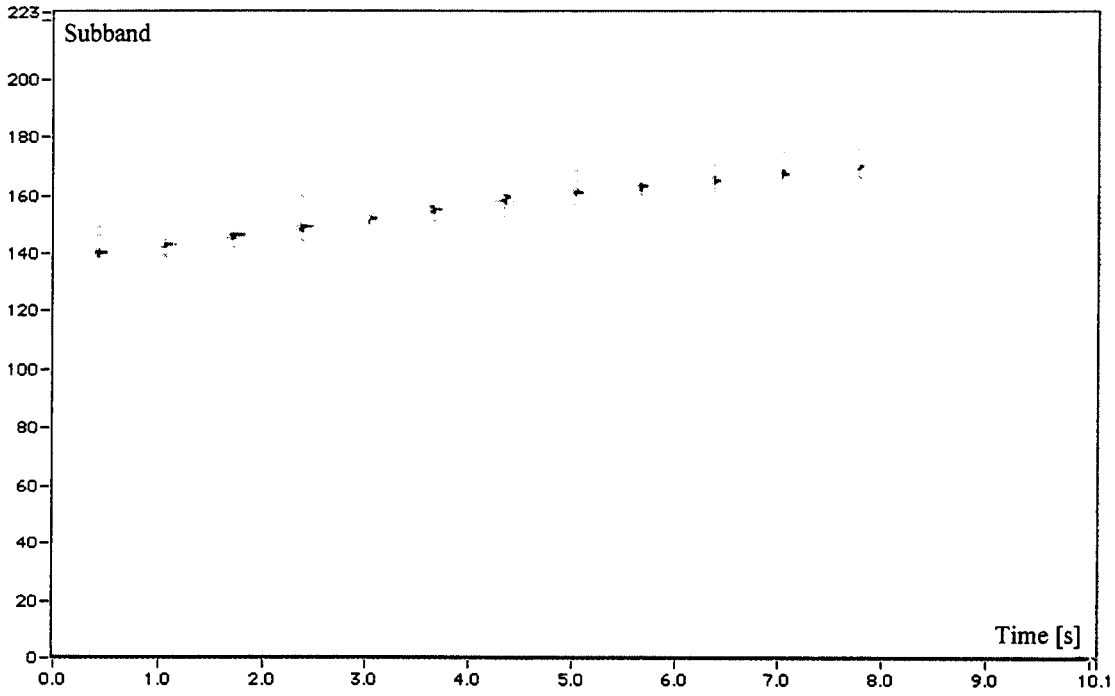
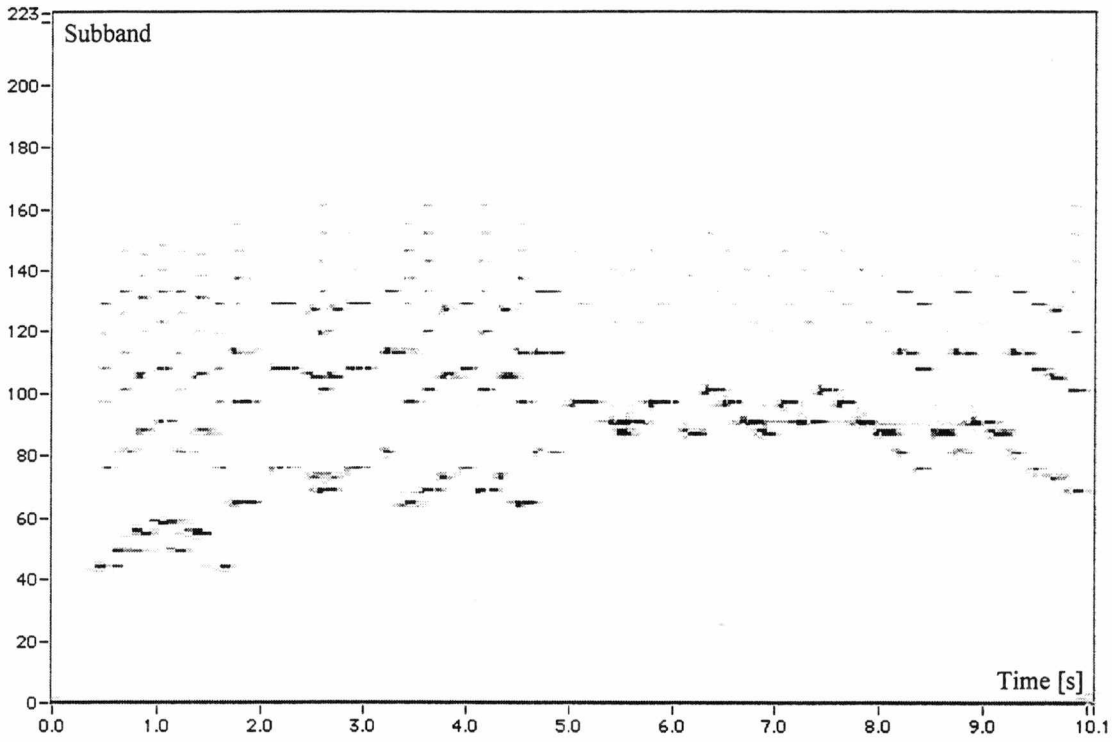


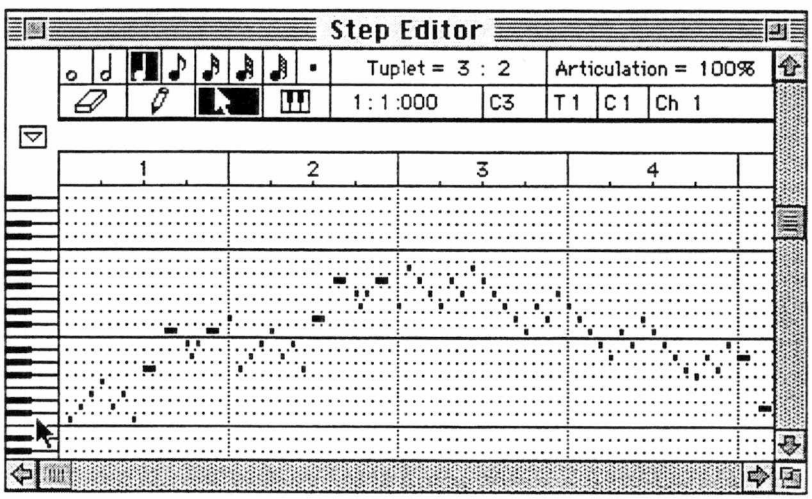
Figure 5.4 Time-frequency representation of the 12-note scale starting from C_6.

Figure 5.5 (a) illustrates the time-frequency representation of a piano piece played from the library of MIDI files in TurboTrax, a sequencer software. The first 10 seconds of the piano piece have their score-like representation given in Figure 5.5 (b). A two-voice piano piece is presented in Figure 5.6 (b), together with its scores in Figure 5.6 (b). A more complicated piano piece with a scale starting at C_2 and three chords at the end is given in Figure 5.7 (a). Figure 5.7 (b) shows the corresponding TurboTrax scores.

The inputs in Figures 5.2 to 5.7, were synthesized, noise-free piano pieces played by the sound module, while in Figure 5.8 (a) a real-life piano recording was used. The piano piece, Beethoven's *Für Elise*, was previously recorded onto audio tape, and then played back and acquired by the equipment described at the beginning of the chapter. As visible from Figure 5.8 (a), the recording itself was relatively noisy, and a 120 Hz disturbance was present giving a stripe in the time-frequency plane located at the

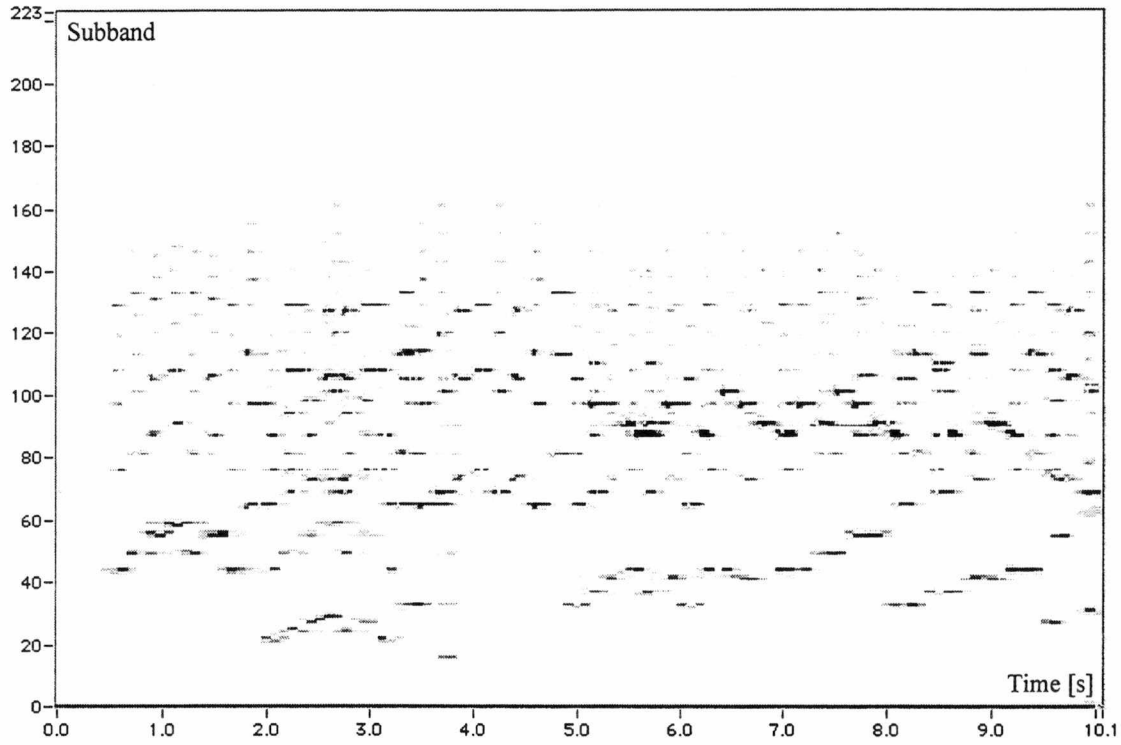


(a)

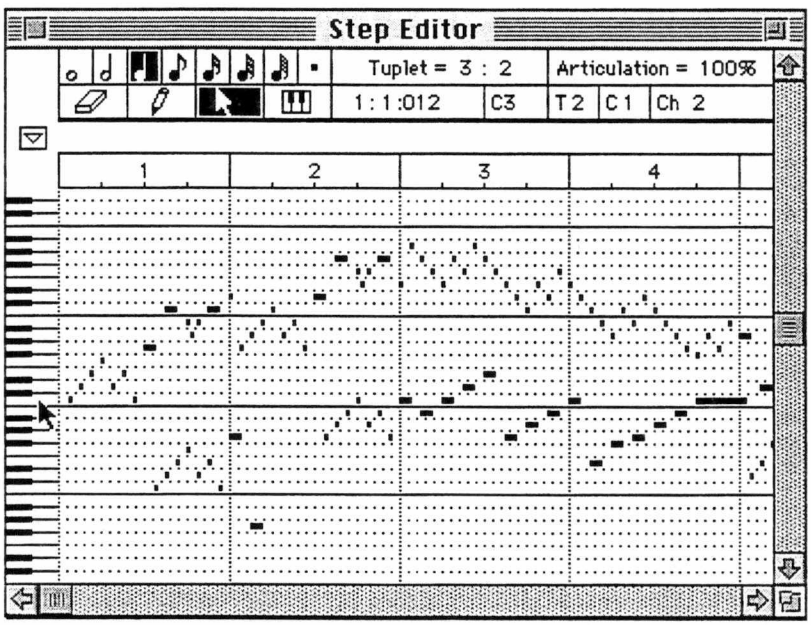


(b)

Figure 5.5 A single-voice piano piece. (a) Time-frequency representation. (b) Corresponding Turbo Trax scores.

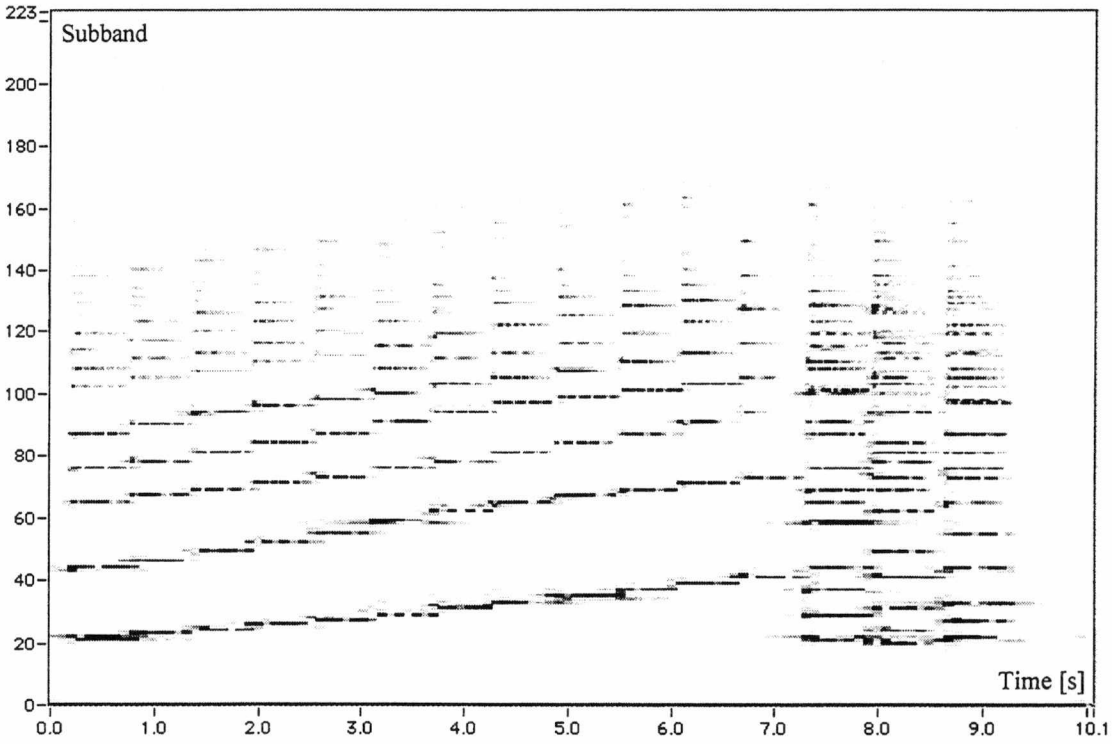


(a)

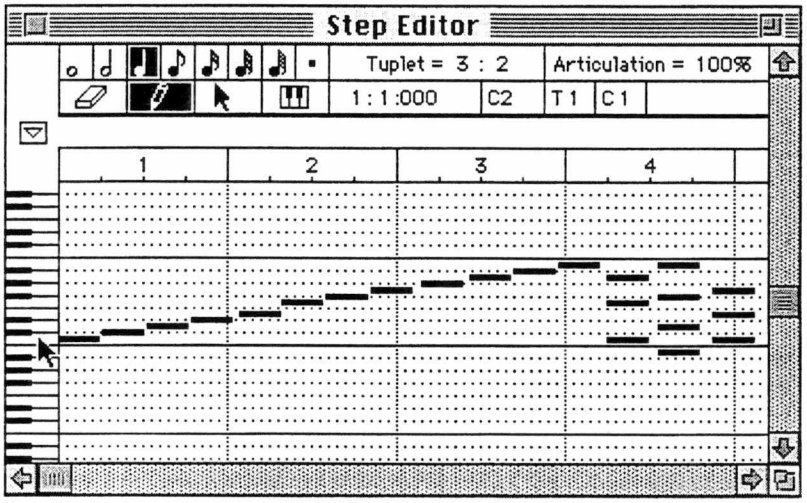


(b)

Figure 5.6 A two-voice piano piece. (a) Time-frequency representation. (b) Corresponding TurboTrax scores.

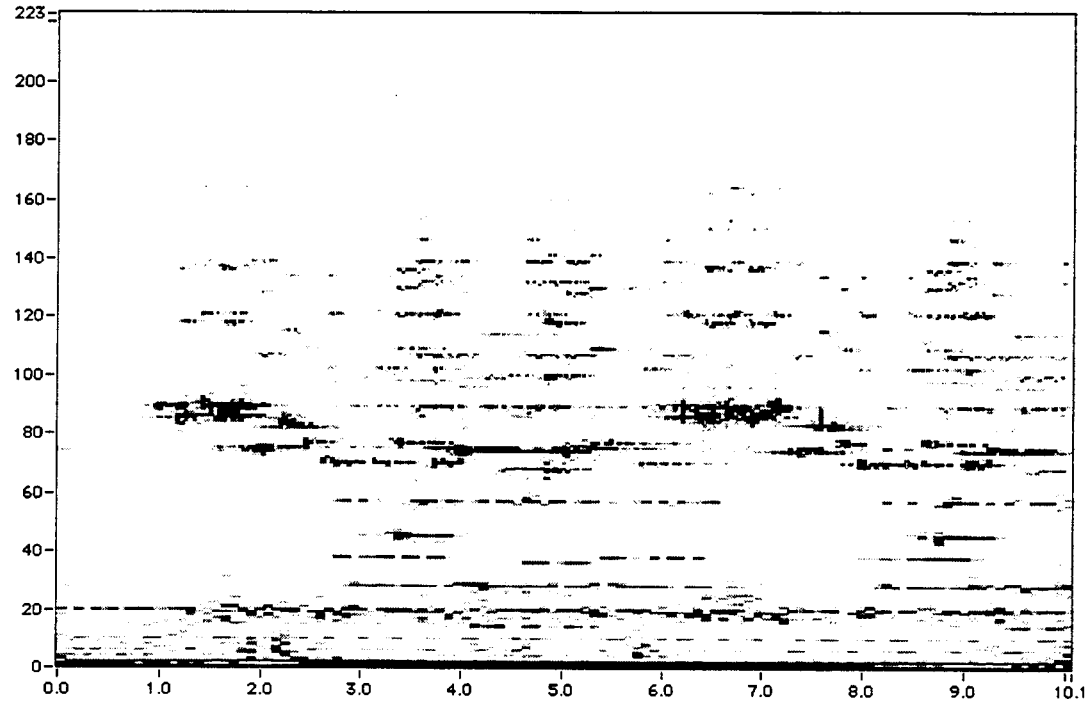


(a)

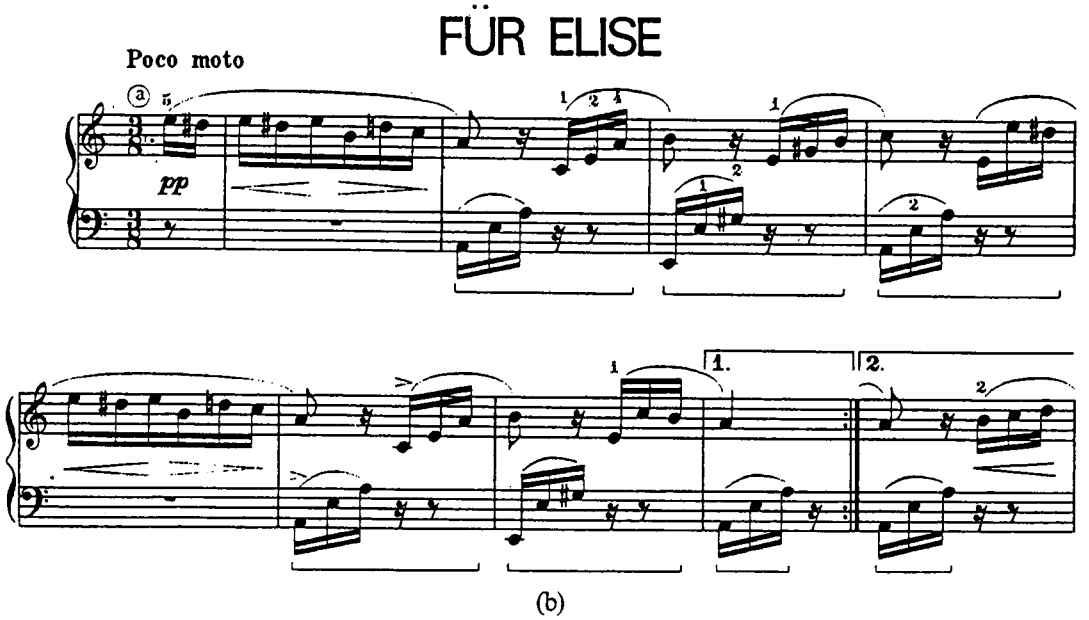


(b)

Figure 5.7 A scale starting at C_2 and three chords at the end. (a) Time-frequency representation. (b) Corresponding TurboTrax scores.



(a)



(b)

Figure 5.8 The first 10 seconds of Beethoven's *Für Elise*. (a) Time-frequency representation. (b) Corresponding sheet music.

20^{th} subband. Also, a lot of signal energy was located at the lowest frequencies, as well as in the DC component. Figure 5.8 (b) shows the sheet music for the corresponding 10 seconds of the recording.

Figures 5.2 through 5.8 illustrate the outputs of the front-end signal processing using the tree-structured filter bank. The test signals are synthesized, monophonic and polyphonic, noise-free piano pieces, as well as a real-life piano recording. The output of the front-end signal processing are the time-frequency coefficients that are used in the feature extraction algorithms described in the next subsections.

5.2.2 Attack Detection Algorithm

The time-frequency coefficients obtained by the tree-structured filter bank, are used for the detection of the attacks. All of the methods for the attack detection from the referenced literature rely on the time-domain envelope analysis. Basically, the algorithms look for the steepest energy rise thus detecting the sharp transients characteristic for the attack. These time-domain based techniques become unreliable if the input signal is corrupted by noise, or beating between two close frequencies occurs.

In this dissertation, an original attack detection algorithm based on the signal energy spread over the time-frequency space was developed. The algorithm relies on the noisy characteristic of the attack in percussive instruments such as a piano. At the time of the attack the signal energy is spread over more subbands, while in the decay part the energy is concentrated in the fundamental frequencies and the overtones.

The attack detection algorithm calculates a measure defined as the number of the subbands in which the time-frequency coefficients exceed an adaptive energy threshold. The measure, $ES(n)$, is given by

$$ES(n) = \sum_{i=0}^{N_{SB}} Z(i,n), \quad n = 0,\ldots,N/2^{M_{\min}} - 1 \tag{5.3}$$

where N is the total number of input samples, $M_{\min}$ is the number of stages in the full-tree

part of the tree-structured filter bank defined in (4.6), N_{SB} is the number of subbands defined in (4.9), and $Z(i,n)$ is 0 or 1 defined by

$$Z(i,n) = \begin{cases} 0, & E(i,n) < E_{tr} \\ 1, & E(i,n) \geq E_{tr} \end{cases}. \tag{5.4}$$

The energy of the time-frequency coefficients, $E(i,n)$, is calculated according to

$$E(i,n) = \sum_{j=L(i,n)}^{K(i)} C^2(i,j) \tag{5.5}$$

where

$$\begin{aligned} L(i,n) &= \left\lfloor n2^{-M_{\min}+\lfloor i2^{-M_{\min}+1} \rfloor} \right\rfloor \\ K(i) &= 2^{\lfloor i2^{-M_{\min}+1} \rfloor} \end{aligned} \tag{5.6}$$

and $C(i,j)$ is the jth time-frequency coefficient in the ith subband. $\lfloor x \rfloor$ denotes the largest integer smaller than or equal to x. The energy threshold, E_{tr}, is calculated as the average energy per input sample, i.e.,

$$E_{tr} = \frac{\alpha}{N} \sum_i \sum_j C^2(i,j), \quad 0 < \alpha < 1 \tag{5.7}$$

where the coefficient α is used to adjust the energy threshold thus changing the sensitivity of the algorithm.

The above described algorithm for the attack detection is applied to input signals whose time-frequency representations are given in Figures 5.2 to 5.8. A LabVIEW program that implements the algorithm is given in Appendix C. Figures 5.9 to 5.15 show the energy spread, $ES(n)$, for the input signals represented by Figures 5.2 to 5.8 respectively.

As the figures show, there exist distinguishable peaks at the attack locations. Note that the time-step in the above graphs is 2.67 ms as explained in Chapter 4. This is much less than the minimum time resolution for the human ear. The minimum time distance between two attacks detectable by the human ear is about 20 ms [36].

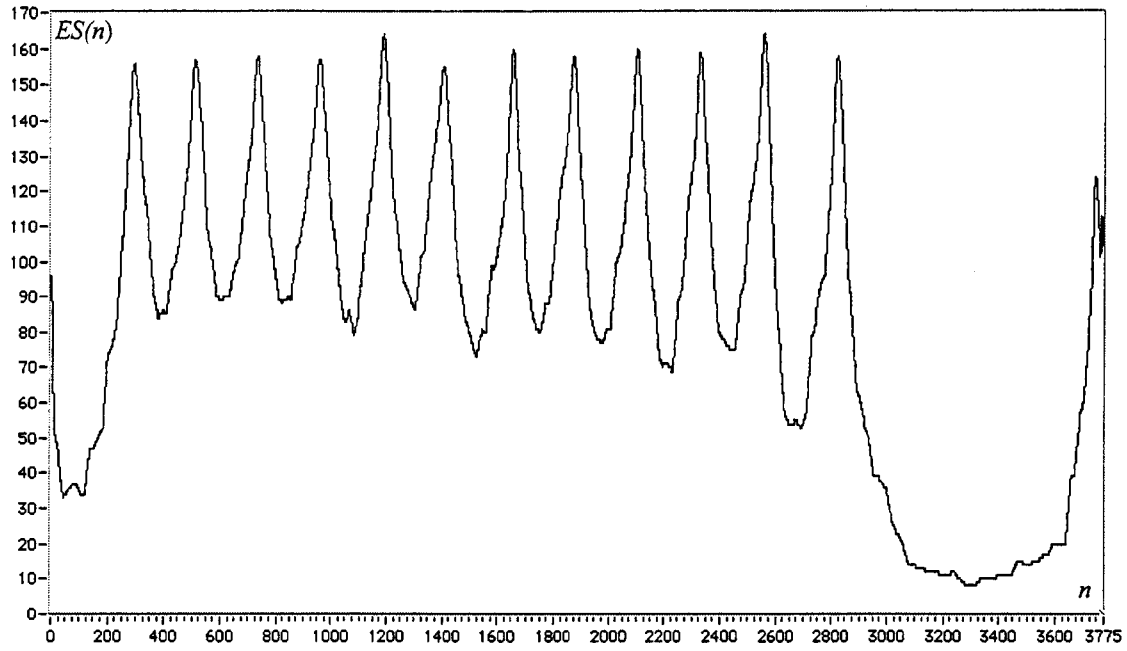


Figure 5.9 Energy spread in frequency domain for the input signal represented in Figure 5.2.

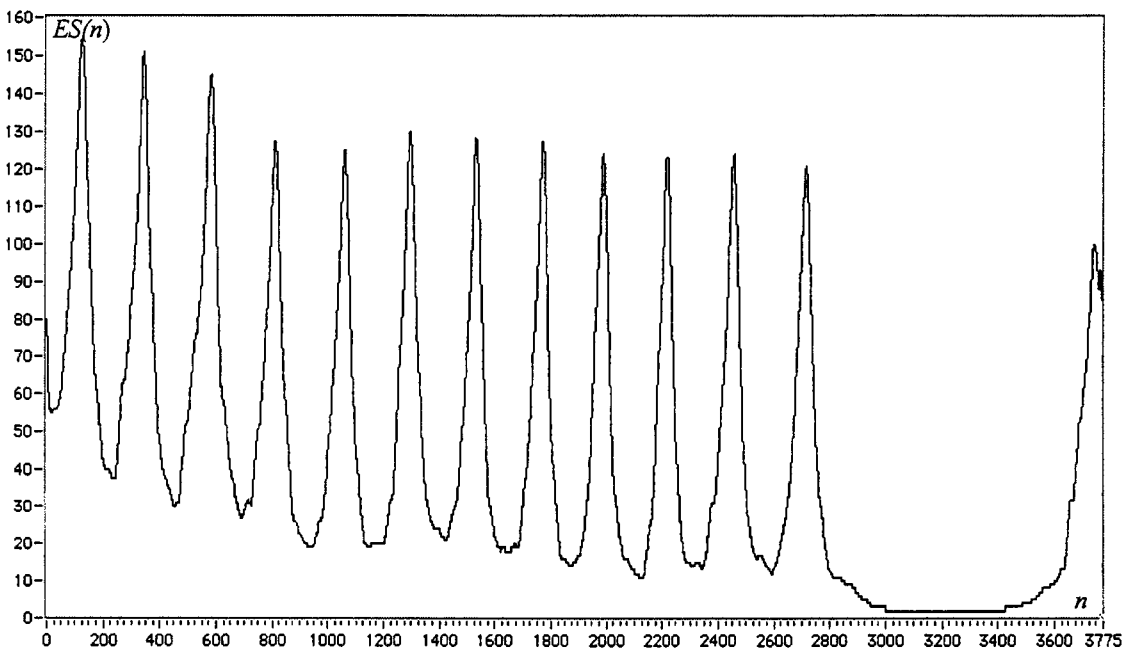


Figure 5.10 Energy spread in frequency domain for the input signal represented in Figure 5.3.

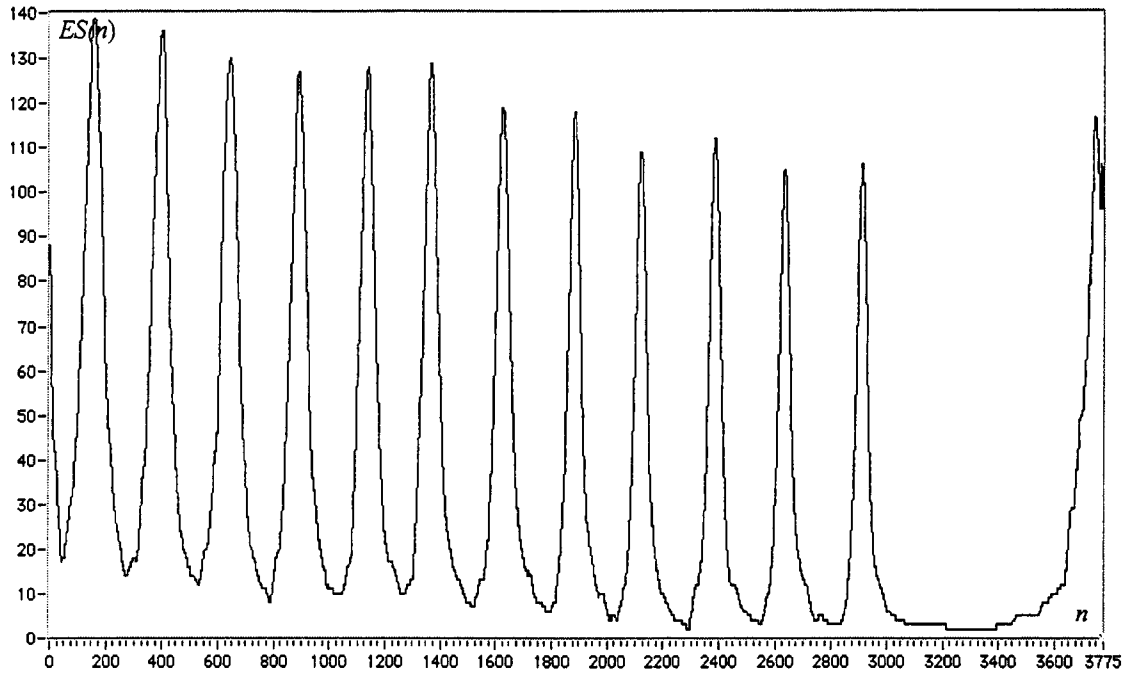


Figure 5.11 Energy spread in frequency domain for the input signal represented in Figure 5.4.

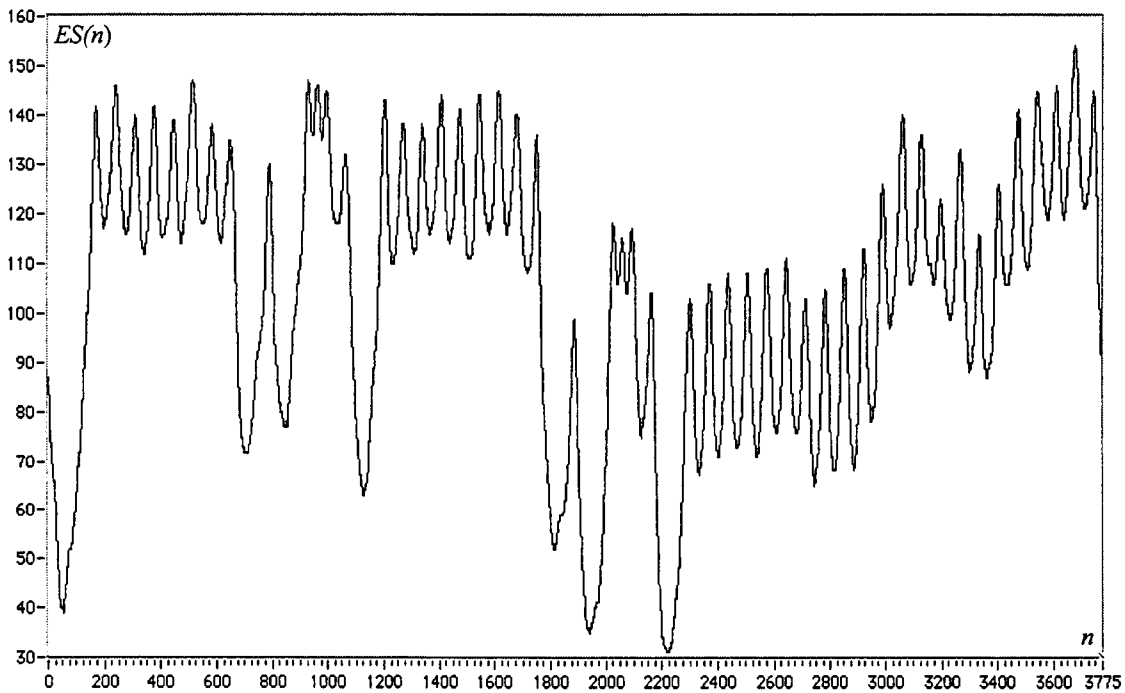


Figure 5.12 Energy spread in frequency domain for the input signal represented in Figure 5.5.

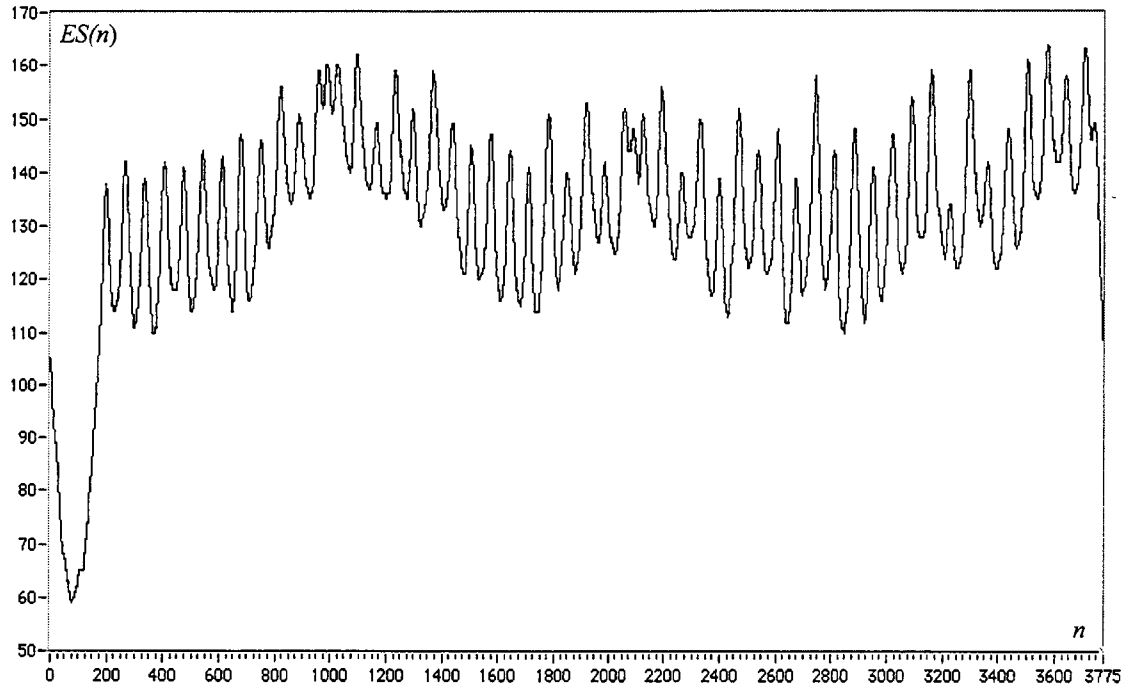


Figure 5.13 Energy spread in frequency domain for the input signal represented in Figure 5.6.

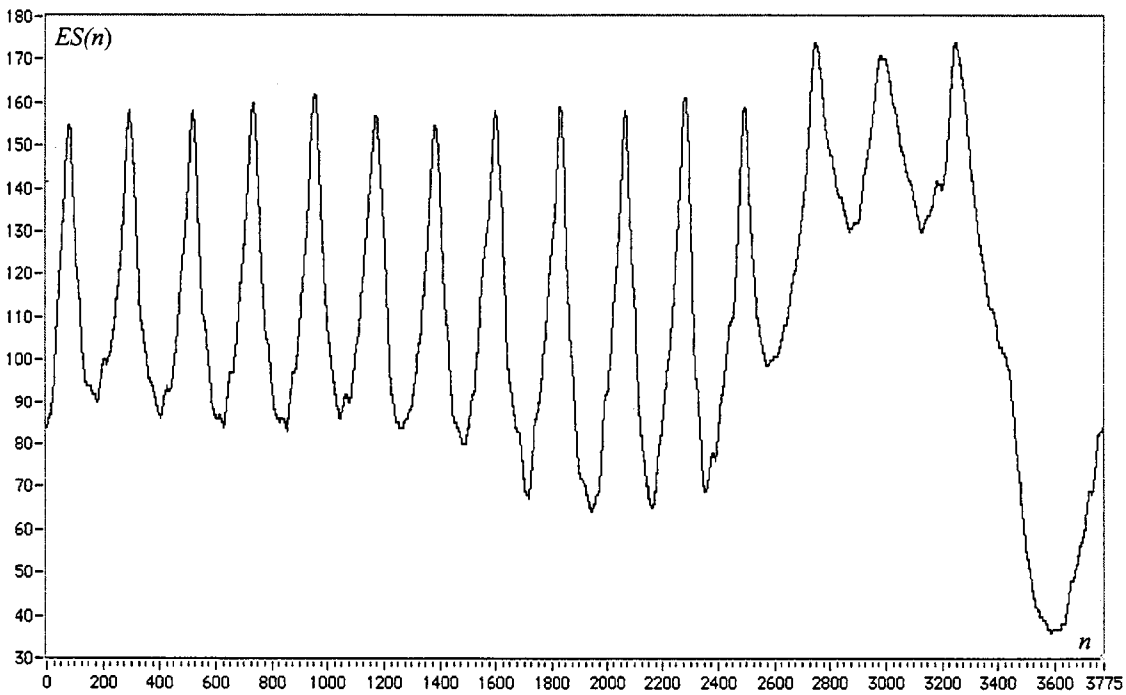


Figure 5.14 Energy spread in frequency domain for the input signal represented in Figure 5.7.

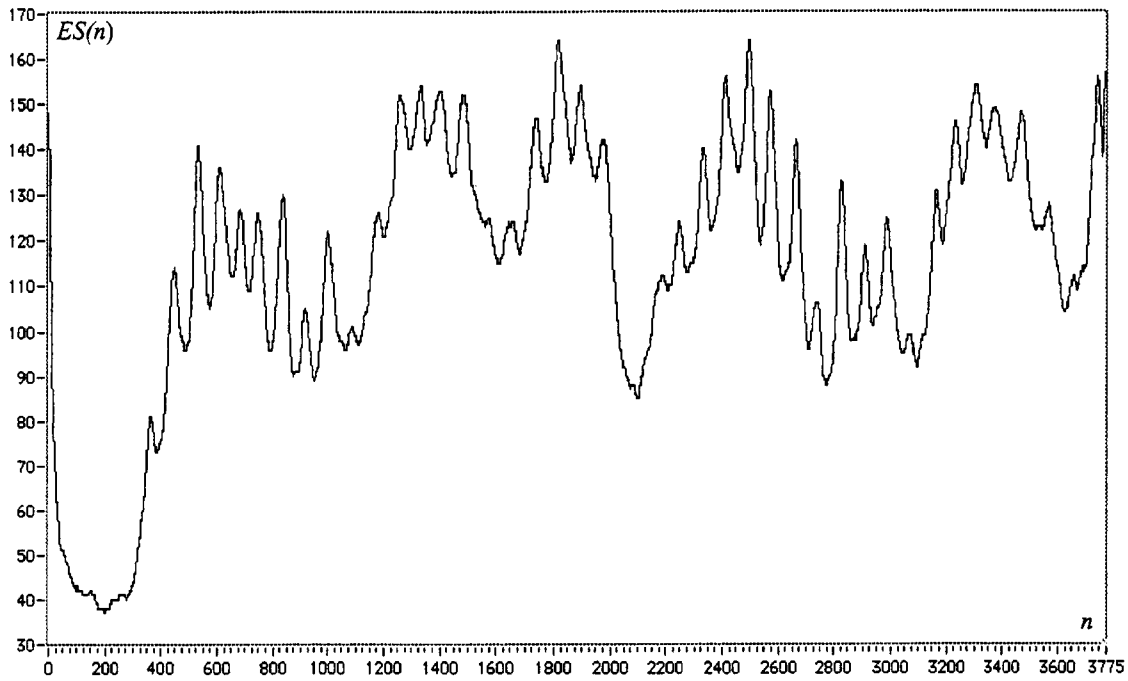


Figure 5.15 Energy spread in frequency domain for the input signal represented in Figure 5.8.

The local maxima in Figures 5.9 to 5.15 are more peaked for the higher notes which is consistent with the fact that the lower notes have a larger number of overtones than the higher ones. Also, if the input signal is noisy as in Figure 5.15, then the maxima are less distinguishable.

What follows next in the attack detection algorithm is the peak detector. In this dissertation, an original peak detection algorithm is developed. It detects the peaks in two steps. In the first step, a sliding window is moved along the time axis, and the index of the maximum value of $ES(n)$ is recorded each time. If the window length is L, then when the window slides over a local peak, the same index will be recorded exactly L times. Obviously, the above described algorithm has the resolution of L, i.e., it can detect two peaks no closer than L samples.

The second step in the peak detection algorithm is used to prevent small disturbances to be detected as valid peaks. It checks the values of $ES(n)$ M samples

apart from the detected peak. These values need to be lower than the detected peak for a preset threshold. If this is satisfied, then the peak is definitely accepted as a valid peak. In the examples that follow, the values for L and M are $L = M = 25$.

The above described peak detection algorithm was applied to the test signals whose $ES(n)$ are represented in Figures 5.9 to 5.15. All of the peaks for the synthesized input signals, Figures 5.9 to 5.14, were detected correctly, and there were no cases of false alarm detection. However, in the case of the real-life recording, Figure 5.15, two attacks were not detected. The missed peaks can be detected by changing the parameters L and M thus increasing the sensitivity of the algorithm. However, then the false alarm rate increases significantly.

It is crucial that all the attacks are detected correctly since the pitch detection algorithm that follows analyzes the frequency contents only at the detected attack locations. To correctly detect all the attacks, the attack detection algorithm was improved by adding an editing feature. Based on previous knowledge about the analyzed piano piece, the user scans $ES(n)$ with a cursor and getting an audio indication if the scanned peak is detected. All undetected peaks are then added to the list of already detected peaks by simply dragging the cursor to the place of the missed peak. Using this same procedure, all the false alarms are removed from the attack list.

The output of the attack detection algorithm is an array containing locations of the attacks. This data is then used in the pitch detection algorithm that determines what notes are played at each of the attacks.

5.2.3 Pitch Detection Algorithm

The algorithm is based on a template matching procedure where the frequency content of the input signal is compared to a list of templates. Each piano note in the range from C_1 to B_6 can be represented by a vector defined as

$$\mathbf{T}_m = \left(a_{m,0}, a_{m,1}, ..., a_{m,N_{SB}}\right), \quad m = 0,1,..., 71 \tag{5.8}$$

where $a_{m,i}$ is the average energy in the ith subband for the mth note, and N_{SB} is the number of subbands defined by (4.9). The indices $m = 0,1,...,71$ correspond to 72 notes from the 6-octave range starting with C_1 (65.41 Hz) as shown in Table 5.3.

Table 5.3 Indices of the notes within 6-octave range starting with C_1.

m	NOTES											
0-11	C_1	C_1#	D_1	D_1#	E_1	F_1	F_1#	G_1	G_1#	A_1	A_1#	B_1
12-23	C_2	C_2#	D_2	D_2#	E_2	F_2	F_2#	G_2	G_2#	A_2	A_2#	B_2
24-35	C_3	C_3#	D_3	D_3#	E_3	F_3	F_3#	G_3	G_3#	A_3	A_3#	B_3
36-47	C_4	C_4#	D_4	D_4#	E_4	F_4	F_4#	G_4	G_4#	A_4	A_4#	B_4
48-59	C_5	C_5#	D_6	D_5#	E_5	F_5	F_5#	G_5	G_5#	A_5	A_5#	B_5
60-71	C_6	C_6#	D_6	D_6#	E_6	F_6	F_6#	G_6	G_6#	A_6	A_6#	B_6

The average subband energy in (5.8) is defined as

$$a_{m,i} = \sum_{j=0}^{K(i)} C_m^2(i,j), \quad m = 0,...,71, \; i = 0,...,N_{SB} \tag{5.9}$$

where $K(i)$ is defined by (5.6). $C_m(i,j)$ in (5.9) are the time-frequency coefficients of the input signal representing the mth note used for the template list generation.

In the template matching procedure, a vector representing the average energy in subbands, $\mathbf{X} = \left(x_0, x_1, ..., x_{N_{SB}}\right)$, is calculated at every attack according to

$$x_i = \sum_{j=0}^{K(i)} C_x^2(i,j), \quad i = 0,...,N_{SB} \tag{5.10}$$

where $C_x(i,j)$ are the time-frequency coefficients of the input signal calculated according to (5.9).

To detect a particular note represented by its template, the measure $d_m(\mathbf{X}, \mathbf{T}_m)$ that describes similarity between vectors $\mathbf{X}$ and $\mathbf{T}_m$ is calculated. However, the vectors $\mathbf{T}_m$ have only several significant components making it possible to reduce the calculation

complexity of the measure. The number of significant components in the vectors $\mathbf{T}_m$ corresponds to the number of overtones in a piano note. The lowest piano notes may have up to 30 overtones, while the highest ones have only few significant overtones. Thus, instead of $N_{SB} = 224$ components, the template vectors may have up to 30 components.

There is an alternative choice for the template vectors; instead of the variable-length template vectors, constant-length template vectors can be used. The pitch detection algorithm was tested with both the variable-length template vectors and the constant-length template vectors, and the best results were achieved using the 6-dimensional constant-length vectors. In that case, each note template is represented by the 6 strongest components in its spectrum. Also, the constant-length templates perform better than the variable-length templates in the case where two notes an octave apart need to be detected.

To keep track of the original locations of the selected components in the vector $\mathbf{T}_m$, a second vector called the location vector, $\mathbf{L}_m = (l_{m,0},...,l_{m,5})$, is defined for each of the notes. The components, $l_{m,i}$, of the vector $\mathbf{L}_m$ are the locations of the 6 largest components of the mth note.

The measure $d_m(\mathbf{X},\mathbf{T}_m)$ needs to indicate how close the shapes of the involved vectors are. Therefore, it is necessary to normalize the vectors $\mathbf{X}$ and $\mathbf{T}_m$. The normalization of $\mathbf{T}_m$ is performed by dividing it by its maximal component, i.e.,

$$\tilde{\mathbf{T}}_m = \mathbf{T}_m / \max_i \{a_{m,i}\}. \tag{5.11}$$

Thus, each of the notes is represented by two 6-dimensional vectors, the normalized envelope vector $\tilde{\mathbf{T}}_m$ and the location vector $\mathbf{L}_m$.

In the pitch detection algorithm, the spectral content, $\mathbf{X}$, of the input signal at the particular attack is now compared with the note templates, $\tilde{\mathbf{T}}_m$, using the measure $d_m(\tilde{\mathbf{X}},\tilde{\mathbf{T}}_m)$ where

$$\tilde{x}_j = x_{l_{m,j}} \Big/ \max_j (x_{l_{m,j}}), \quad j = 0,...,5. \tag{5.12}$$

In other words, the template vector of the note m is compared to a 6-dimensional vector whose components are equal to the normalized components of the vector $\mathbf{X}$ located at $l_{m,0},...,l_{m,5}$.

The choice of the measure $d_m\left(\tilde{\mathbf{X}}, \tilde{\mathbf{T}}_m\right)$ is crucial in the template matching procedure. The standard mean square error (MSE) is not suitable since it does not take into account the absolute values of the vector components. Therefore, a modification of the standard MSE needs to be done where each of the components of the difference vector is normalized by the corresponding component of the vector $\tilde{\mathbf{X}}$, i.e.,

$$d_m\left(\tilde{\mathbf{X}}, \tilde{\mathbf{T}}_m\right) = \sum_{j=0}^{5} \frac{\left(\tilde{x}_j - \tilde{a}_j\right)^2}{\tilde{x}_j^2}. \tag{5.13}$$

Defined as in (5.13), the error measure becomes large when one or more of the components in the vector $\tilde{\mathbf{X}}$ are small. In this way, the false alarm errors during the detection are reduced since only the vectors without zero components will have a small error measure. Also, the detection of the octave apart notes is more reliable using this measure.

As mentioned earlier, the error measure (5.13) gives the information about how close the shapes of the spectral envelopes of the note templates and the input vector are. It means that even an input vector with very low components may have the small error measure defined in (5.13).

To include the energy level of the input signal into the decision procedure, the note energies are calculated from the input vector $\mathbf{X}$ as follows

$$NE_m = \sum_{j=0}^{5} x_{l_{m,j}}, \quad m = 0,...,71. \tag{5.14}$$

Thus, the final form of the similarity measure for the template matching procedure that takes into consideration both the spectral shape and the energy level, is given by

$$s_m\left(\tilde{\mathbf{X}},\tilde{\mathbf{T}}_m\right)=\frac{NE_m}{d_m\left(\tilde{\mathbf{X}},\tilde{\mathbf{T}}_m\right)}$$
$$=\frac{\sum_{j=0}^{5}x_{l_{m,j}}}{\sum_{j=0}^{5}\frac{\left(\tilde{x}_j-\tilde{a}_j\right)^2}{\tilde{x}_j^2}},\quad m=0,\ldots,71 \tag{5.15}$$

where $l_{m,j}$, $j = 0,\ldots,5$ are the components of the location vector $\mathbf{L}_m$.

The pitch detection algorithm calculates the similarity measure (5.15), and looks for the indices m where the measure has its maxima. Obviously, if there is only one maximum detected in the measure, then only one note with the index m exists at the particular attack. More than one peak in the similarity measure indicates the existence of a polyphonic attack. The question is: how to determine which peaks are valid and which are the result of the some disturbances or imperfections of the measure. To solve this problem, the pitch detection algorithm locates the global maximum in the similarity measure, and gives its index m as the first detected note. Then, the components of the input vector $\mathbf{X}$ with their locations defined by the location vector $\mathbf{L}_m$ are removed, and new similarity measures are calculated. If the remaining components in the input vector have low energy, then there are no more notes to be detected. However, if the remaining energy is significant compared to the initial input signal energy, then there exists at least one more notes.

To increase the robustness of the algorithm, an additional test on the global maximum in a multiple-peak case is imposed. To represent a valid note, the maximum in the similarity measure at index m needs to have a significant component in the position of the fundamental frequency of the note m. The locations of the fundamental frequencies are stored in a separate 72-element list and used in the test.

The most difficult scenario in the polyphonic music transcription is the simultaneous playing of two or more notes that are integer number octaves apart. This

algorithm offers an acceptable solution that can be improved by increasing the computational complexity of the algorithm. If a single note with the index m is played, the similarity measure (5.15) will have its local maxima at the indices m, m+12, m+24,.... Because of the way the error measure in (5.13) is defined, there will be no significant peaks at the locations m-12, m-24,.... The algorithm then takes the lowest index m among the set of candidates $\{m, m+12, m+24,\ldots\}$ as the index of an existing note. In the next step, the spectral content of the note is removed from the input vector, and the process repeats until all the notes at the particular attack are detected. Then, the complete algorithm repeats for all previously detected attacks. After all the notes in the input signal are detected, the corresponding note amplitudes determined by (5.14) are assigned to the detected notes. Figure 5.16 shows the block diagram of the pitch detection algorithm described above.

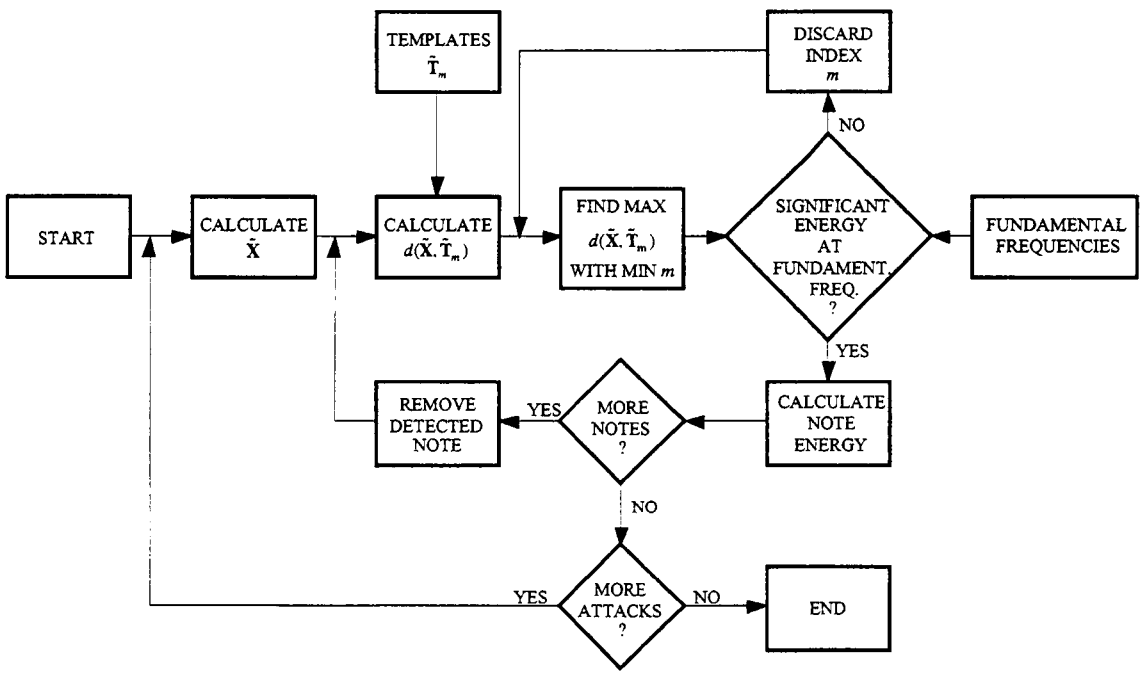


Figure 5.16 Block diagram of the pitch detection algorithm.

A LabVIEW program that implements the pitch detection algorithm is given in Appendix D. The algorithm is applied to the test inputs represented by Figures 5.2 to 5.8. The outputs of the pitch detection algorithm for the corresponding input signals are given in Figures 5.17 to 5.23, and they represent the computer generated scores for each of the inputs. All of the notes have the same duration, since the actual duration of the notes are not known at the time.

As Figures 5.17 to 5.22 show, the outputs of the automated pitch detection algorithm for synthesized inputs were almost error-free, i.e., only one error occurred in Figure 5.21. However, in Figure 5.23 the error rate is higher, and most of the errors are of the false alarm type. The cause of the increased number of errors is a higher noise level in the real-life recording, as well as a larger dynamic range. Also, the templates used in the detection procedure were made using the synthesized input signals so that the spectral envelopes of the templates may differ from the spectral envelopes of the notes played on a particular piano.

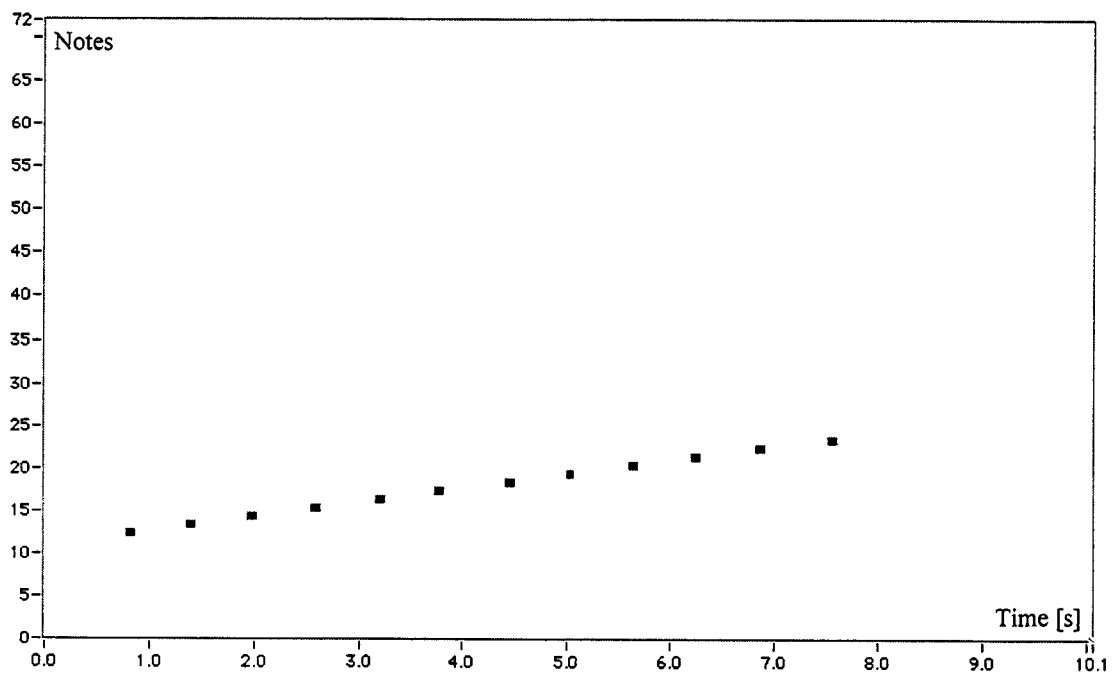


Figure 5.17 Output of the pitch detection algorithm for the input signal of Figure 5.2.

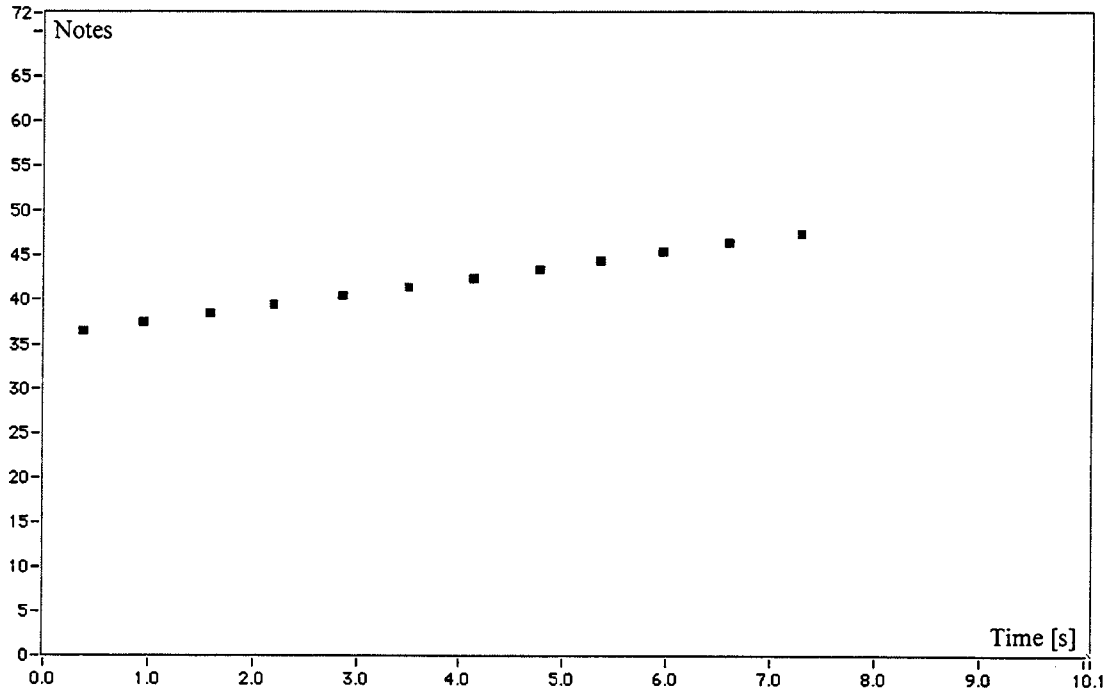


Figure 5.18 Output of the pitch detection algorithm for the input signal of Figure 5.3.

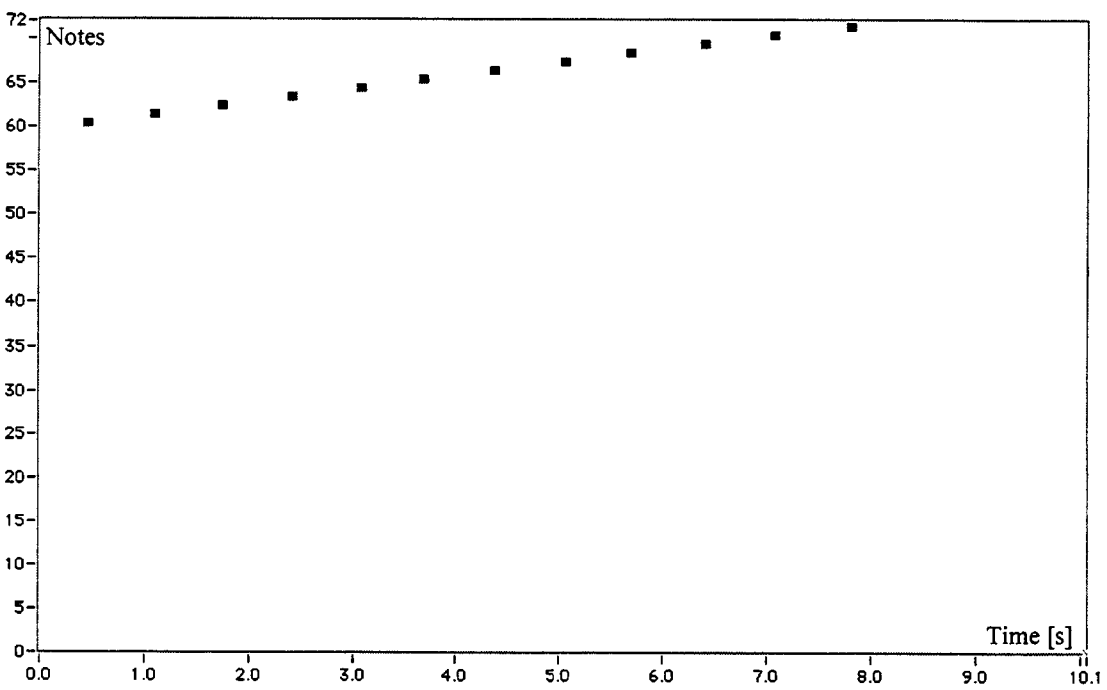


Figure 5.19 Output of the pitch detection algorithm for the input signal of Figure 5.4.

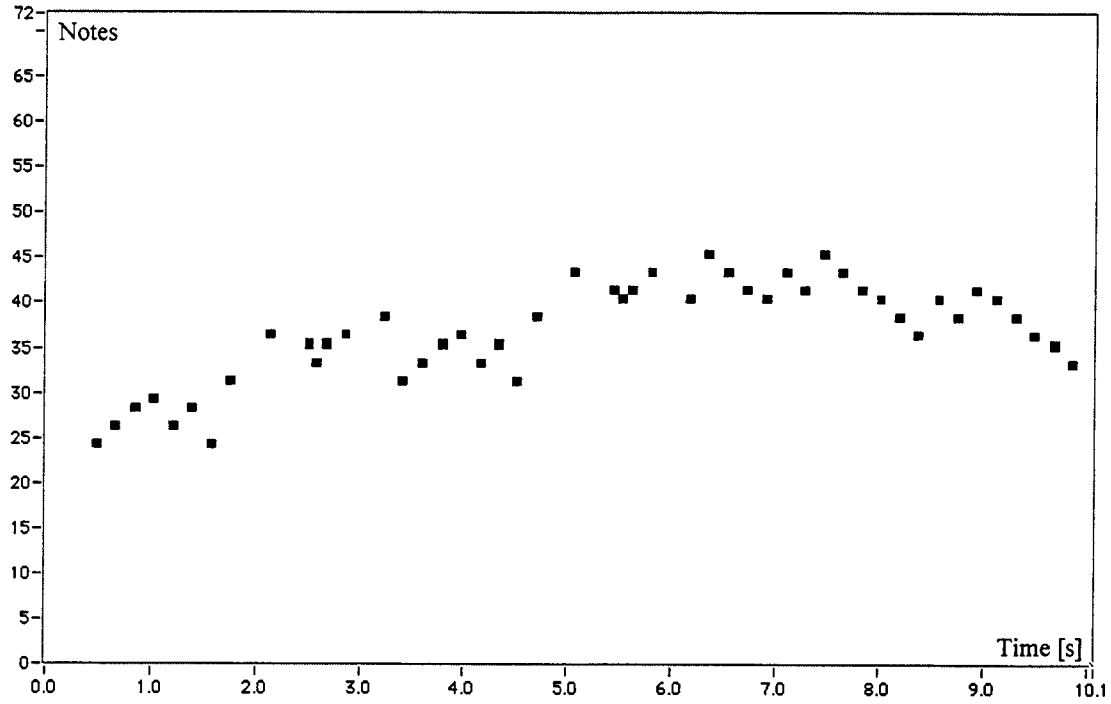


Figure 5.20 Output of the pitch detection algorithm for the input signal of Figure 5.5.

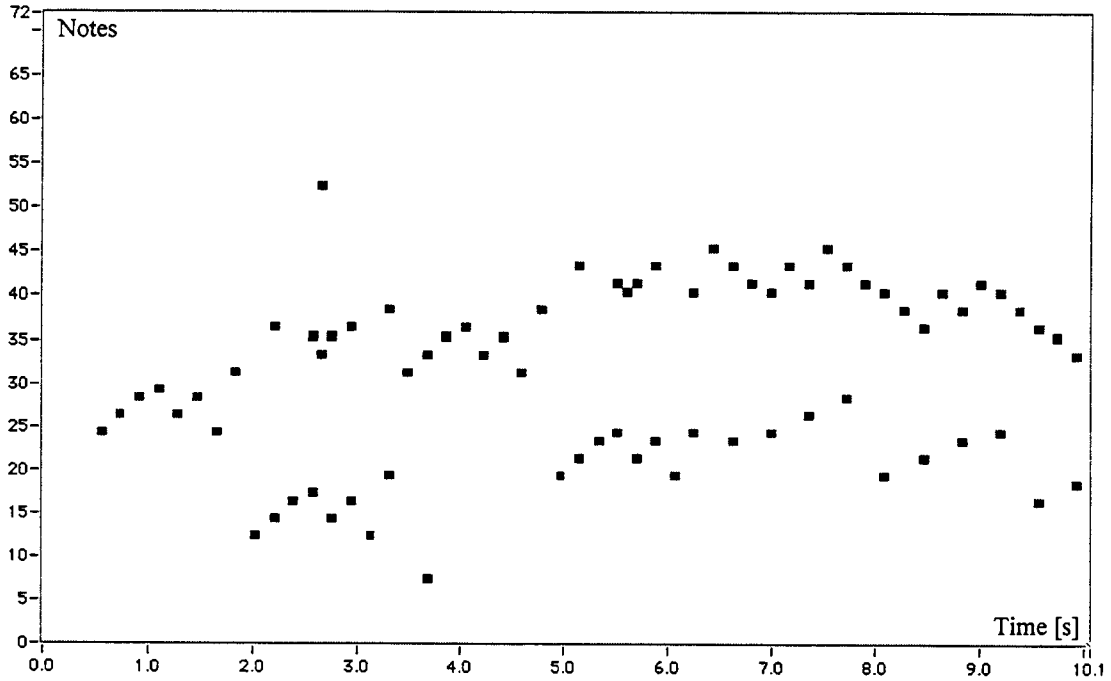


Figure 5.21 Output of the pitch detection algorithm for the input signal of Figure 5.6.

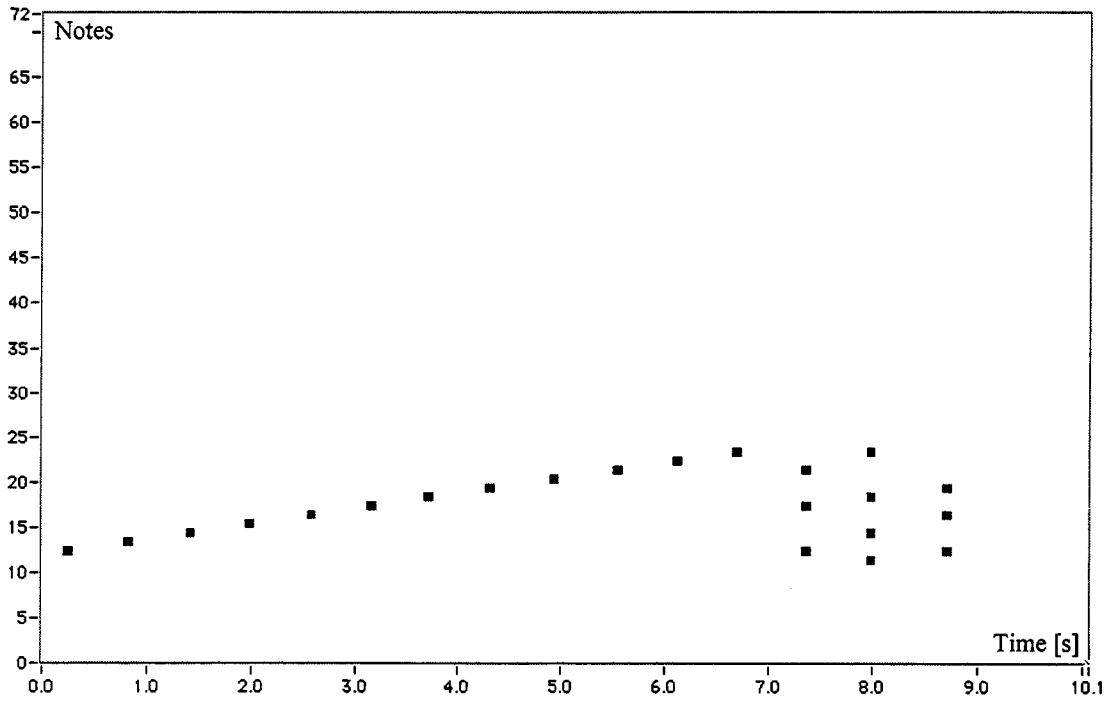


Figure 5.22 Output of the pitch detection algorithm for the input signal of Figure 5.7.

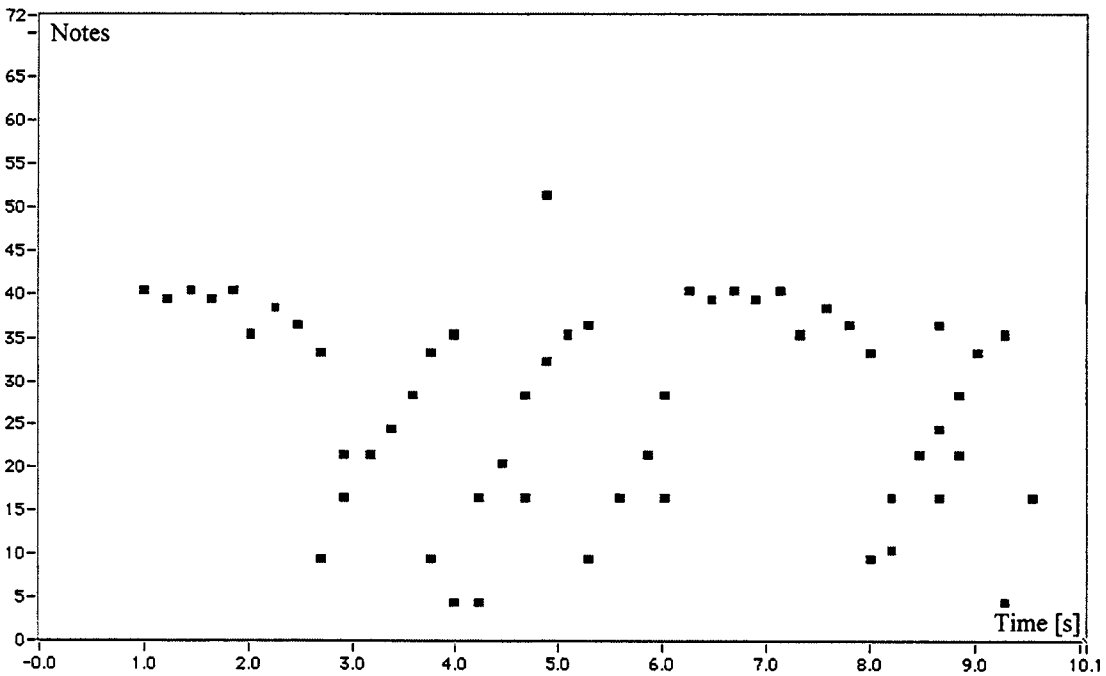


Figure 5.23 Output of the pitch detection algorithm for the input signal of Figure 5.8.

To eliminate the errors in the algorithm for the real-life recording, an interactive feature is added to the pitch detection algorithm. Similarly to the interactive algorithm for the attack detection, one can add or remove notes, thus correcting all of the errors occurring after the automated part of the algorithm.

In this subsection, the pitch detection algorithm for polyphonic piano music was developed. It was shown that an automated algorithm performed well for the synthesized input signals. For real-life recordings, an interactive algorithm that allowed correction of the detection errors was developed

5.2.4 Detection of the Note Duration

The duration detection algorithm is based on the energy estimation for each of the previously detected notes. As mentioned at the beginning of the chapter, piano notes do not have a sustained part. They decay immediately after the attack, and if the key is held down, there is an uncertainty about the exact end of the note. However, if the key is released before the note energy vanishes completely, then a distinct energy drop exists thus helping the detection of the note end.

The place where the note ends, D, is determined as the place where the short-time note energy drops to 5% of its value at the attack. The short-time note energy is calculated according to

$$STE(n) = \sum_{i=l_0}^{l_5} \sum_{j=L(i,n)}^{K(i)} C^2(i,j), \quad n = 0,...,D \tag{5.16}$$

where D is the smallest integer that satisfies

$$STE(D) < 0.05 \times STE(0). \tag{5.17}$$

The limits in the summation in (5.16), $K(i)$ and $L(i,n)$, were defined in (5.6).

The above described algorithm was applied to the input signals represented in Figures 5.2 through 5.8 using the previously determined attacks and pitches. Figures 5.24 trough 5.30 show computer generated scores of the corresponding piano pieces. The

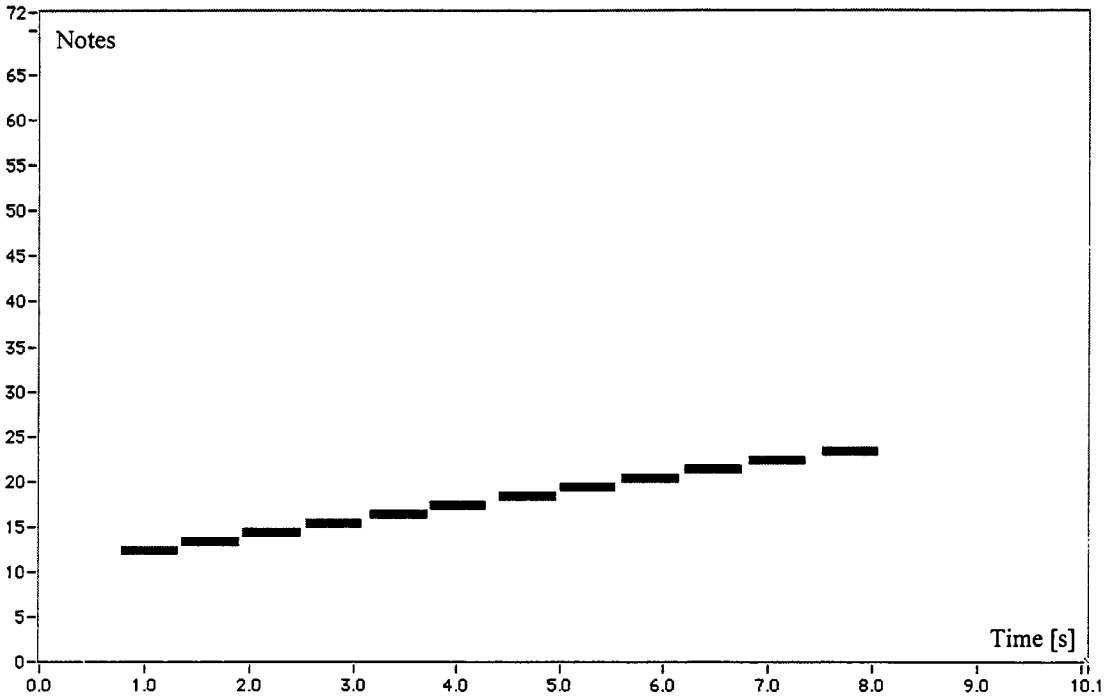


Figure 5.24 Computer generated scores for the input signal of Figure 5.2.

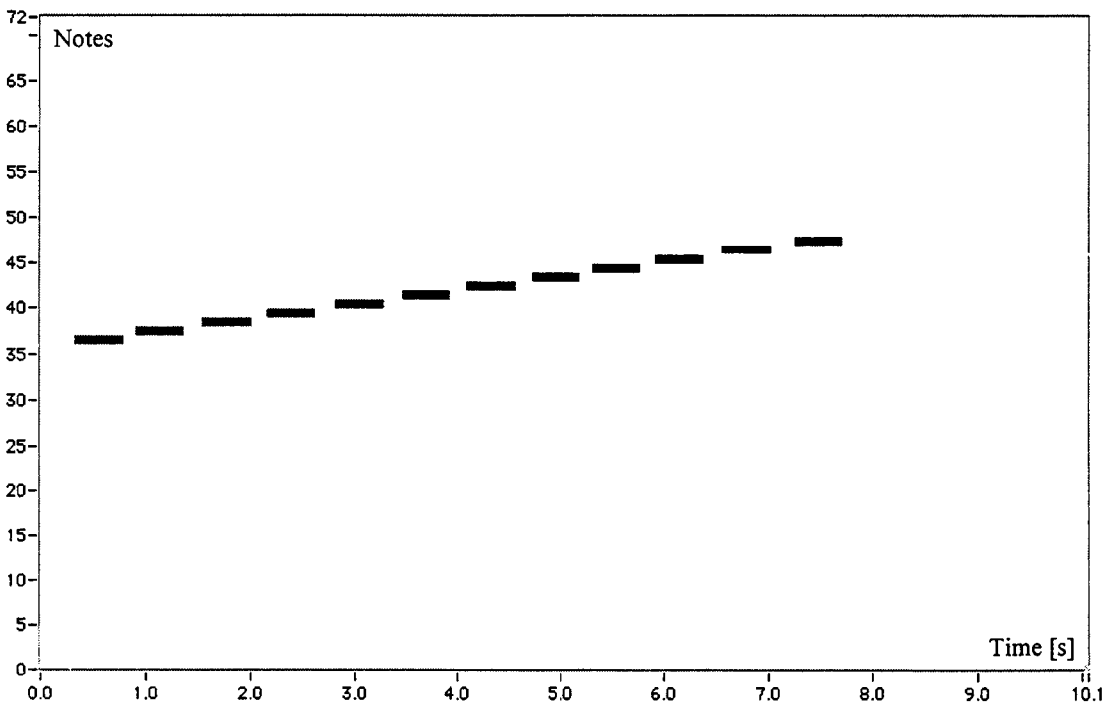


Figure 5.25 Computer generated scores for the input signal of Figure 5.3.

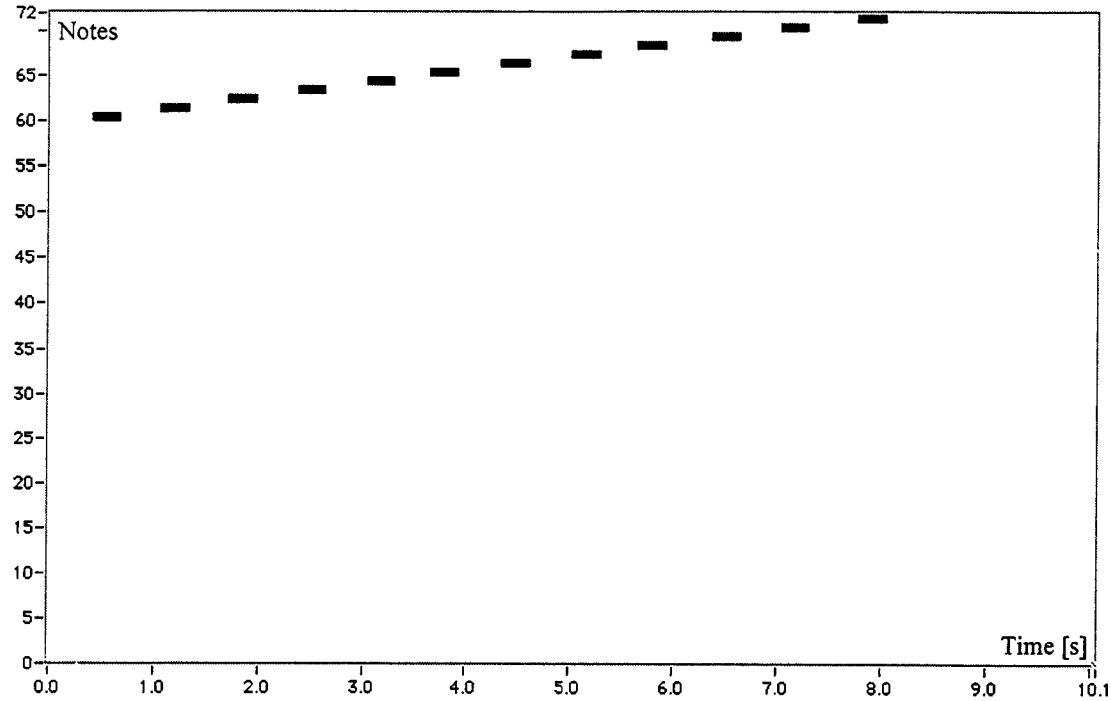


Figure 5.26 Computer generated scores for the input signal of Figure 5.4.

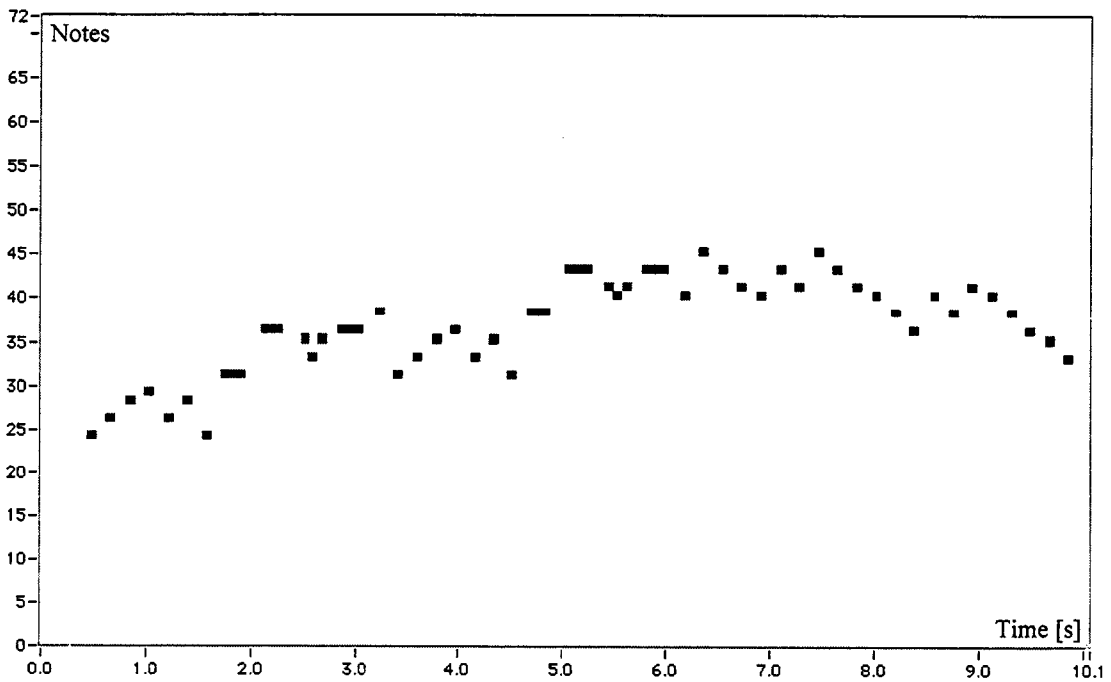


Figure 5.27 Computer generated scores for the input signal of Figure 5.5.

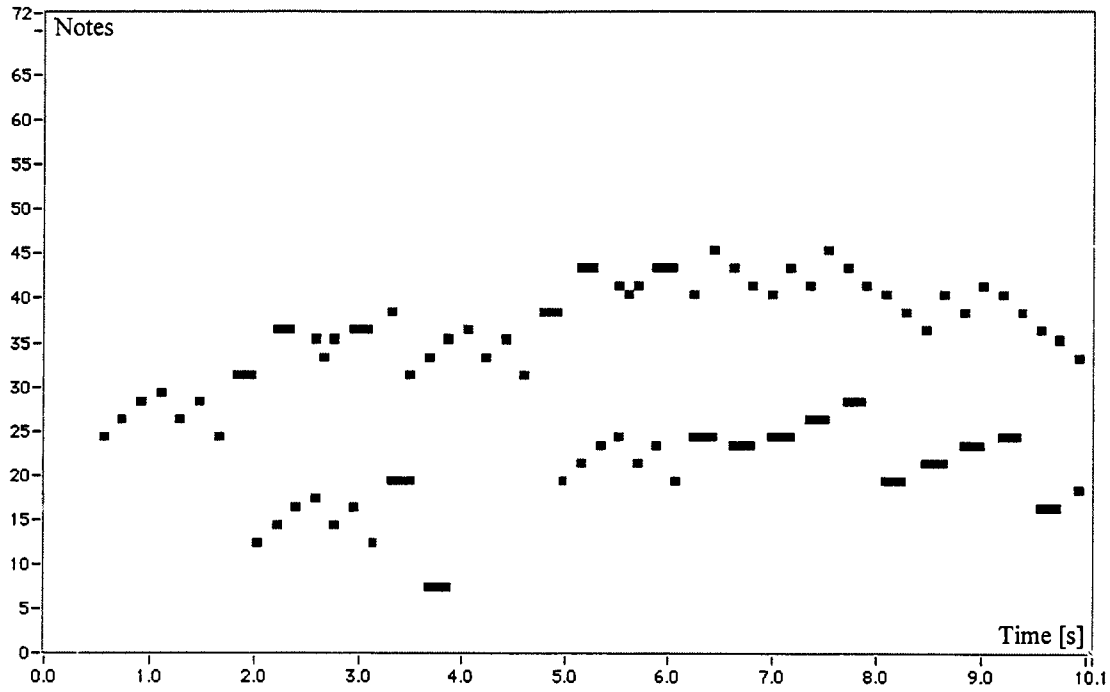


Figure 5.28 Computer generated scores for the input signal of Figure 5.6.

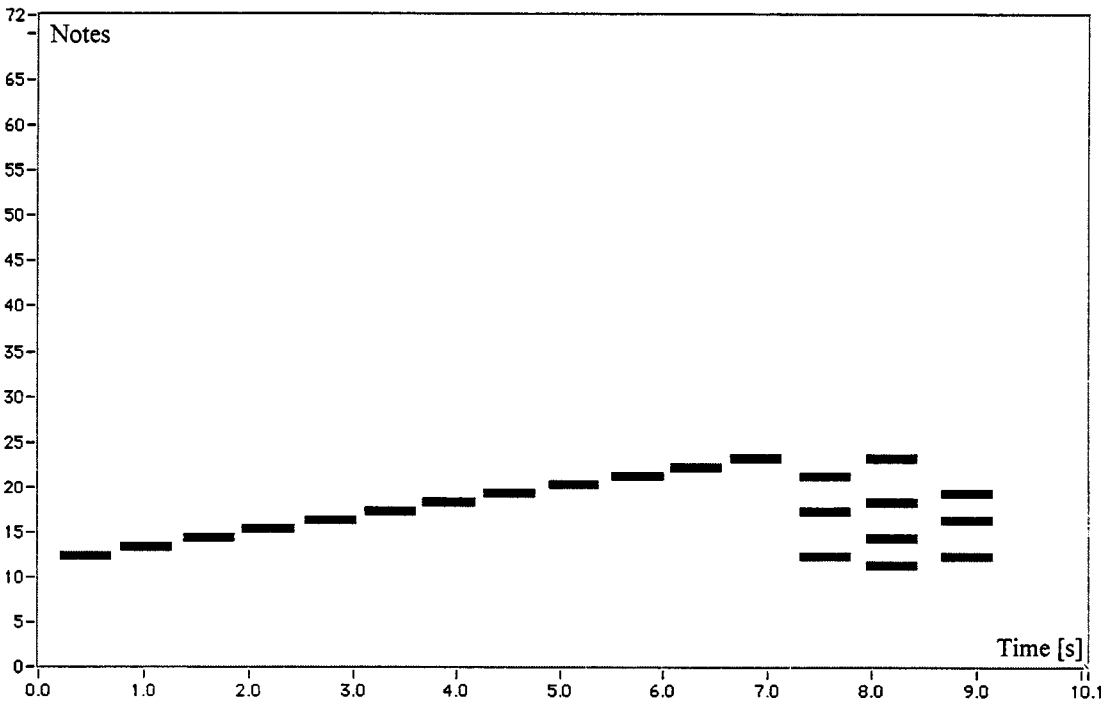


Figure 5.29 Computer generated scores for the input signal of Figure 5.7.

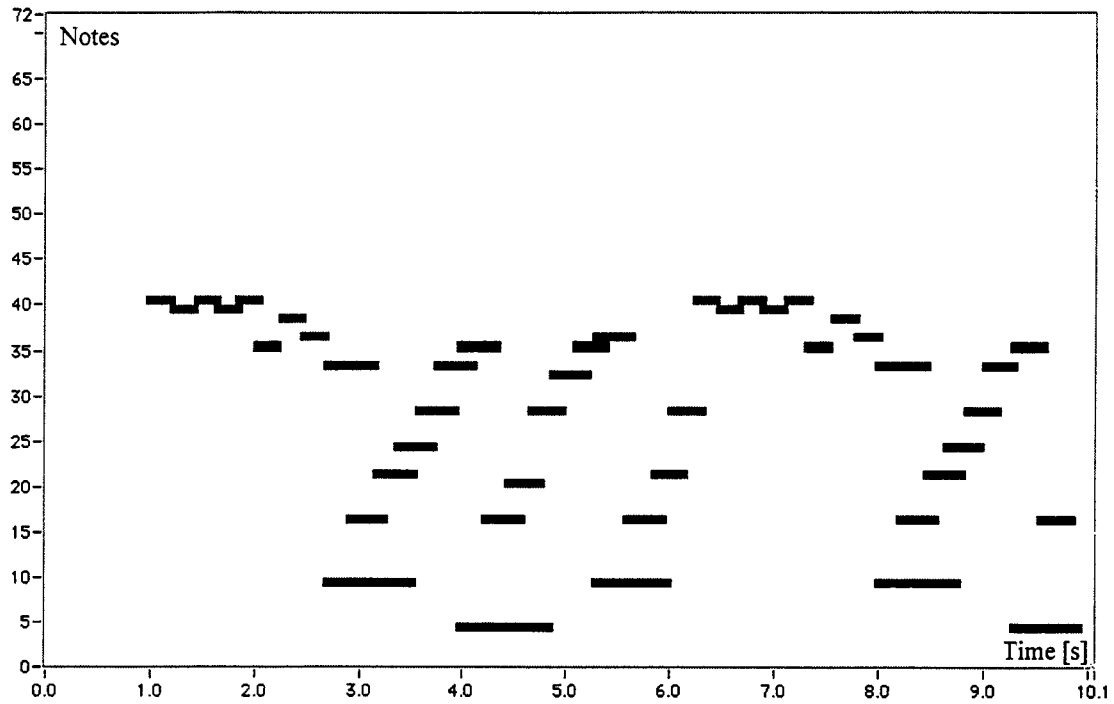


Figure 5.30 Computer generated scores for the input signal of Figure 5.8.

output of the algorithm is a file containing the attack times, pitches, amplitudes, and durations of the notes in the input signal. A LabVIEW program that implements the duration detection algorithm is given in Appendix E.

The duration detection algorithm shows some limitations that exist when very low notes need to be analyzed. As mentioned earlier, the time resolution at low frequencies is relatively poor, and there exists an uncertainty about the exact end of the note energy at lower subbands. Therefore, the algorithm was improved by adding the interactive part that allows editing of the durations of the notes. Thus, each of the notes can be extended or shortened to match the original. However, this requires the user to have some musical background.

The attack, pitch, amplitude, and note duration detection algorithms described in this section were used to detect notes in a piano piece, i.e., to determine their attacks, pitches, amplitudes, and durations. As shown in Figure 5.1, these features are coded into a MIDI stream allowing one to play back the input signals on a MIDI driven instrument.

5.2.5 MIDI Coding

After the notes in a piano piece are detected, their features are coded into a MIDI stream. The MIDI data are organized in a so-called Standard MIDI File (SMF). The SMF contains two data chunks; the header chunk and the track chunk. The header chunk is a 14-byte long data block that specifies the SMF format, the number of tracks, and the timing resolution per quarter note. The track chunk contains the actual data representing a particular music piece. It is divided into track events that may be either so-called meta-events describing the track name, timing, and key signature; or MIDI messages representing the most important characteristics of a note. A MIDI message contains up to eight bytes; up to four bytes represent the delta time (the time elapsed after the previous message given in so-called clock ticks), one byte represents note on or note off, one byte defines the pitch, and one byte defines the note amplitude. A detailed description of the SMF structure can be found in [91].

To convert previously determined features of the input piano pieces into MIDI files, a LabVIEW program given in Appendix F was written and implemented. As an example, Figure 5.31 shows the TurboTrax scores for the real-life piano recording given in Figure 5.8.

Once a piano piece is transformed into the Standard MIDI File, it can be played back on any MIDI driven instrument or sound module. In this dissertation, the corresponding MIDI files were processed by TurboTrax, a sequencer software, and played back on PROformance Plus, a piano sound module made by E-mu Systems. This allowed subjective tests to be performed on the played back pieces. The tests showed that the synthesized input signals were exactly reproduced when playing back their MIDI files. The MIDI file for the real-life recording reproduced onset times, pitches, and amplitudes correctly. However, the above mentioned uncertainty in the duration detection caused small subjectively noticeable differences in the reproduced music. As mentioned above, durations of the notes can be corrected afterwards using the interactive

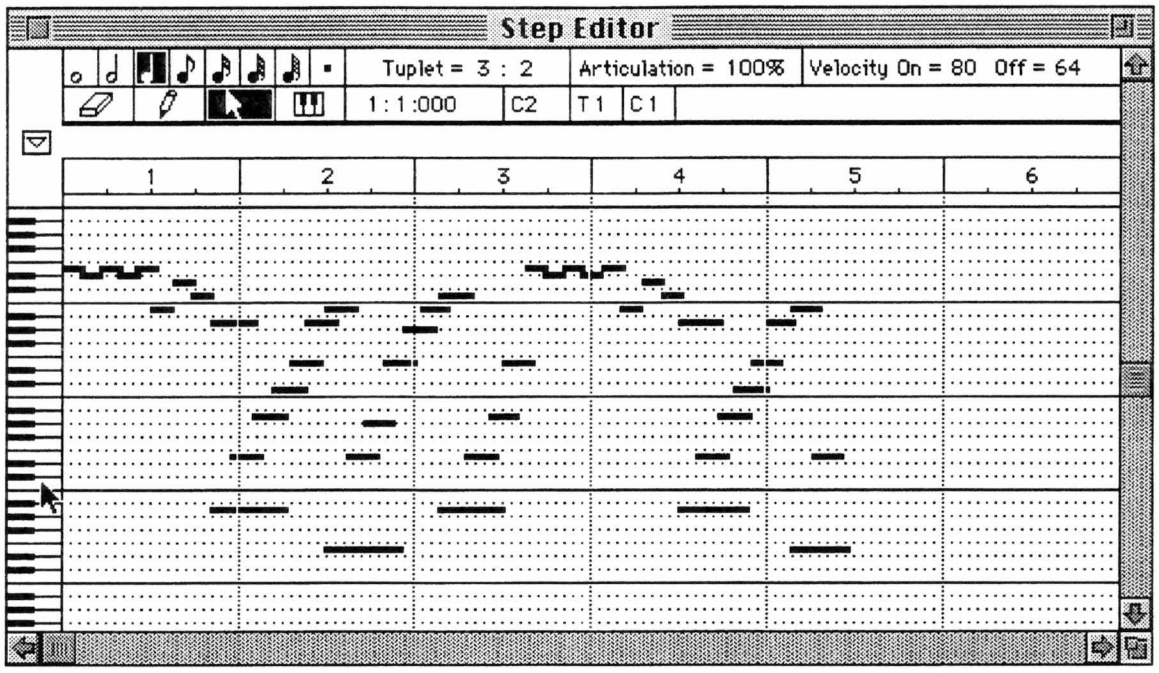


Figure 5.31 TurboTrax scores for the first 10 seconds of the real-life recording of Beethoven's *Für Elise*.

algorithm for the note duration detection.

There are many advantages offered by the MIDI representation of a piano piece. It allows one to play back the original input music slower or faster, with different voices, or transposed to a different key signature. Furthermore, the MIDI coding performs a large compression of the input music signal making MIDI files very attractive for music transfer and storage. In the above presented examples, the compression ratio was 500:1.

5.3 Piano Voice Separation and Denoising

The following implementation of the tree-structured filter banks is based on the small reconstruction error in the analysis/synthesis structure. As shown in Figure 4.3, the reconstruction error is below the round-off error if the 256-tap QMFs are used in the structure. This makes it possible to use the analysis/synthesis structure for separation of voices in a music piece, or for the removal of different kind of distortions or noise from a

music recording.

Voice separation is based on the fact that the tree-structured filter banks have good time-frequency resolution. Thus, each note in the time-frequency space is well localized allowing one to remove the time-frequency coefficients belonging to a particular note without affecting any other coefficients. As shown in Chapter 4, there may be some overlapping of the overtones making separation a difficult task. Furthermore, besides the frequency overlapping, there may be some overlapping in time as well, especially at lower frequencies where the time resolution is lower. Even though there exist some limitations, the tree-structured filter banks are the only orthogonal structures that have good time-frequency resolution over a wide range of frequencies.

For voice separation in polyphonic piano music, a LabVIEW program given in Appendix G was written and implemented. The algorithm takes the time-frequency coefficients of the input signal, and using an editor, sets to zero all time-frequency coefficients belonging to the note that needs to be removed. Therefore, the frequency structure of each of the notes needs to be known beforehand. After the time-frequency space has been edited, the synthesis part of the tree-structured filter banks was applied to get the edited signal in the time domain. The synthesis part of the tree-structured filter bank of Figure 4.1 (b) was implemented in LabVIEW using the program given in Appendix H.

The voice separation algorithm was applied to the real-life recording whose time-frequency representation is given in Figure 5.8. Figure 5.32 shows this real-life signal in the time domain. As mentioned above, the real-life recording represents ten seconds of the Beethoven's piano piece *Für Elise*. Using the voice separation algorithm, the left-hand voice has been removed from the piece.

The original recording was recorded and played back on poor-quality equipment, and a relatively high level of noise was captured during the signal acquisition. The

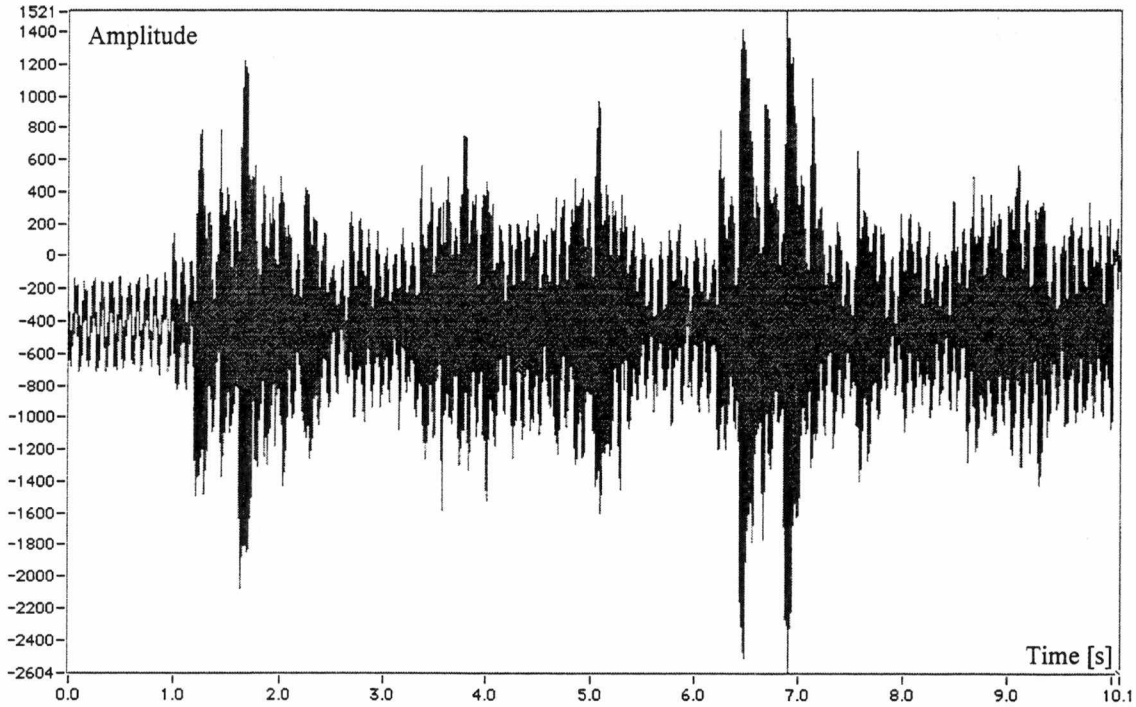


Figure 5.32 The real-life piano recording of the first 10 seconds of Beethoven's *Für Elise* in the time domain.

estimated signal-to-noise ratio of the input signal was about 20 dB. Therefore, the remaining right-hand voice was further processed to reduce the noise.

The following procedure was used to reduce the noise. As Figure 5.8 shows, there is no significant signal energy in the top 50 subbands of the time-frequency representation of the input signal. This means that the highest signal frequency is lower than 4.2 kHz according to Table 5.1. Therefore, the time-frequency coefficients in the top 50 subbands were set to zero. Knowing that the lowest note in the right-hand voice is C_3, and that the lowest frequency in this note corresponds to the 45^{th} subband, all time-frequency coefficients in the 44 lowest subbands were set to zero as well. Thus, all the signal energy below 264 Hz was removed. Further noise reduction can be done setting to zero all the coefficients corresponding to the first second of the time-frequency space since the first note starts 0.965 seconds after the beginning of the signal.

The above described "cut-off" procedure cannot be applied in the time-frequency

space where note energy is present. Therefore, for further noise reduction, the *thresholding* technique [6] was applied. According to the technique, all the time-frequency coefficients below a threshold are set to zero. The threshold needs to be carefully selected so that the maximum noise energy is removed while keeping an acceptable quality level of the reproduced music. If the threshold is too high, the reproduced piano piece tends to sound like trumpet music since the piano attacks are then blurred by the thresholding.

Figure 5.33 shows the time-frequency representation of the real-life recording after the left-hand notes were removed and the above described denoising procedure applied.

The edited time-frequency representation of the real-life recording was then transformed back to the time domain using the synthesis part of the tree-structured filter banks given in Figure 4.1 (b). Figure 5.34 shows the time-domain signal of the right-hand voice of the original recording.

The noise level in the reproduced signal is now reduced by about 10 dB. The listening tests of the separated piano piece showed that that left-hand voice was completely removed while the right-hand voice was reproduced with all the nuances of the original piece. Also, the perceived noise level was significantly lower in the reproduced single voice piece.

In this chapter, two different applications of the tree-structured filter banks were presented. In the first one, the tree-structured filter banks were used as the front-end signal processing in a polyphonic music transcription system. Besides the front-end signal processing, several algorithms for the extraction of the most important note features were developed and applied. At the end, these features were coded into the MIDI stream allowing one to play back the original piano piece. Besides all the advantages offered by MIDI for editing and playing back the original input music, MIDI files offer a significant compression of music signals.

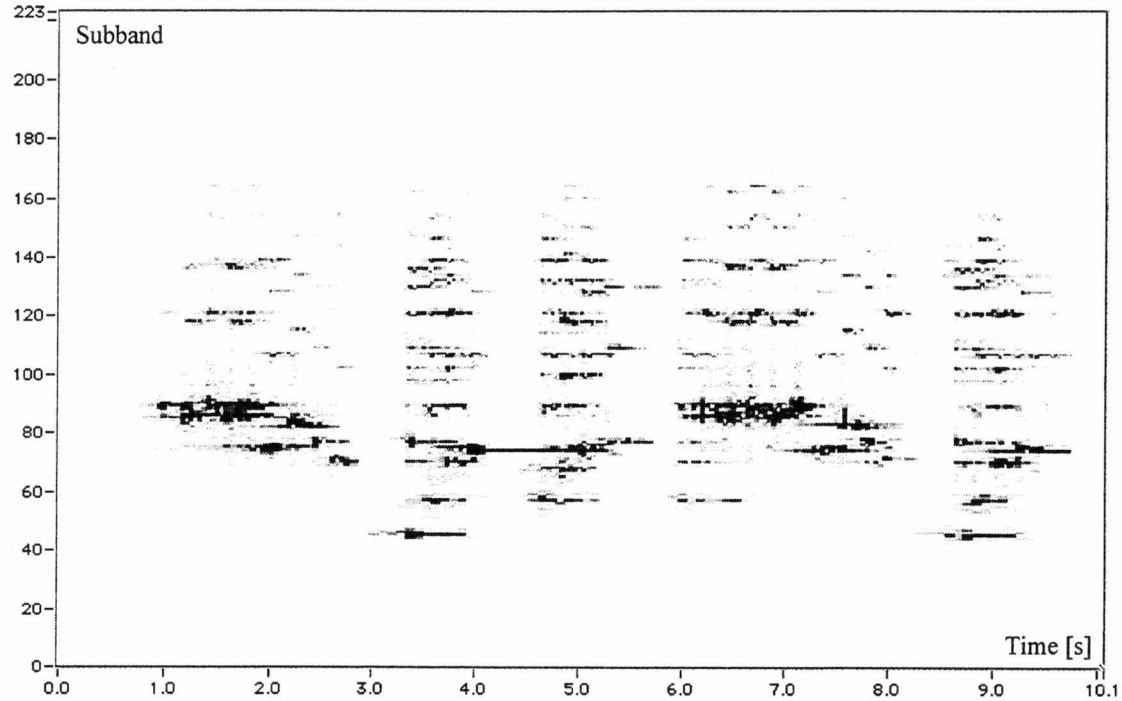


Figure 5.33 Time-frequency representation of the real-life recording after the left-hand voice was removed.

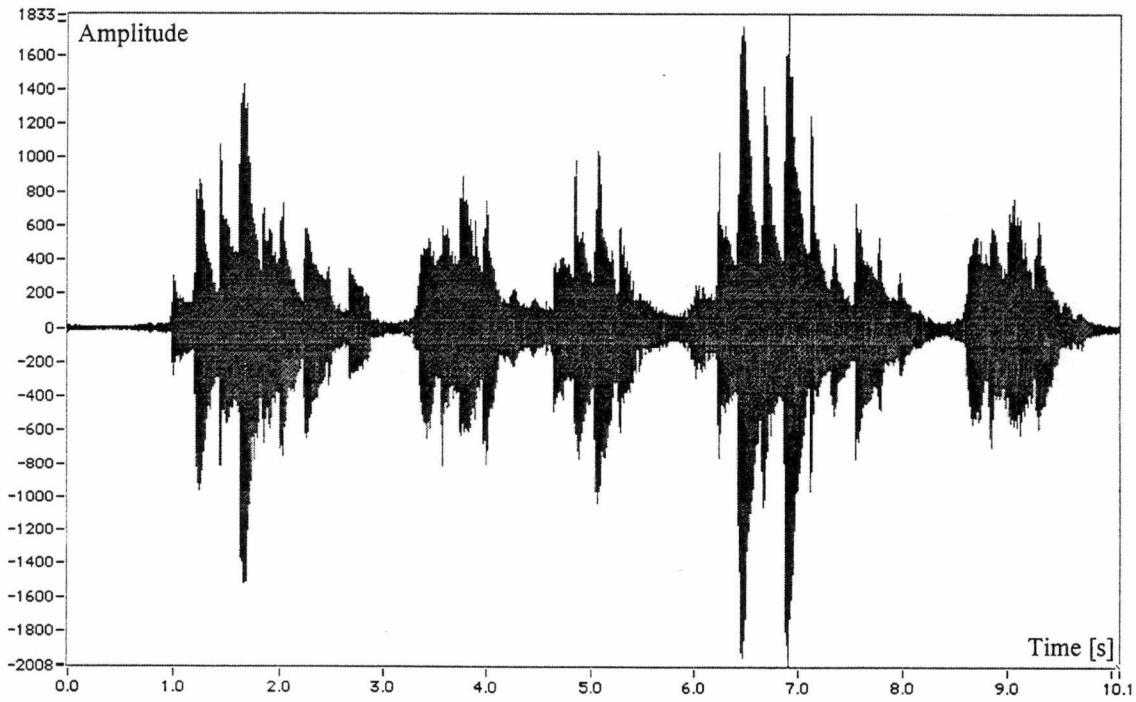


Figure 5.34 Right-hand voice of the first 10 seconds of Beethoven's *Für Elise* in the time domain.

The polyphonic music transcription algorithm was tested using several different synthesized input signals, as well as a real-life piano recording. It was shown that the algorithm can be automated for the synthesized inputs transcribing the music with a small error rate. For real-life recordings an interactive algorithm needs to be used instead, to correct the errors occurring in the automated part of the algorithm.

In the other application, the tree-structured filter banks were used for the voice separation in polyphonic piano music. While in the first application only the analysis part of the tree-structured filter banks was used, for the voice separation both the analysis and the synthesis structures were implemented. Also, it is necessary that the whole structure has a low reconstruction error. In the application, a two-voice, real-life recording was processed, and the right-hand voice was separated from the left-hand voice. Besides the separation, the noise reduction in the original piece was achieved.

CHAPTER 6

CONCLUSIONS

In this chapter, a summary of the dissertation as well as the contributions in the area of time-frequency representation of audio signals is presented. Also, future research directions in the area are recommended.

6.1 Summary and Contributions

For front-end signal processing in polyphonic music analysis, many different signal processing techniques were used so far. They range from bandpass filtering to different forms of the STFT. However, the STFT shows some limitations when used in segmentation of music signals since they have logarithmic distribution of the signal spectrum. A constant-bandwidth analysis obtained by the STFT does not correspond to the constant-Q structure of music signals. Furthermore, there is no orthogonal STFT with good time-frequency resolution.

The Wavelet Transform offers a constant-Q analysis of the input signal, but the Q-factor is too small for use in music signal segmentation. Contrary to the STFT, it is possible to have an orthogonal wavelet transform that has a good time-frequency localization. As shown in the dissertation, the Wavelet Transform may be implemented using an octave-band filter bank that uses a perfect reconstruction filter pair.

Both the STFT and the Wavelet Transform are special cases of general tree-structured filter banks. While the STFT corresponds to the full-tree structure, the Wavelet Transform is performed using the octave-band filter bank. An arbitrary tree structure corresponds to the wavelet packets, and it may split the signal spectrum in many different ways while retaining the orthogonality of the structure.

In this dissertation, tree-structured filter banks for the front-end signal processing of music signals were designed and implemented. It was shown that sufficient frequency resolution over almost seven octaves can be achieved using an 11-stage filter bank.

To reduce frequency leakage in the tree-structure, a new class of high-quality QMFs was developed and implemented. The new QMFs are designed using an iterative procedure that converges fast and makes it possible to design filters with high orders. Techniques for the design of QMFs and perfect reconstruction filters used so far are based on nonlinear optimization and suffer from divergence if longer filters need to be designed.

Even though the new QMFs do not have a perfect reconstruction property, the reconstruction error of the tree-structured filter bank that use these QMFs is very small and lower than the round-off error.

In the dissertation, special attention is paid to implementation issues and the computational complexity of tree-structured filter banks. It was shown that the computational complexity can be significantly reduced if the QMFs are represented by their polyphase components. Furthermore, if FFT-based convolution is applied in the tree structure, then the computational complexity reduces even more.

The tree-structured filter banks were implemented in two different applications. The first one is the transcription of polyphonic piano music where tree-structured filter banks were used for the front-end signal processing. After the input signals were transformed into the time-frequency space, the time-frequency coefficients were used for feature extraction. In this dissertation, an original method for attack detection was designed and implemented . At the end of the transcription system, the features were coded into MIDI streams making it possible to compare the transcribed music with the original input music. Input signals in the application were synthesized piano pieces as well as a real-life piano recording. It was shown that an automated transcription system

performs well for synthesized input signals. For real-life recordings, human intervention was necessary using an interactive algorithm for attack, pitch and duration detection.

In the second application, an editor for the time-frequency space was developed and implemented. Here, both the analysis and the synthesis part of the tree-structured filter banks were implemented. Using the editor, the left-hand voice was removed from a two-voice piano recording so that only the right-hand voice was left in the original recording. Also, the editor was used to remove noise as well as other disturbances from the real-life recording.

These two application have shown that the tree-structured filter banks are a very effective tool for the segmentation of music signals. They have sufficient frequency resolution over a wide range of signal frequencies, and sufficient time resolution at high frequencies. Their small reconstruction error makes them attractive for use in music signal enhancement and coding.

6.2 Recommendations for Future Research

In conclusion of this dissertation, a few areas in which further research and experimentation may prove to be productive are outlined below.

- Implementation of tree-structured filter banks for separation of human voice and instrumental music, or for removal of audience noise from a life recording should be examined.
- Use of multiband filters instead of two-channel filters in the filter banks for music signal analysis should be investigated.
- Hybrid structures for music signal segmentation that combine two-channel filters, multiband filters, and the standard FFT processing might offer some improvements in the implementation performances and the reconstruction error.

- Two-dimensional tree-structured filter banks are possible. Their use for the processing of images where good resolution in both dimensions is needed should be investigated.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] S.V. Vaseghi, R. Frayling–Cork, "Restoration of Old Gramophone Recordings", *J. Audio Eng. Soc.* vol. 40, pp. 791–800, October 1992.

[2] S.F. Boll, "Suppression of Acoustic Noise in Speech Using Spectral Subtraction", *IEEE Trans. Acoust., Speech, Signal Processing,* vol. ASSP–27, pp. 113–120, April 1979.

[3] M.R. Sambur, "Adaptive Noise Canceling for Speech Signals", *IEEE Trans. Acoust., Speech, Signal Processing,* vol. ASSP–26, pp. 419–423, October 1978.

[4] O. Cappe, J. Laroche, "Evaluation of Short–Time Spectral Attenuation Techniques for the Restoration of Music Recordings", *IEEE Trans. Speech Audio Proc.,* vol. 3, pp. 84–93, January 1995.

[5] O. Cappe, "Elimination of the Musical Noise Phenomenon with the Ephraim and Malah Noise Suppressor", *IEEE Trans. Speech Audio Proc.,* vol. 2, pp. 345–349, April 1994.

[6] J. Berger, R.R. Coifman, M.J. Goldberg, "Removing Noise from Music Using Local Trigonometric Bases and Wavelet Packets", *J. Audio Eng. Soc.* vol. 42, pp. 808–818, October 1994.

[7] O. Cappe, "Enhancement of Musical Signals Degraded by Background Noise, using Long–Term Behavior of the Short–Term Spectral Components", *Proc. IEEE ICASSP* pp. I–217–I–220, 1993.

[8] R.C. Maher, "A Method for Extrapolation of Missing Digital Audio Data", *J. Audio Eng. Soc.* vol. 42, pp. 350–356, May 1994.

[9] G. Stoll, Y.F. Dehery, "High Quality Audio Bit Rate Reduction System Family for Different Applications", *Proc. Supercomm. '90,* Atlanta, GA, 1990.

[10] R.N.J. Veldhius, "Bit Rates in Audio Source Coding", *IEEE J. Select. Areas Comm.* Vol. 38, pp. 747–765, March 1992.

[11] P. Voros, "High Quality Sound Coding with 2 x 64 kb/s using Spontaneous Dynamic Bit Allocation", *Proc. ICASSP '88,* pp. 2536–2539.

[12] G.C.P. Lokhoff, "Precision Adaptive Subband Coding (PASC) for the Digital Compact Cassette (DCC)", *IEEE Trans. Consumer Electronics,* vol. 38, pp. 784–789, November 1992.

[13] J.D. Johnston, "Transform coding of Audio Signals using Perceptual Noise Criteria", *IEEE J. Select. Areas Comm.* vol. 6, pp. 314–323, 1988.

[14] J.D. Johnston, "Perceptual Transform Coding of Wideband Stereo Signals", *Proc. ICASSP '80,* pp. 1993–1996.

[15] D. Sinha, A.H. Tewfik, "Low Bit Rate Transparent Audio Compression using Adapted Wavelets", *IEEE Trans. Signal Processing,* vol. 41, pp. 3463–3481, December 1993.

[16] G. Stoll, F. Dehery, "High Quality Audio Bit Rate Reduction Family for Different Applications", *Proc. IEEE Int. Conf. Commun.* pp. 937-941, April 1990.

[17] M. Bosi, G. Davison, "High-Quality, Low-Rate Audio Transform Coding for Transmission and Multimedia Applications", *Convention of the AES,* San Francisco, CA, October 1992.

[18] P. Marmelstein, "G.722, A New CCITT Coding Standard for Digital Transmission of Wideband Audio Signals", *IEEE Comm. Mag.,* 8(15), 1988.

[19] S. Singhal, "High Quality Audio Coding using Multipulse LPC", *Proc. ICASSP,* pp. 1101–1104, April 1990.

[20] R.B. Dannenberg, "Music Representation: Issues, Techniques, and Systems", *Computer Music Journal,* vol. 17, pp. 20–30, Fall 1993.

[21] J. Laroche, "Musical Signal Spectrum Analysis with the Matrix Pencil Method", *J. Acoust. Soc. Am.,* vol. 95, no. 5, pt. 2, May 1994.

[22] J. Laroche, "A New Analysis/Synthesis System of Musical Signals Using Prony's Method. Application to Heavily Damped Percussive Sounds", *Proc. ICASSP,* pp. 2053-2056, 1989.

[23] R. Roy, A. Paulraj, T. Kailath, "Estimation of Signal Parameters via Rational Invariance Techniques – ESPRIT", *IEEE Trans. Acoust., Speech, Signal Processing,* vol. ASSP–34, pp. 1340–1342, 1986.

[24] R. Kumaresan, D.W. Tufs, "Estimating the Parameters of Exponentially Damped Sinusoids and Pole–Zero Modeling in Noise", *IEEE Trans. Acoust., Speech, Signal Processing,* vol. ASSP–30, pp. 833–840, 1982.

[25] J. Strawn, "Analysis and Synthesis of Musical Transitions Using the Discrete Short–Time Fourier Transform", *J. Audio Eng. Soc.,* vol. 35, pp. 3–13, Jan/Feb. 1987.

[26] X. Serra, J. Smith, "Spectral Modeling Synthesis: A Sound Analysis/Synthesis System Based on a Deterministic plus Stochastic Decomposition", *Computer Music Journal,* vol. 14, pp. 12–24, Winter 1990.

[27] D. Arfib, N. Delprat, "Musical Transformations Using the Modification of Time–Frequency Images", *Computer Music Journal,* vol. 17:2, pp. 66–72, Summer 1993.

[28] X. Yang, K. Wang, S.A. Shamma, "Auditory Representations of Acoustic Signals", *IEEE Trans. Inf. Theory,* vol. 38, pp. 824–839, March 1992.

[29] D.E. Newland, "Harmonic Wavelet Analysis", *Proc. R. Soc. Lond. A,* vol. 443, pp. 203–225, 1993.

[30] D.E. Newland, "Harmonic Wavelet Analysis", *Proc. R. Soc. Lond. A,* vol. 444, pp. 605–620, 1994.

[31] J.M. Chowning, B. Mont-Reyond, "Intelligent Analysis of Composite Acoustic Signals", *STAN-M-36,* Stanford, California, May 1986.

[32] B. Mont-Reyond, D.K. Mellinger, "Source Separation by Frequency Co-modulation", *Proc. First Int. Conf. on Music Perception and Cognition,* Kyoto, Japan, October 1988.

[33] C. Chafe, D. Jaffe, "Source Separation and Note Identification in Polyphonic Music", *Proc. of IEEE ICASSP Conference,* Tokyo, Japan, April 1986.

[34] B. Mont Reyond, "Pattern Recognition Problems in Music", *AI East Conference '87,* Atlantic City, 1987.

[35] A.L. Wang, "Instantaneous and Frequency-Warped Signal Processing Techniques for Auditory Source Separation", Ph.D. dissertation, Stanford University, Stanford, August 1994.

[36] J.A. Moorer, "On the Segmentation and Analysis of Continuous Musical Sounds by Digital Computer", Technical Report STAN–M–3, Stanford University, Department of Music, 1975.

[37] J.A. Moorer, "On the Transcription of Musical Sound by Computer", *Computer Music Journal,* November 1977.

[38] M. Piszczalski, B.A. Galler, "Automatic Music Transcription", *Computer Music Journal,* November 1977.

[39] M. Piszczalski, B.A. Galler, "Predicting Musical Pitch from Component

Frequency Ratios", *J. Acoust. Soc. Am.* 66(3), September 1979.

[40] J.C. Brown, B. Zhang, "Musical Frequency Tracking using the Methods of Conventional and "Narrowed" Autocorrelation", *J. Acoust. Soc. Am.* 89(5), May 1991.

[41] J.C. Brown, "A High Resolution Fundamental Frequency Determination Based on Phase Changes of the Fourier Transform", *J. Acoust. Soc. Am.,* 94(2), August 1993.

[42] J.C. Brown, "Musical Fundamental frequency Tracking Using a Pattern Recognition Method ", *J. Acoust. Soc. Am.,* 92(3), September 1992

[43] R.C. Maher, J.W. Beauchamp, "Fundamental Frequency Estimation of Music Signals using a Two-Way Mismatch Procedure", *J. Acoust. Soc. Am.,* April 1994.

[44] B. Doval, X. Rodet, "Fundamental Frequency Estimation and Tracking using Maximum Likelihood Harmonic Matching and HMMs", *Proc. ICASSP,* pp. I-221-224, 1993.

[45] E. Dorken, S.H. Nawab, "Conditioned Constant-Q Spectra for Improved Frequency Tracking in the Presence of Interfering Signals", *J. Acoust. Soc. Am.* 95(4), April 1994.

[46] W.B. Kuhn, "A Real–Time Pitch Recognition Algorithm for Music Applications", *Computer Music Journal,* vol. 14, pp. 60–71, Fall 1990.

[47] A. Driesse, "Tempo Tracking in Real Time", M.S. Thesis, Dept. Comp. Inf. Sci., Queen's University, Kingston, Ontario, Canada, June, 1992.

[48] P.E. Allen, R.B. Dannenberg, "Tracking Musical Beats in Real Time", *Proc. Int. Comp. Music Conf.,* pp. 140–143, 1990.

[49] R.B. Dannenberg, H. Mukiano, "Real-Time Computer Accompaniment of Keyboard Performances", *Proc. Int. Comp. Music Conf.,* pp. 279–289, 1985.

[50] J.J. Bloch, R.B. Dannenberg, "New Techniques for Enhanced Quality of Computer Accompaniment", *Proc. Int. Comp. Music Conf.,* pp. 243–249, 1988.

[51] M. Dolson, "Recent Advances in Musique Concrete at CARL", *Proc. Int. Comp. Music Conf.,* pp. 85–60, 1985.

[52] F.J. Harris, "High-Resolution Spectral Analysis with Arbitrary Spectral Centers and Arbitrary Spectral Resolutions", *Comp. Elect. Eng.,* 3, pp. 171-191, 1976.

[53] G.W. Schwede, "An Algorithm and Architecture for Constant-Q Spectrum Analysis", *Proc. ICASSP* 29.2, pp. 1384-1387, 1983.

[54] K. Kashima, "The Bounded-Q Frequency Transform", *Department of Music Technical Report STAN-M-28,* 1985.

[55] J.C. Brown, "Calculation of a Constant-Q Spectral Transform", *J. Acoust. Soc. Am.,* 89(1), January 1991.

[56] J.C. Brown, "An Efficient Algorithm for the Calculation of a Constant-Q Transform", *J. Acoust. Soc. Am.,* 92(5), November 1992.

[57] R. Kronland-Martinet, "The Wavelet Transform for Analysis, Synthesis, and Processing of Speech and Music Sounds", *Comp. Music. J.,* 12, 11-20, 1988.

[58] E.R.S. Pearson, "The Multiresolution Fourier Transform and its Application to Polyphonic Audio Analysis", Ph.D. dissertation, The University of Warwick, UK, 1991.

[59] R. Wilson, A.D. Calway, E.R.S. Pearson, "Generalized Wavelet Transform for Fourier Analysis: The Multiresolution Fourier Transform and Its Application to Image and Audio Signal Analysis", *IEEE Trans. Inf Theory,* vol. 38, pp. 674–690, March 1992.

[60] C. Chafe, D. Jaffe, K. Kashima, B. Mont-Reyond, J. Smith, "Techniques for Note Identification in Polyphonic Music", *Proc. ICMC,* 1985.

[61] I. Daubechies, *Ten Lectures on Wavelets,* SIAM, Philadelphia, 1992.

[62] I. Daubechies, "Orthonormal Bases of Compactly Supported Wavelets", *Comm. Pure and Applied Math.,* vol. XLI, pp. 909–996, 1988.

[63] I. Daubechies, "The Wavelet Transform, Time–Frequency Localization and Signal Analysis", *IEEE Trans. Inf Theory",* vol. 36, pp. 961– 1005, September 1990.

[64] M. Vetterly, J. Kovacevic, *Wavelets and Subband Coding,* Prentice-Hall, Englewood Cliffs, NJ, 1995.

[65] M. Veterly, "Filter Banks Allowing Perfect Reconstruction", *Signal Proc.,* vol. 10(3), pp. 219–244, April 1986.

[66] M. Vetterly, C. Harley, "Wavelets and Filter Banks: Theory and Design", *IEEE Trans. Signal Proc.,* vol. 40, pp. 2207-2232, September 1992.

[67] P.P.Vaidyanathan, "Multirate Digital Filters, Filter Banks, Polyphase Networks,

and Applications: A Tutorial", *Proceedings of the IEEE,* vol. 78, pp. 56–93, January 1990.

[68] Nguyen, P.P. Vaidyanathan, "Two–Channel Perfect–Reconstruction FIR QMF Structures Which Yield Linear–Phase Analysis and Synthesis Filters", *IEEE Trans. Acoust., Speech, Signal Processing,* vol. 37, pp. 676–690, May 1989.

[69] L. Cohen, *Time-Frequency Analysis,* Prentice-Hall, Englewood Cliffs, NJ, 1995.

[70] P. Goupillaud, A. Grossman, J. Morlet, "Cycle–Octave and Related Transforms in Seismic Signal Analysis", *Geoexploration,* Elsevier Science Pub., vol 23, pp. 85–102, 1984/85.

[71] S.G. Mallat, "A Theory for Multiresolution Signal Decomposition: The Wavelet Representation", *IEEE Trans. Pattern Anal. Machine Intell.",* vol. 38. pp. 674–693, July 1989.

[72] A. Haar, "Zur Theorie der orthogonalen Functionensysteme", *Math. Annal.,* 69:331-337, 1910.

[73] Y. Meyer, *"Wavelets, Algorithms, and Applications",* SIAM, Philadelphia, 1993.

[74] C.K. Chui, editor, *Wavelets: A Tutorial in Theory and Applications,* Academic Press, New York, 1992.

[75] A. Croisier, D. Esteban, C. Galand, "Perfect Channel Splitting by use of Interpolation/Decimation/Tree Decomposition Techniques", *Int. Conf. on Inf. Sciences and Systems,* pp. 443–446, Patras, Greece, August 1976.

[76] M.R. Portnoff, "Time-Frequency Representation of Digital Signals and Systems Based on Short-Time Fourier Analysis", *IEEE Trans. ASSP,* vol. ASSP-28, pp. 55-69, February 1980.

[77] G. Evangelista, "Discrete–Time Wavelet Transform", Ph.D. dissertation, University of California, Irvine, 1990.

[78] C. Herley, "Wavelets and Filter Banks", Ph.D. dissertation, Columbia University, New York, 1993.

[79] M.J.T.Smith, T.P. Barnwell, III, "A Procedure for Designing Exact Reconstruction filter Banks for Tree Structured Subband Coders", *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing,* San Diego, March 1984.

[80] M.J.T.Smith, T.P. Barnwell, III, "A New Filter Bank Theory for Time–Frequency Representation", *IEEE Trans. Acoust., Speech, Signal Processing,* vol.

ASSP–35, pp. 314–326, March 1987.

[81] G.W. Medlin, J.W. Adams, C.T. Leondes, "Lagrange Multiplier Approach to the Design of FIR Filters for Multirate Applications", *IEEE Trans. Circuit Syst.,* vol. 35, pp. 1210-1219, October 1988.

[82] G.W. Medlin, J.W. Adams, "A New Technique for Maximally Linear Differentiators", *Proc. Int. Conf. ASSP,* pp. 825-828, 1989.

[83] B.Hong, A.N. Wilson, "Lagrange Multiplier Approaches to the Design of Two–Cannel Perfect–Reconstruction Linear–Phase FIR Filter Banks", *IEEE Trans. Acoust., Speech, Signal Processing,* vol. 40, pp. 364–374, February 1992.

[84] P.P Vaidyanathan, *Multirate Systems and Filter Banks,* Prentice-Hall, Englewood Cliffs, New Jersey, 1993.

[85] P.P. Vaidyanathan, P.Q. Hoang, "Lattice Structures for Optimal Design and Robust Implementation of Two-Channel Perfect Reconstruction Filter Banks", *IEEE Trans. ASSP,* vol. 36, pp. 81-94, January 1988.

[86] R.E. Crochiere, L.R. Rabiner, *Multirate Digital Signal Processing,* Prentice-Hal, Englewood Cliffs, New Jersey, 1983.

[87] J.D. Johnston, "A Filter Family Designed for use in Quadrature Mirror Filter Banks", *Proc. Int. Conf. Acoust., Speech, Signal Proc.,* pp. 291–294, April 1980.

[88] V.K. Jain, R.E. Crochiere, "Quadrature Mirror Filter Design in the Time Domain", *IEEE Trans. Acoust., Speech, Signal Processing,* pp.353–361, April 1984.

[89] G. Strang, T. Nguyen, *Wavelets and Filter Banks,* Wellesley-Cambridge Press, Wellesley, MA, 1996.

[90] M.V. Wickerhauser, "Acoustic Signal Compression with Wavelet Packets", In C.K. Chui, editor, *Wavelets: A Tutorial in Theory and Applications,* Academic Press, New York, 1992.

[91] J. Rothstein, *MIDI. A Comprehensive Introduction,* A-R Editions, Inc., Madison, WI, 1992.

[92] O. Rioul, P. Duhamel, "Fast Algorithms for Discrete and Continuous Wavelet Transforms",*IEEE Trans. Inf Theory,* vol. 38, pp. 569-586, March 1992.

[93] H.J. Nussbaumer, *Fast Fourier Transform and Convolution Algorithms,* Springer-Verlag, Berlin, 1982.

[94] Z.J. Mou, P. Duhamel, "Short-Length FIR Filters and their Use in Fast Nonrecursive Filtering", *IEEE Trans. Signal Proc.,* vol. 39, pp. 1322-1332, June 1991.

[95] M. Vetterly, "Running FIR and IIR Filtering Using Multirate Filter Banks," *IEE Trans. ASSP,* vol. ASSP-36, pp. 730-738, May 1988.

[96] R. Chassing, R. Thayer, "Multirate Filtering Using the TMS320C30 Floating Point Digital Signal Processor," *Proc. ASEE Annual Conference,* pp. 34-37, 1991.

[97] S. Kercel, "A Near-Real-Time Instrument for Wideband Magnetic Field Monitoring with Simultaneous Time and Frequency Localization by Multiresolution Filter Bank," Ph.D. Dissertation, The University of Tennessee, August 1995.

[98] T. Lin, "Architectures and Circuits for High-Performance Multirate Digital Filters with Applications to Tunable Modulator/Demodulator/Bandpass-Filter Systems," Ph.D. Dissertation, University of California at Los Angeles, 1993.

[99] M. Vishwanath, R.M. Owens, M.J. Irwin, "VLSI Architectures for the Discrete Wavelet Transform," *IEEE Trans. on Circuit and Systems II: Analog and Digital Signal Processing,* vol. 42, pp. 305-316, May 1995.

[100] K.K. Parhi, T. Nishitani, "VLSI Architectures for Discrete Wavelet Transforms," *IEEE Trans. on VLSI Systems,* vol. 1, No. 2, June 1993.

[101] Passport Designs, Inc., "TurboTrax", MIDI Sequencing Software.

APPENDICES

APPENDIX A

A NEW METHOD FOR THE DESIGN OF HIGH-QUALITY TWO-CHANNEL LINEAR PHASE QMF BANK

In a standard two-channel filter bank with the analysis filters $H_0(z)$ and $H_1(z)$, and the synthesis filters $G_0(z)$ and $G_1(z)$, the output $\hat{X}(z)$ is given by

$$\begin{aligned}\hat{X}(z) = &\left[H_0(z)G_0(z) + H_1(z)G_1(z)\right]X(z)/2 \\ &+\left[H_0(-z)G_0(z) + H_1(-z)G_1(z)\right]X(-z)/2.\end{aligned} \tag{A.1}$$

To cancel the aliasing part in the output one may choose

$$\begin{aligned}G_0(z) &= 2H_1(-z) \\ G_1(z) &= -2H_0(-z).\end{aligned} \tag{A.2}$$

Then (A.1) becomes

$$\hat{X}(z) = \left[H_0(z)H_1(-z) - H_0(-z)H_1(z)\right]X(z). \tag{A.3}$$

For perfect reconstruction, the following should be satisfied

$$\left|H_0(z)H_1(-z) - H_0(-z)H_1(z)\right| = 1, \tag{A.4}$$

and if $H_0(z)$ and $H_1(z)$ satisfy the QMF condition, $H_1(z) = H_0(-z)$, then (A.4) becomes

$$\left|H_0^2(z) - H_0^2(-z)\right| = 1. \tag{A.5}$$

For even-order FIR filters with linear phase (A.5) changes to

$$\left|H_0(e^{j\omega})\right|^2 + \left|H_0(e^{j(\omega+\pi)})\right|^2 = 1. \tag{A.6}$$

Additionally, it is desired that $H_0(z)$ approximates the ideal low-pass condition

$$\left|H_0(e^{j\omega})\right| = \begin{cases}1, & 0 \le \omega < \pi/2 \\ 0, & \pi/2 \le \omega \le \pi\end{cases} \tag{A.7}$$

Conditions (A.6) and (A.7) cannot be met simultaneously. The algorithms for the near-perfect reconstruction filter design minimize the power-complementary error

$$\varepsilon = \max\left\{\left|10\log\left(\left|H_0(e^{j\omega})\right|^2 + \left|H_0(e^{j(\omega+\pi)})\right|^2\right)\right|\right\} \tag{A.8}$$

thus approximating both (A.6) and (A.7). On the other hand, the perfect reconstruction filter design approximates (A.7) only, satisfying (A.4) exactly. However, then the QMF condition, $H_1(z) = H_0(-z)$, must be given up.

In the new algorithm, (A.6) is approximated thus minimizing the power-complementary error (A.8), while (A.7) is approximated by choosing a good lowpass filter. The algorithm is described below.

STEP 1. Denote the iteration index by i and set it to 0. Using any filter design algorithm design a lowpass FIR filter, $h_{FIR}^{(i)}(n)$, with desired transition band and sidelobe attenuation. Order N of the filter must be even.

STEP 2 Adjust the cut-off frequency of the lowpass filter so that the power-complementary error defined in (A.8) is minimized. If both time and frequency are discrete, then the power-complementary error is given by

$$\varepsilon_d(i) = \max\left\{\left|10\log\left(\left|H_{FIR}^{(i)}(k)\right|^2 + \left|H_{FIR}^{(i)}(k+M/2)\right|^2\right)\right|\right\},\ k = 0,...,M-1 \qquad \text{(A.9)}$$

where

$$H_{FIR}^{(i)} = \sum_{n=0}^{M-1} h_{FIR}^{(i)}(n)e^{-j2\pi nk/M} \qquad \text{(A.10)}$$

represents an M-point FFT of the filter $h_{FIR}^{(i)}(n)$ in ith step. The FFT order M needs to satisfy $M >> N$.

STEP 3 Calculate a new frequency response with zero power-complementary error as

$$H_{IIR}^{(i)}(k) = \frac{H_{FIR}^{(i)}(k)}{\sqrt{\left|H_{FIR}^{(i)}(k)\right|^2 + \left|H_{FIR}^{(i)}(k+M/2)\right|^2}},\ k = 0,...,M-1. \qquad \text{(A.11)}$$

and then the corresponding impulse response as

$$h_{IIR}^{(i)}(n) = \frac{1}{M}\sum_{n=0}^{M-1} H_{IIR}^{(i)}(k)e^{j2\pi nk/M},\ n = 0,...,M-1. \qquad \text{(A.12)}$$

STEP 4 Multiply $h_{IIR}^{(i)}(n)$ by a rectangular widow so that only N of M points have nonzero values. The new impulse response is given by

$$h_{FIR}^{(i+1)}(n) = \begin{cases} h_{IIR}^{(i)}(n), & n < N \\ 0, & n \geq N \end{cases}, n = 0,...,M-1. \qquad \text{(A.13)}$$

STEP 5 Calculate the error (A.9) for $h_{FIR}^{(i+1)}(n)$, and either stop if the error is acceptable low, or increase the index i and go back to STEP 3.

Figure A.1 shows frequency responses of the new design filters and the corresponding power-complementary errors.

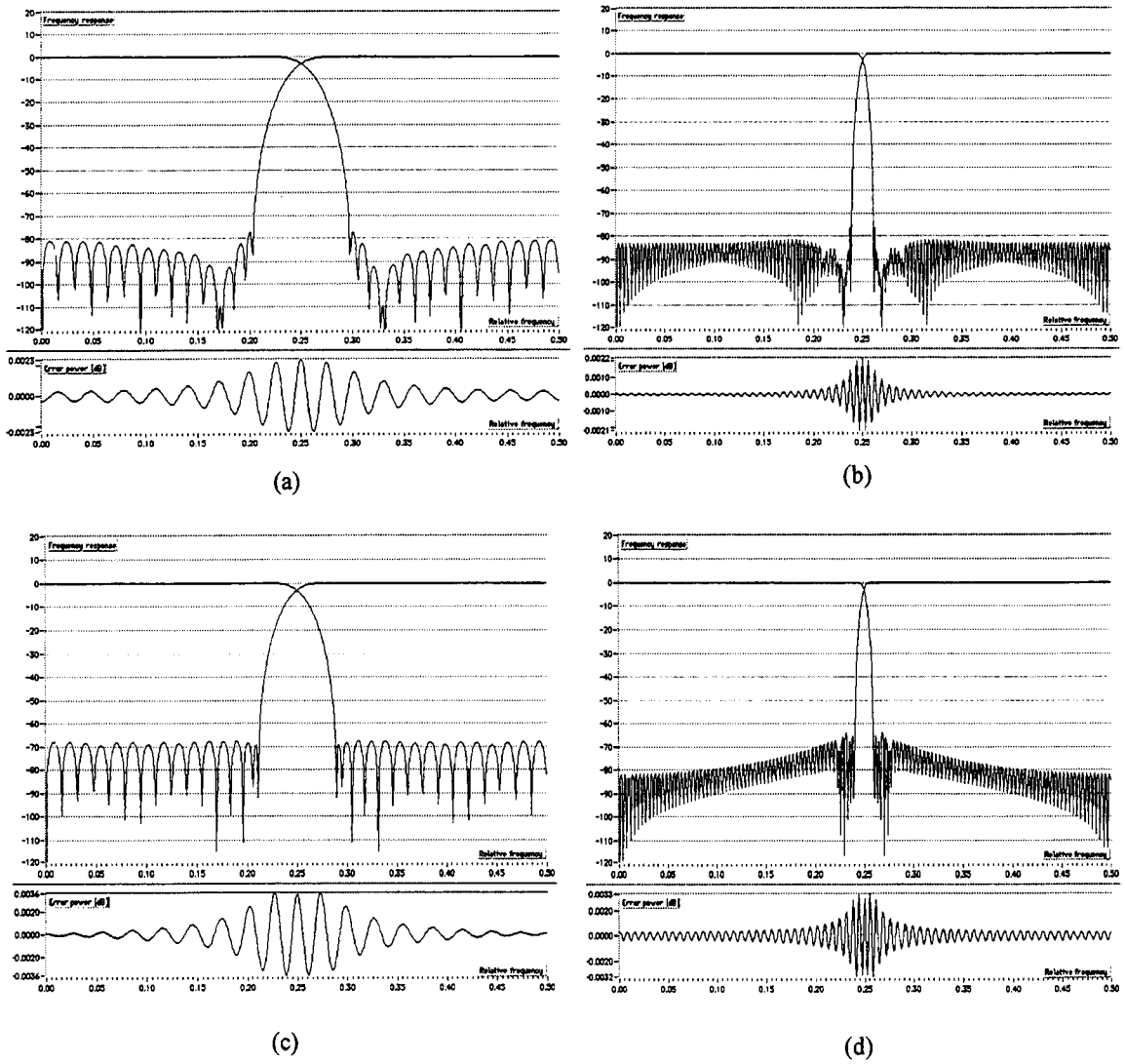
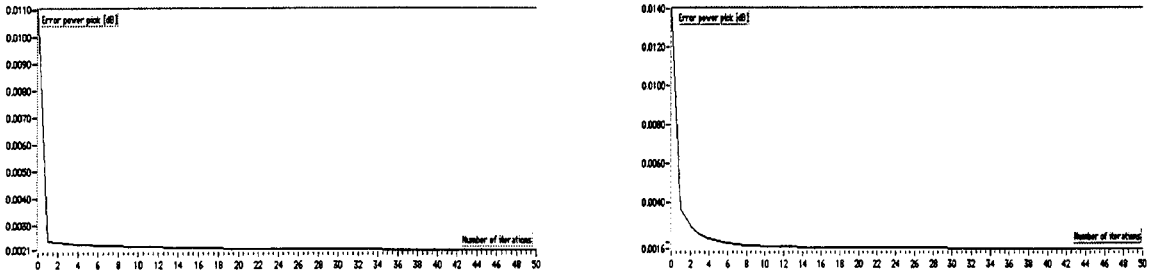


Figure A.1 New design filters and their power-complementary errors for M = 1024. The starting filter is designed using the windowed FIR design and the Kaiser-Bessel window for N = 64 (a) and N = 256 (b), and the Parks-McClellan algorithm for N = 64 (c) and N = 256 (d).

Figure A.2 shows that convergence in the algorithm is fast and occurs after about 10 iterations for a 64-tap filter and after about 20 iterations for a 256-tap filter.



(a) (b)

Figure A.2 Power-complementary error as a function of the number of iterations for 64-tap filters (a) and 256-tap filters (b).

APPENDIX B

LABVIEW PROGRAM FOR TREE-STRUCTURED FILTER BANK IMPLEMENTATION (ANALYSIS PART)

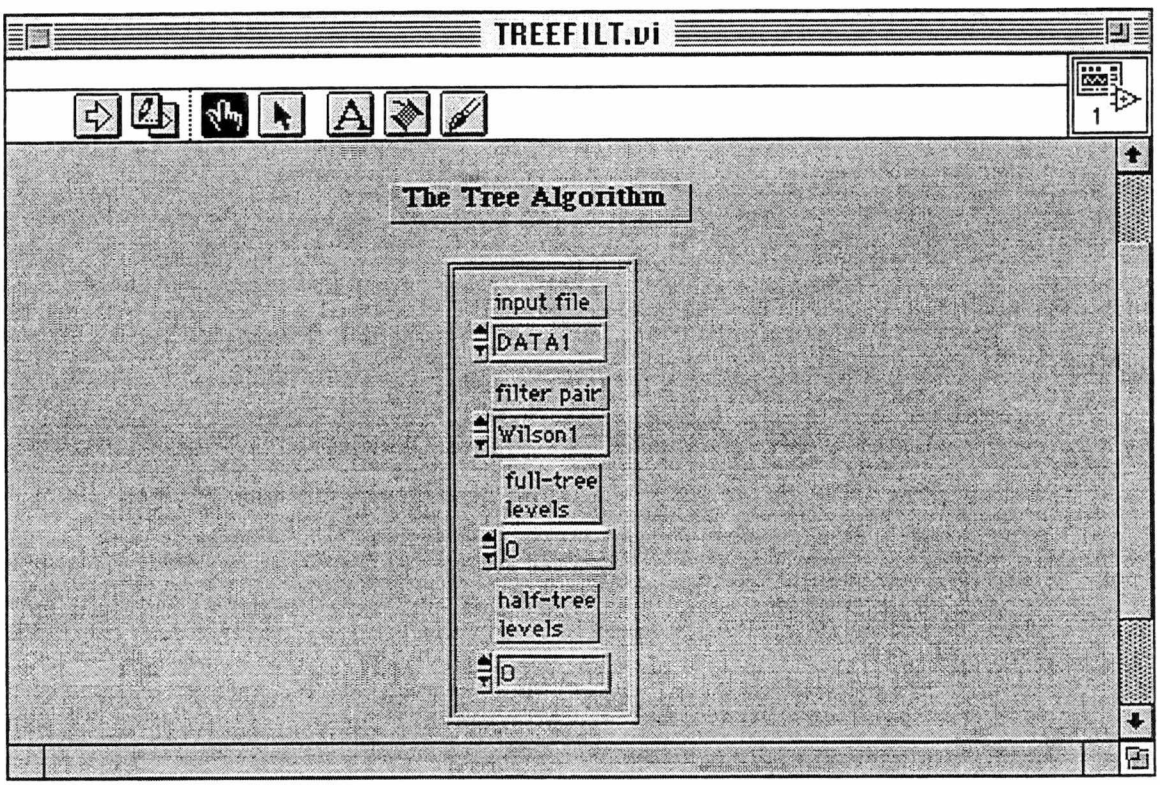


Figure B.1 Front panel of TREEFILT.vi.

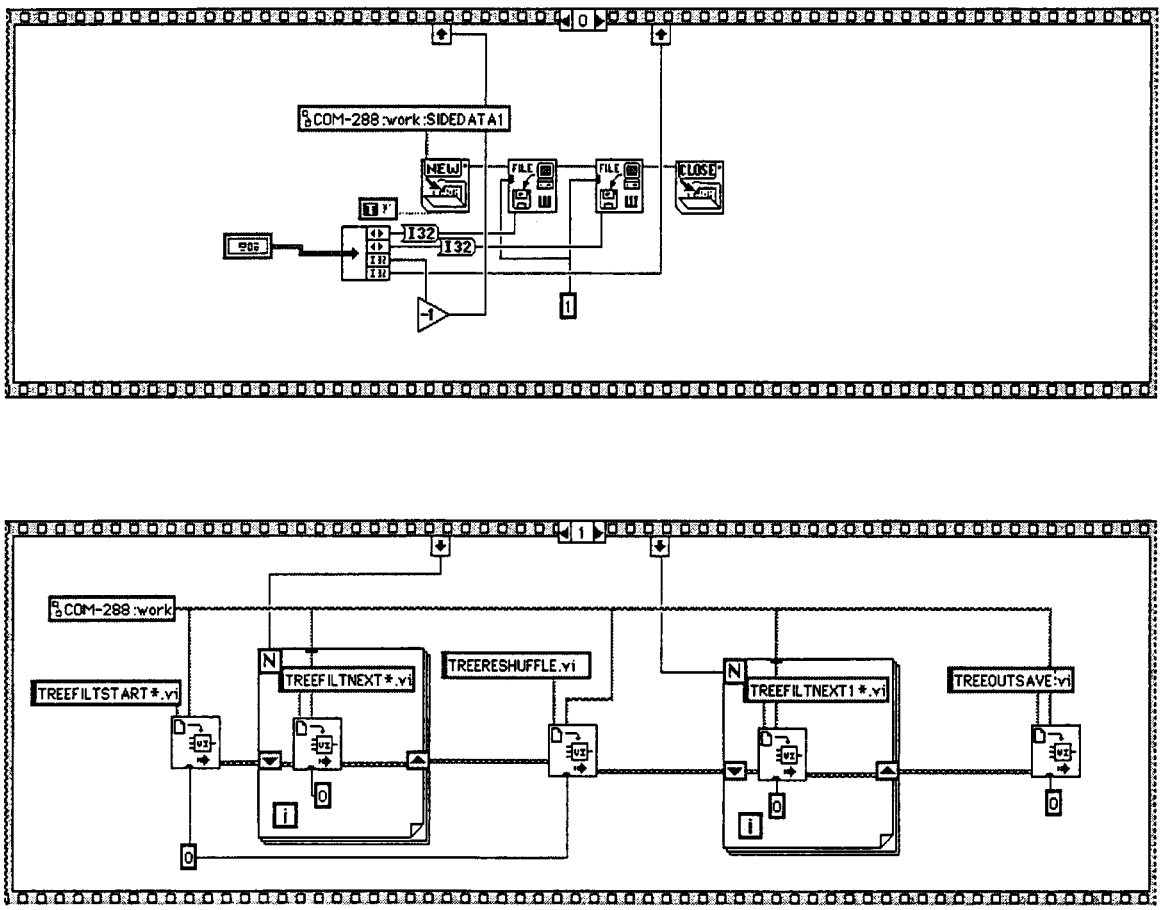


Figure B.2 Diagram of TREEFILT.vi.

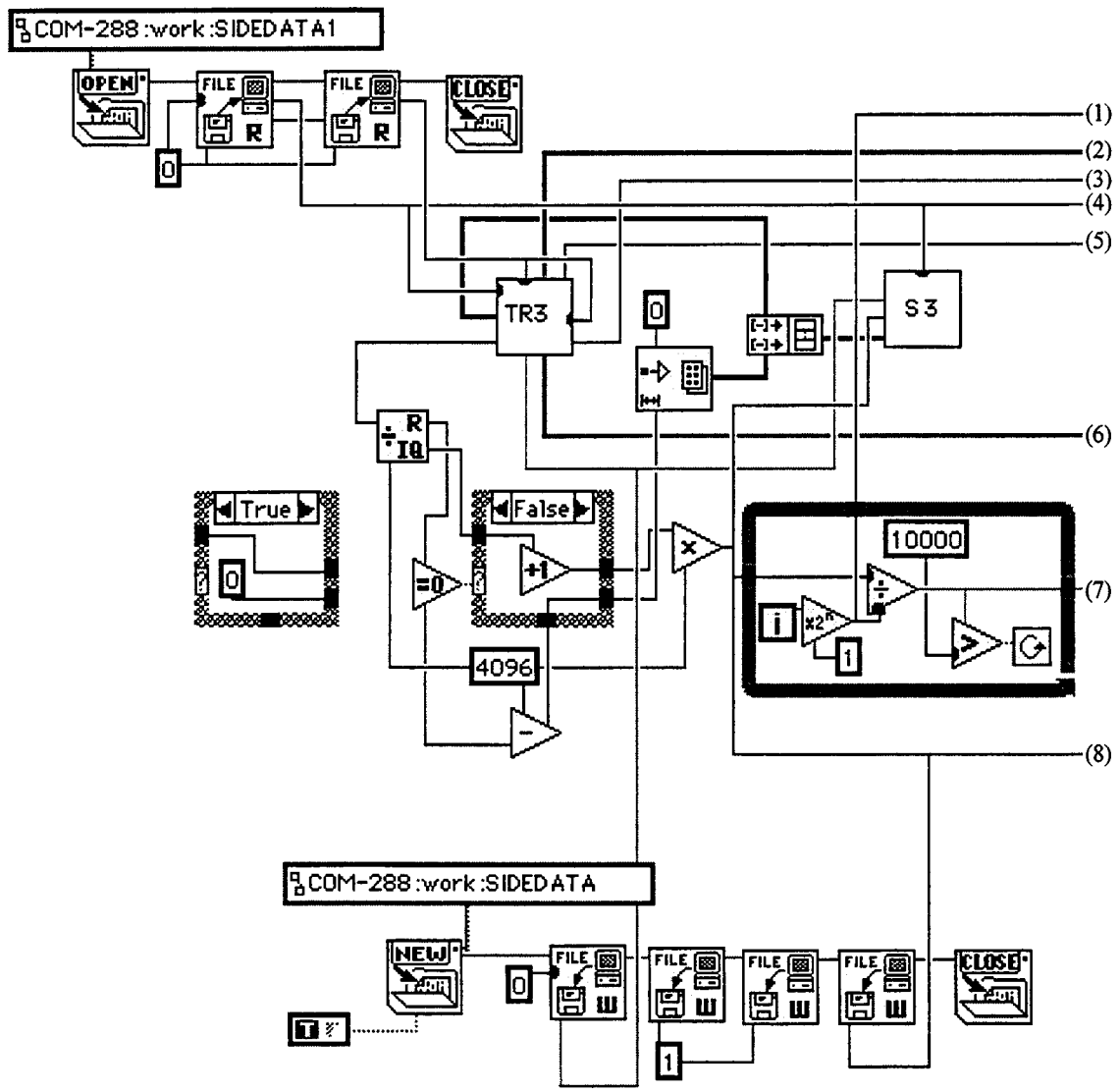


Figure B.3 Diagram of TREEFILTSTART*.vi.

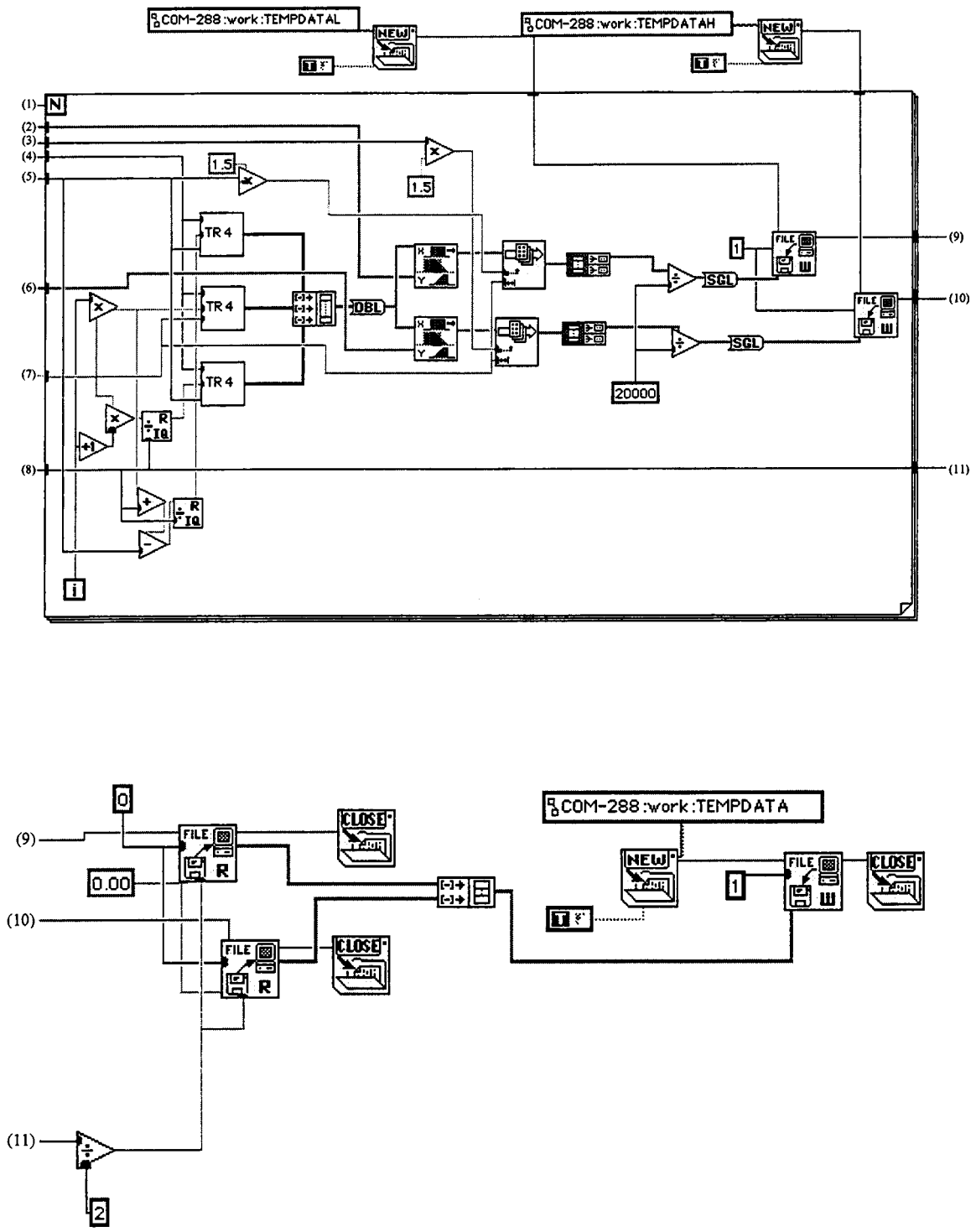


Figure B.3 (continued).

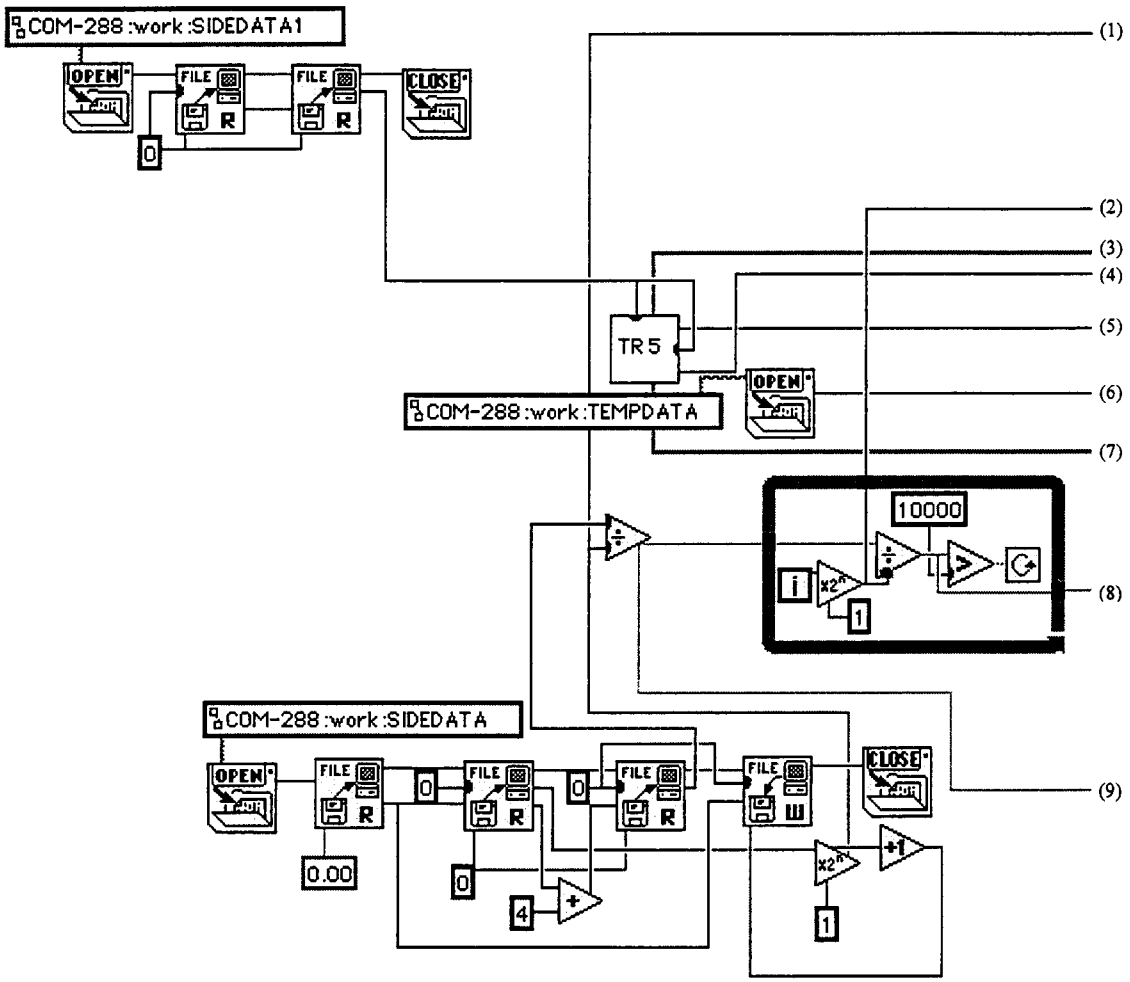


Figure B.4 Diagram of TREEFILTNEXT*.vi.

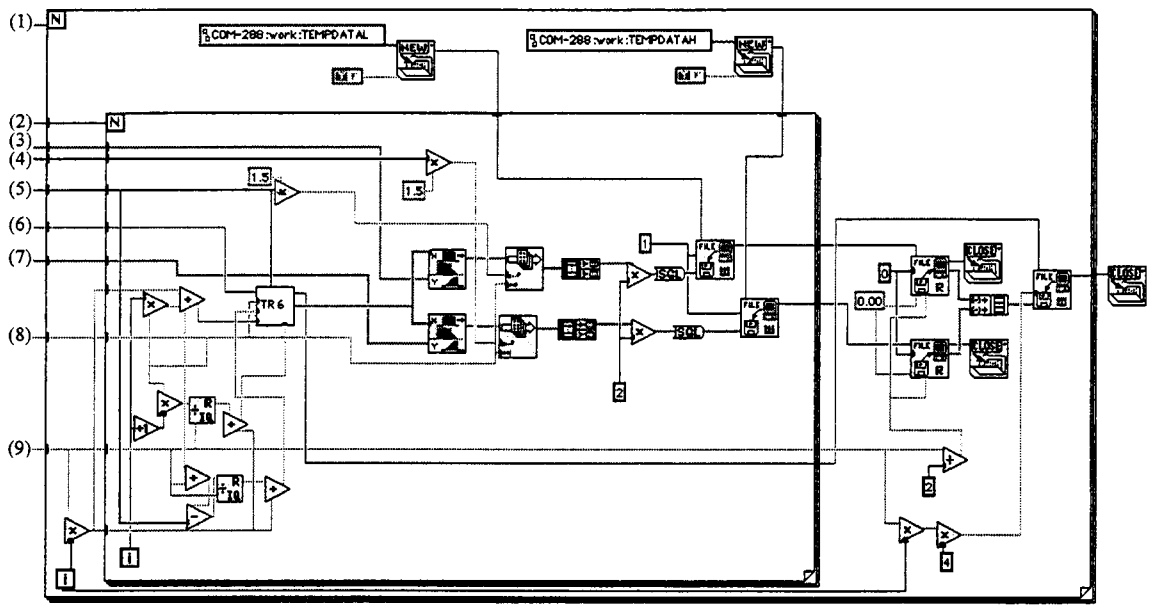


Figure B.4 (continued).

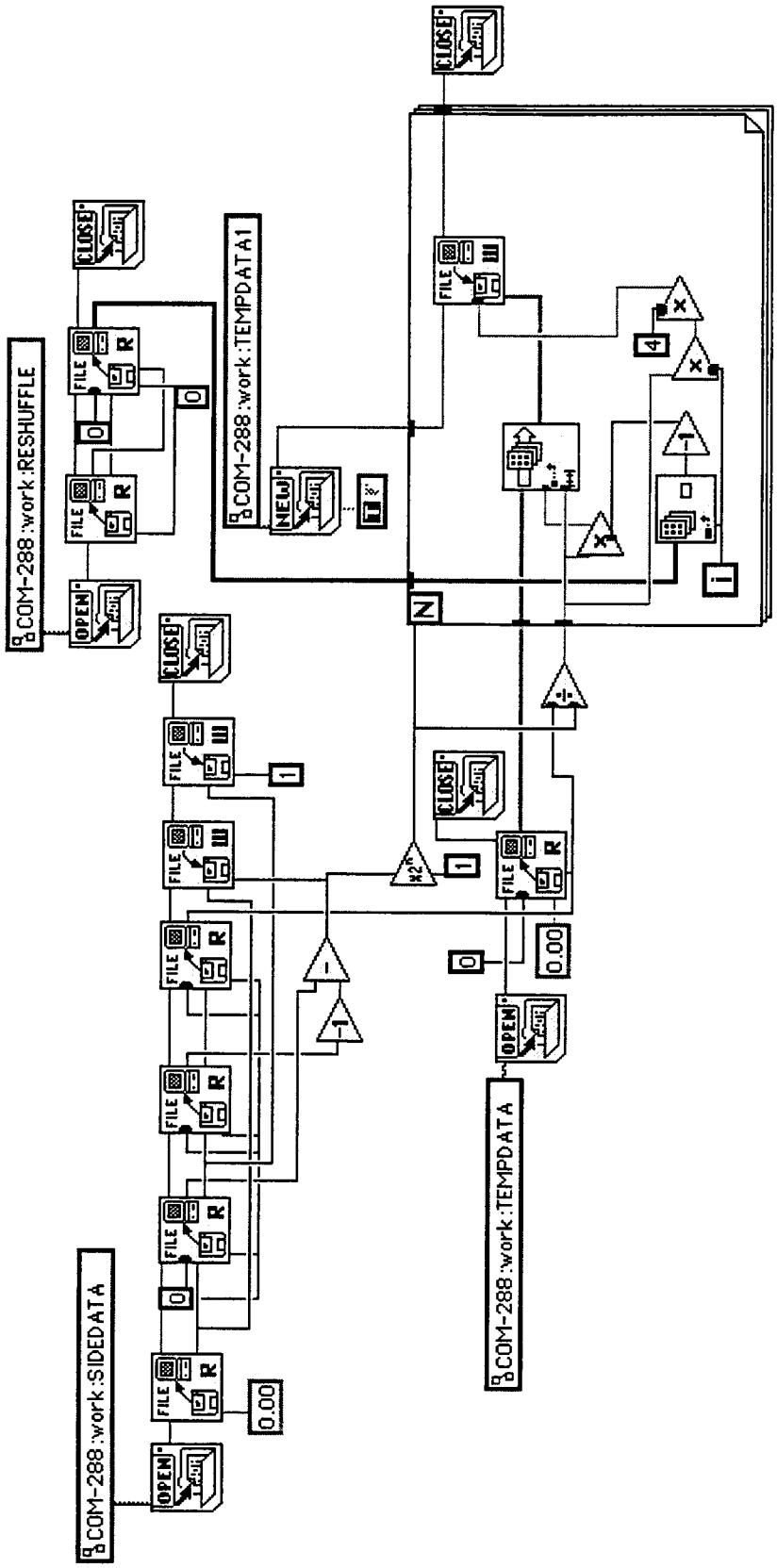


Figure B.5 Diagram of TREERESHUFFLE.vi.

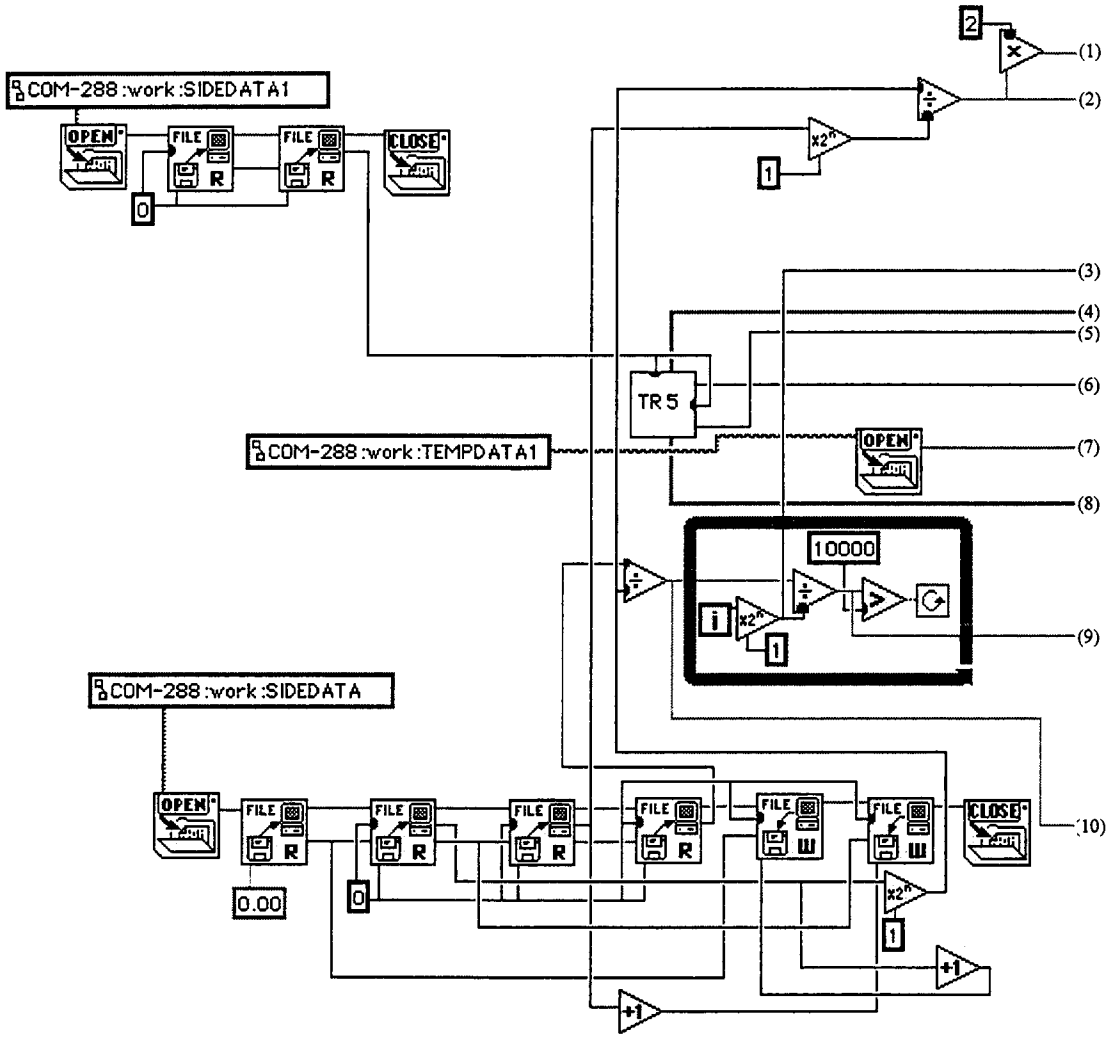


Figure B.6 Diagram of TREEFILTNEXT1*.vi.

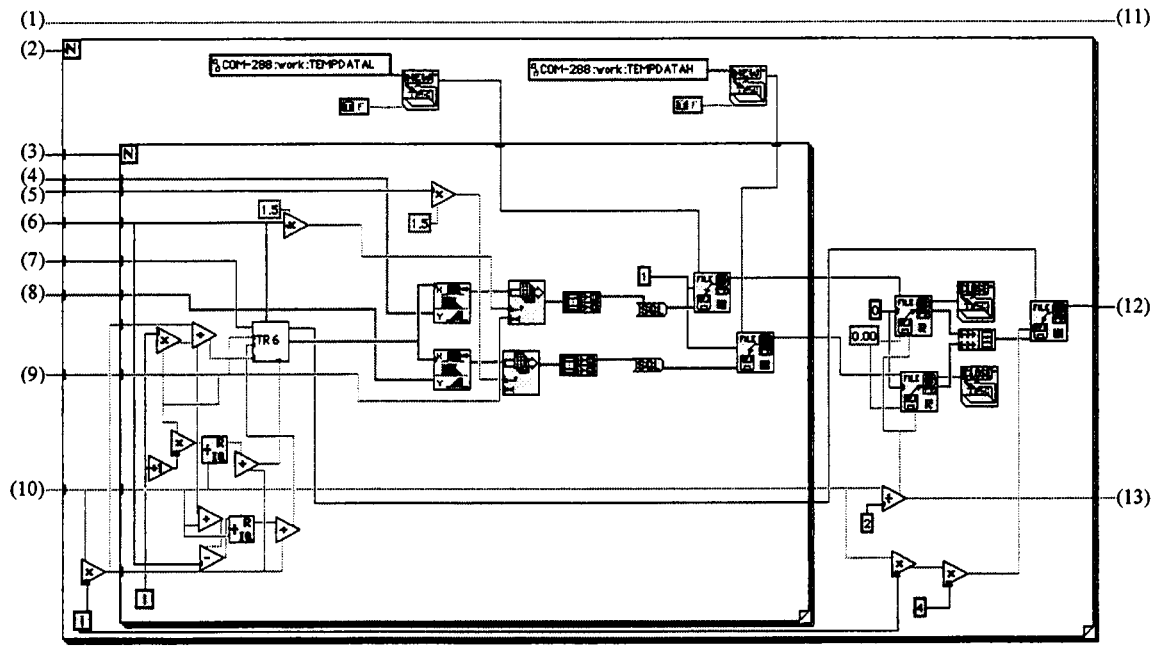


Figure B.6 (continued)

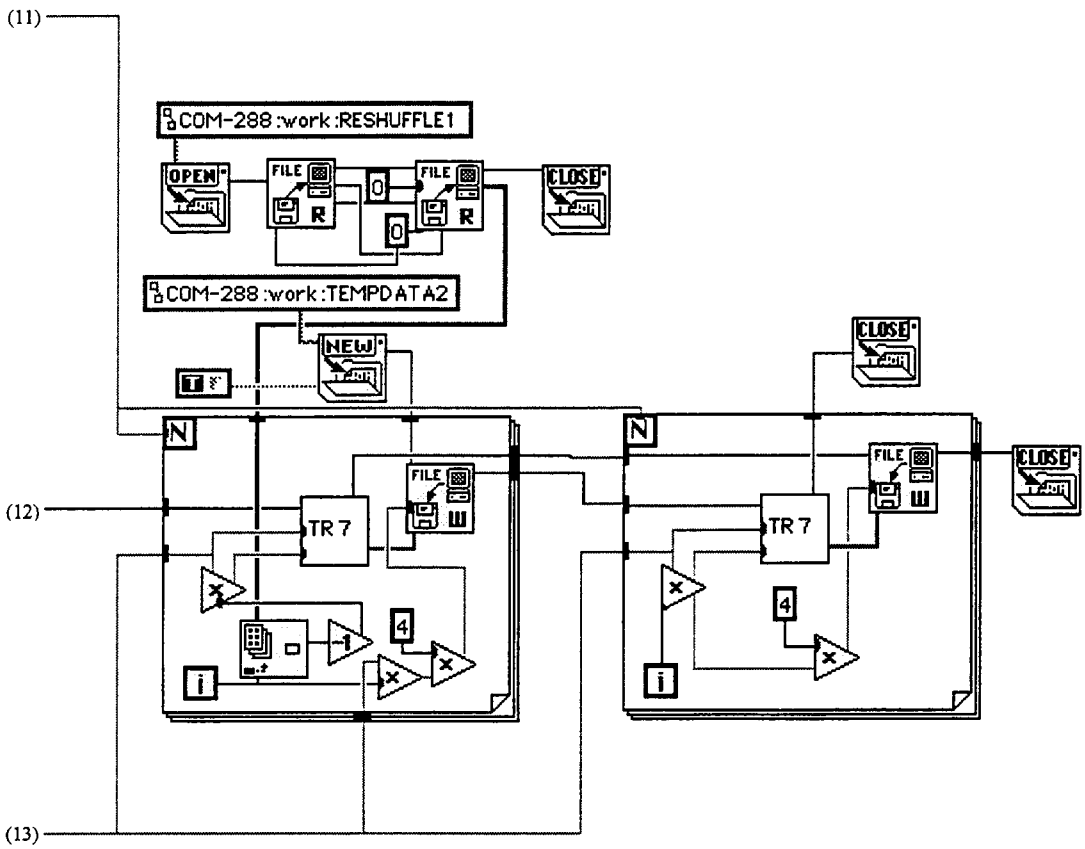


Figure B.6 (continued).

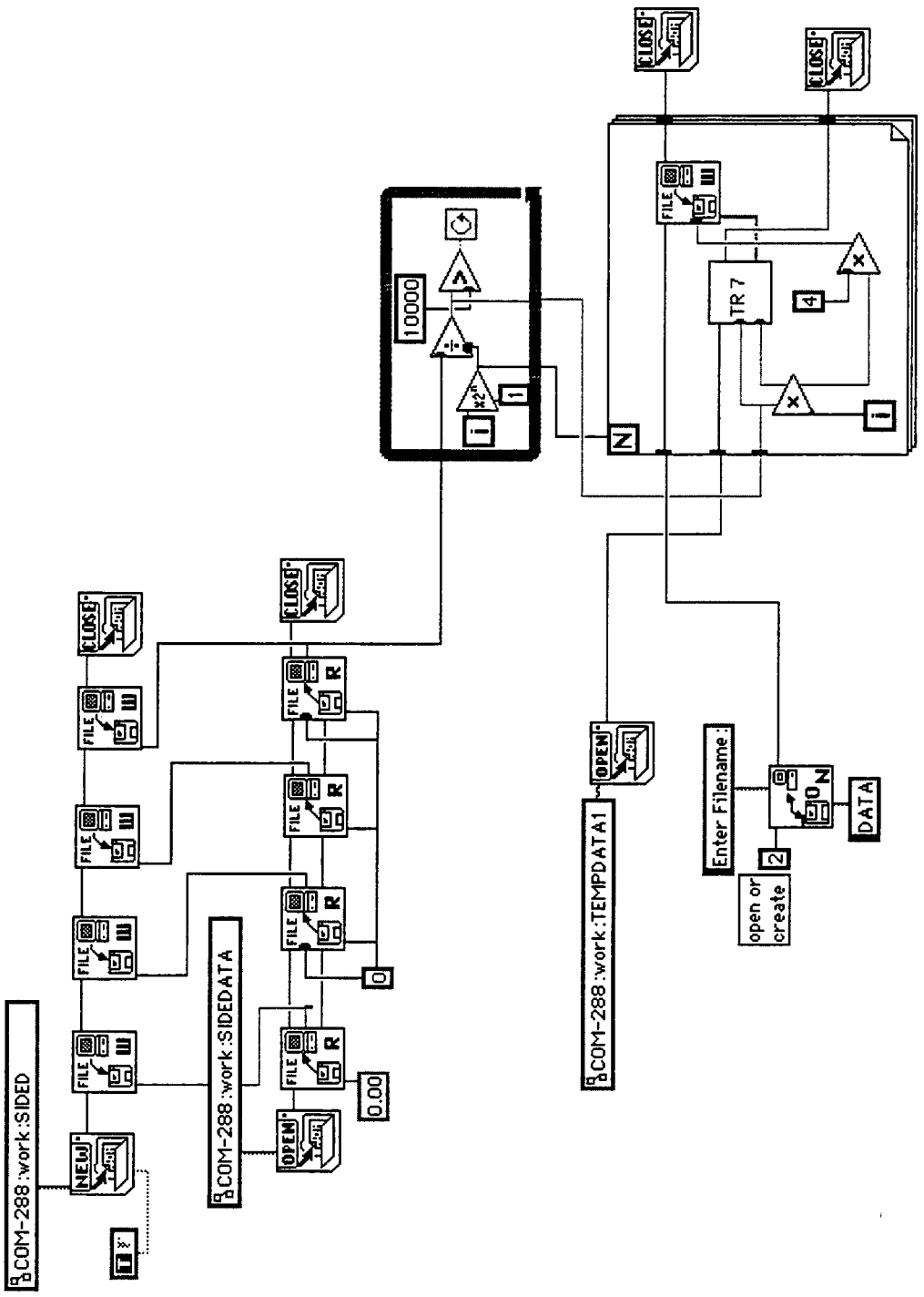


Figure B.7 Diagram of TREEOUTSAVE.vi.

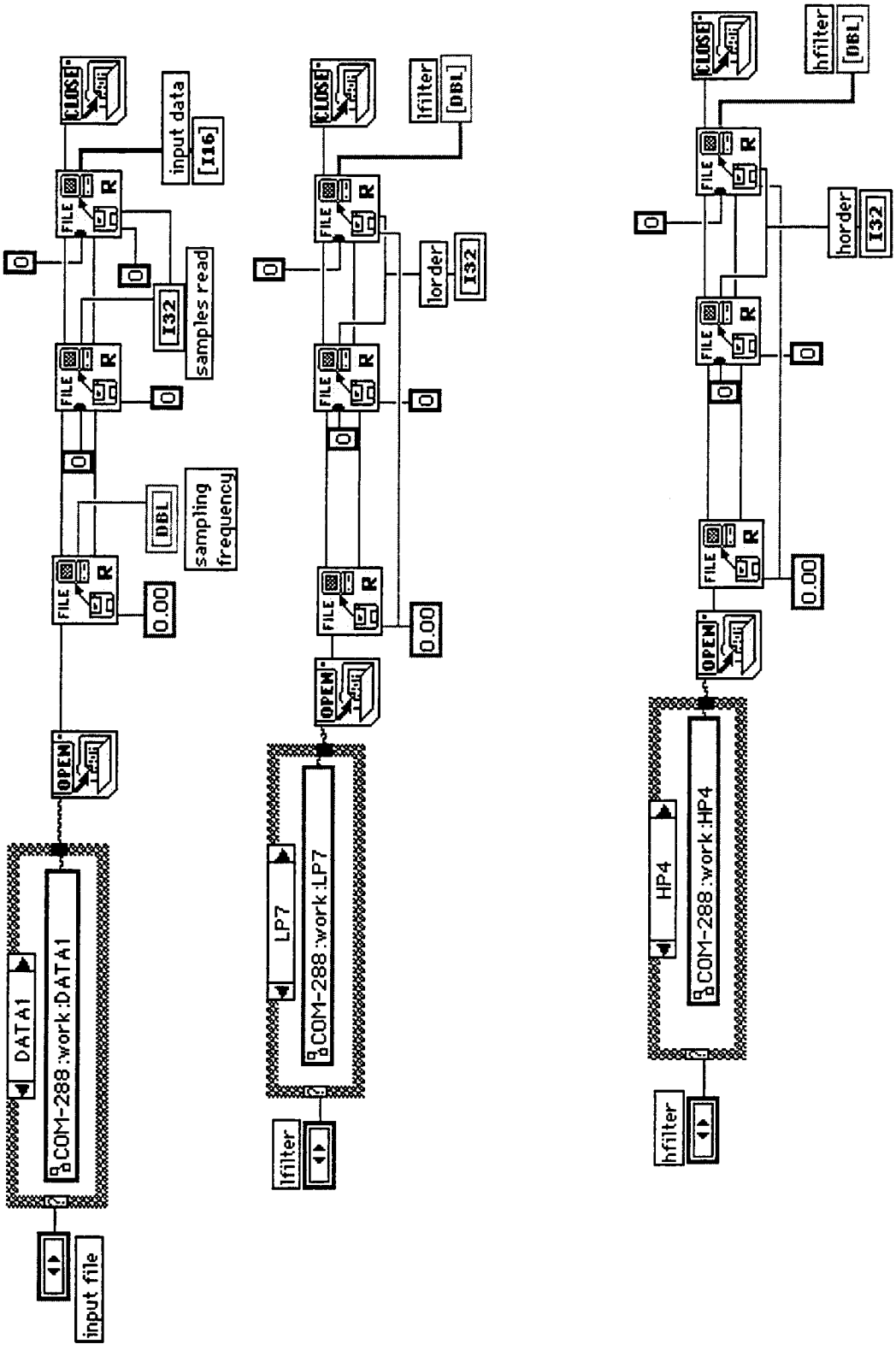


Figure B.8 Diagram of TR3.vi

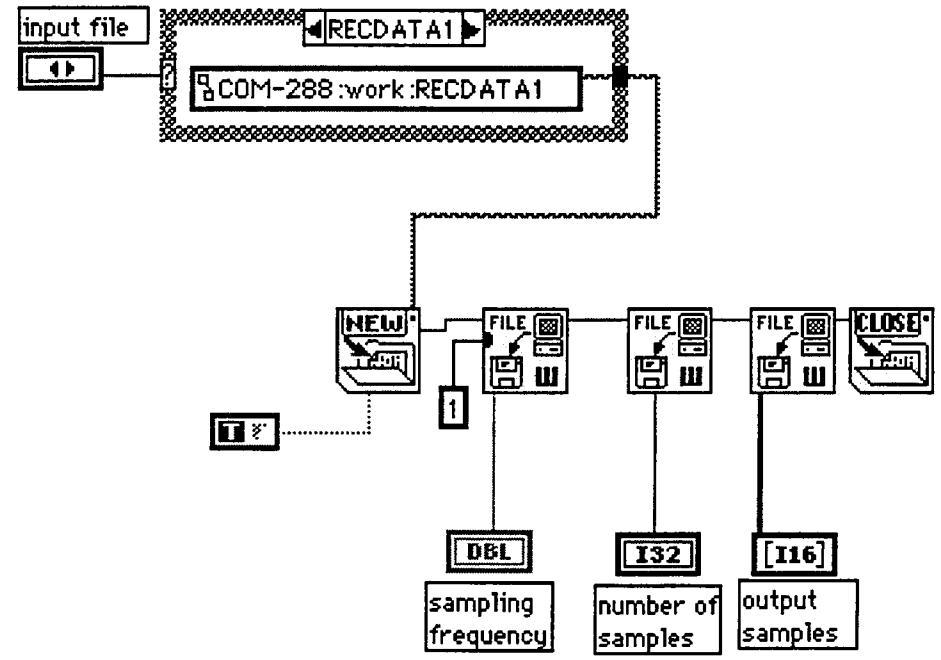


Figure B.9 Diagram of S3.vi.

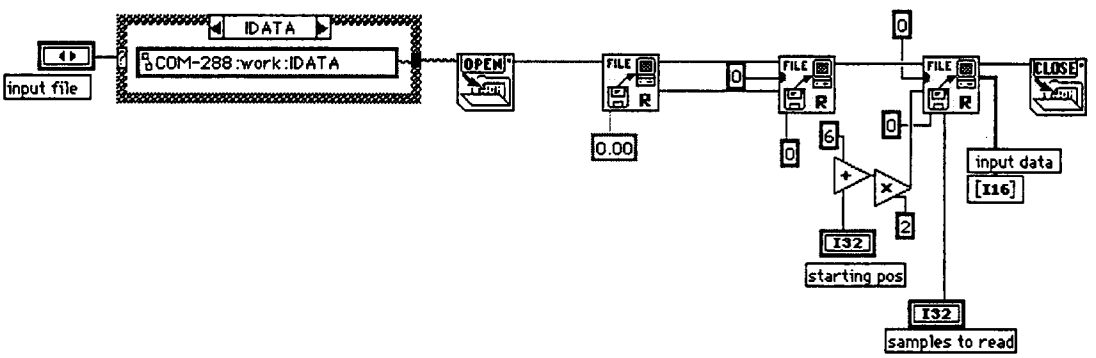


Figure B.10 Diagram of TR4.vi.

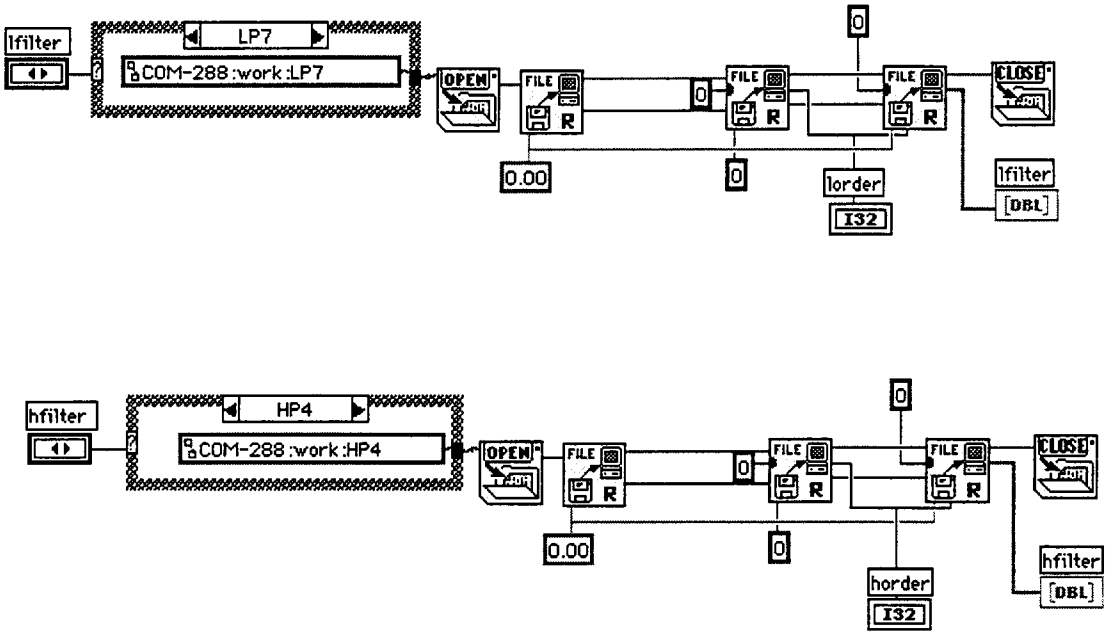


Figure B.11 Diagram of TR5.vi

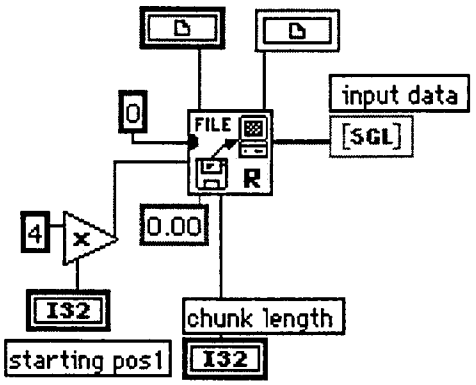


Figure B.12 Diagram of TR6.vi.

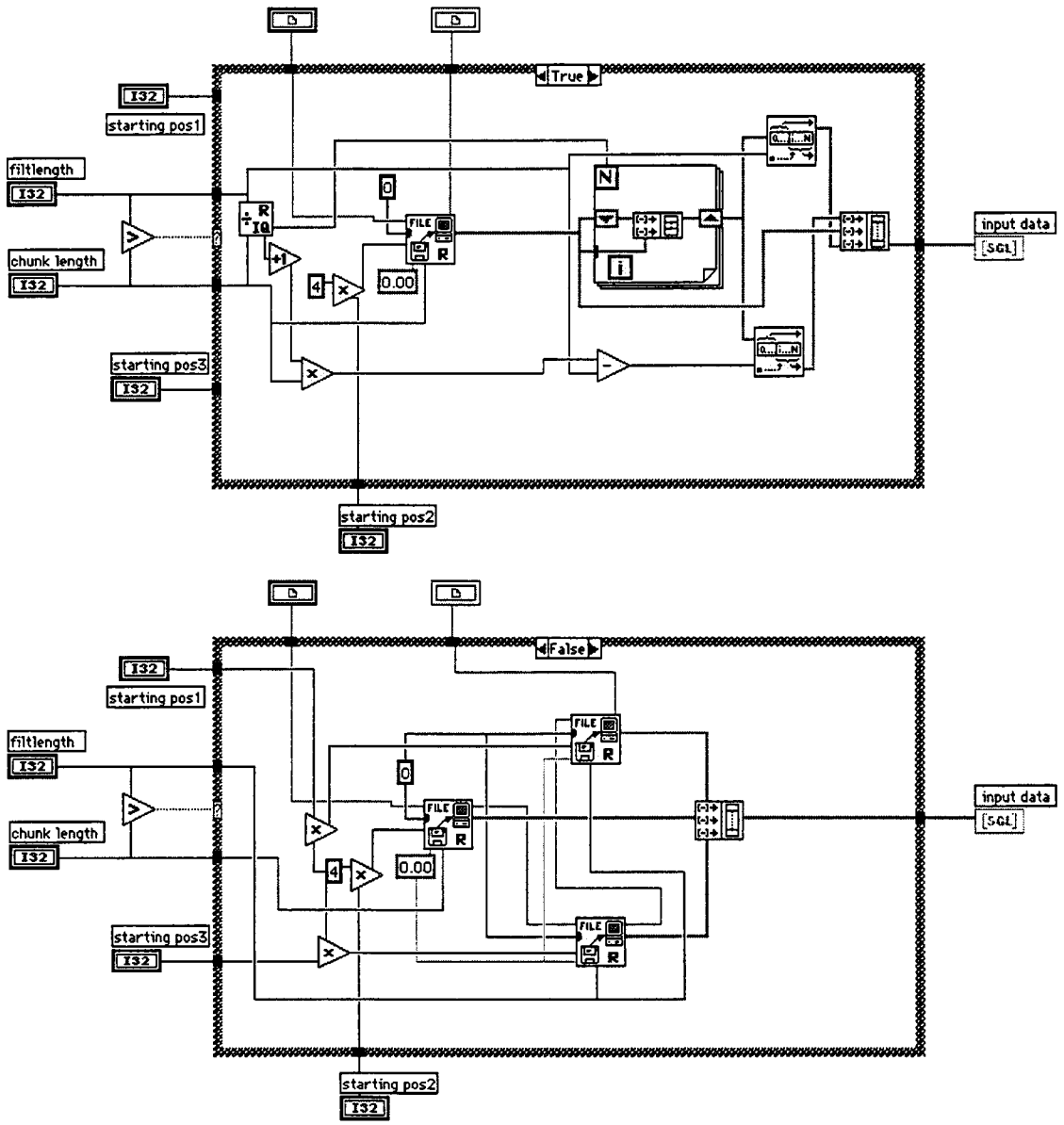


Figure B.13 Diagram of TR7.vi.

APPENDIX C

LABVIEW PROGRAM FOR ATTACK DETECTION

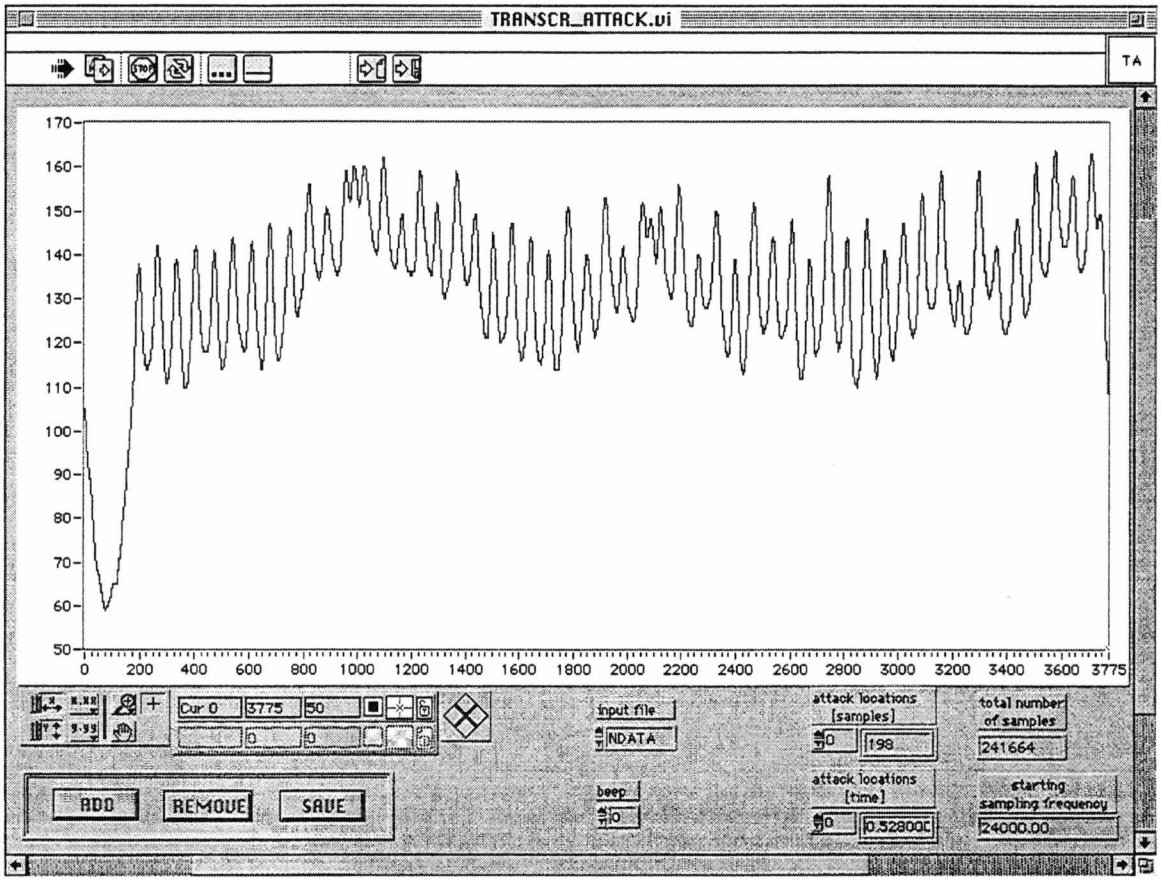


Figure C.1 Front panel of TRANSCR_ATTACK.vi.

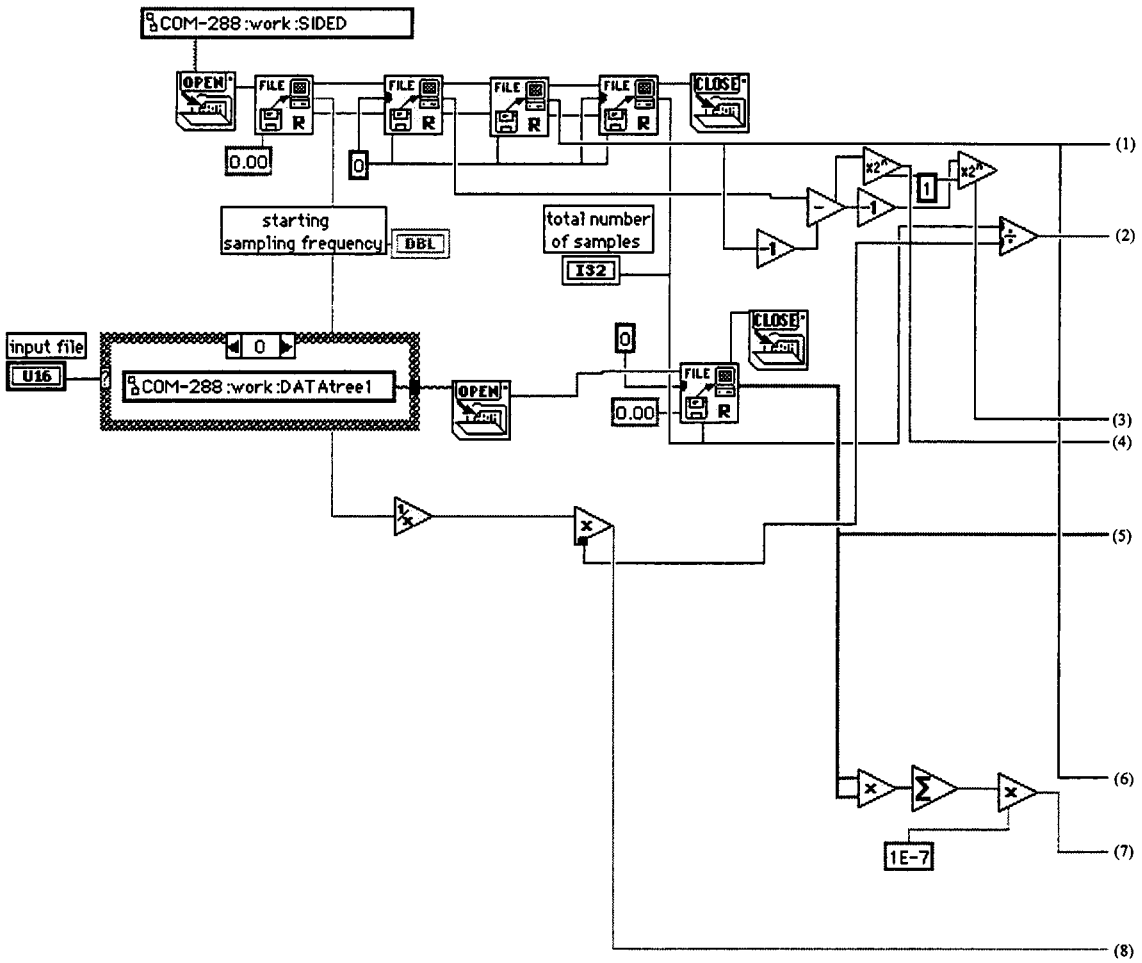


Figure C.2 Diagram of TRANSCR_ATTACK.vi.

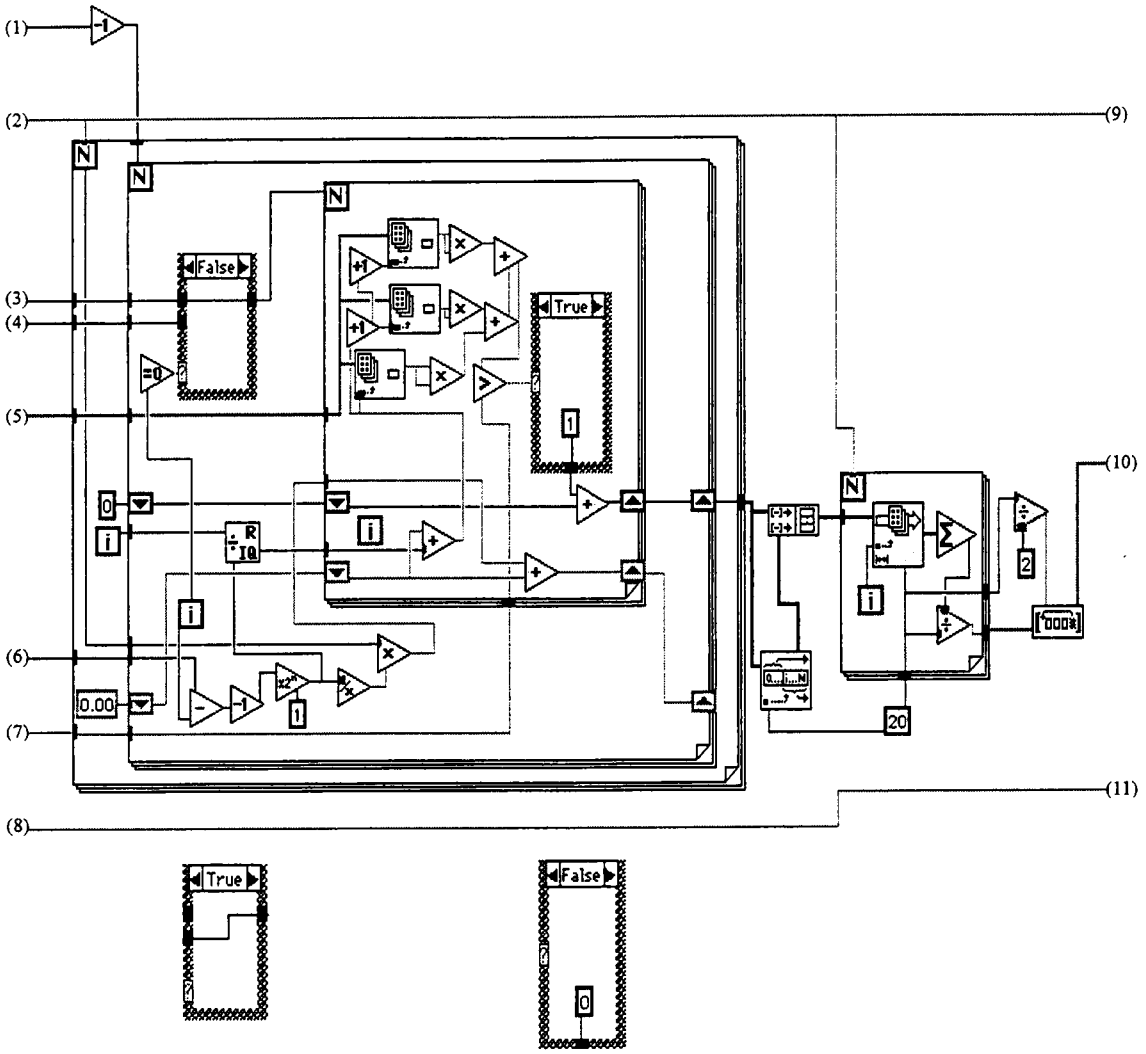


Figure C.2 (continued).

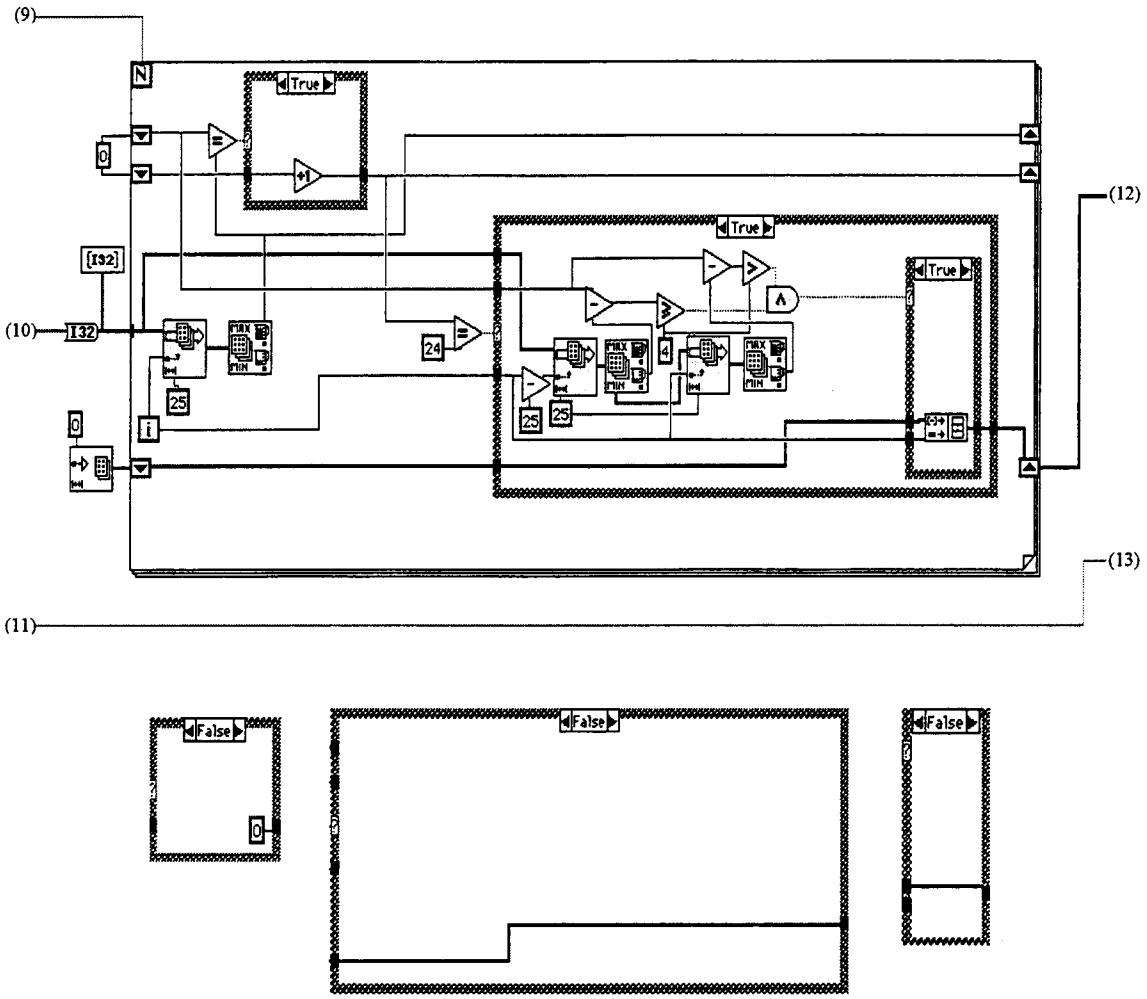


Figure C.2 (continued).

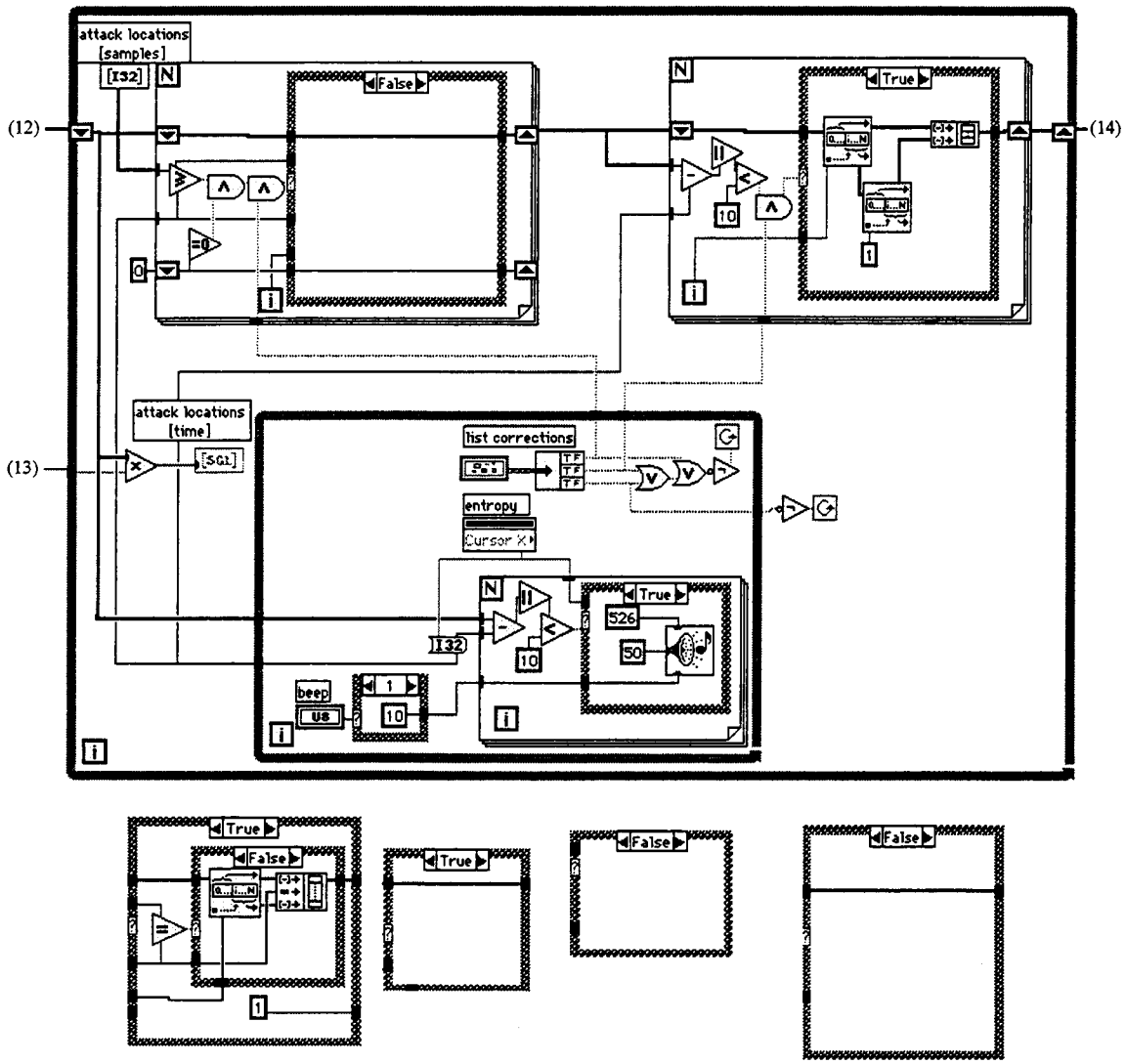


Figure C.2 (continued).

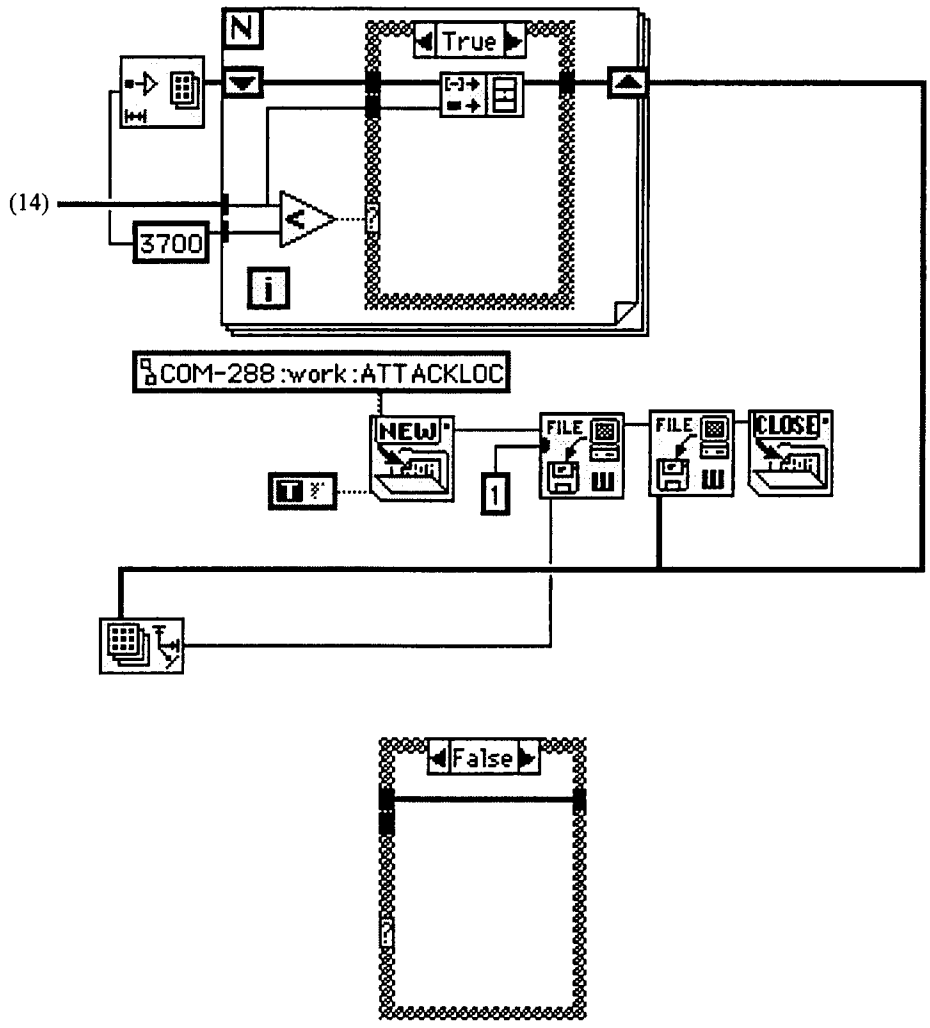


Figure C.2 (continued).

APPENDIX D

LABVIEW PROGRAM FOR PITCH DETECTION

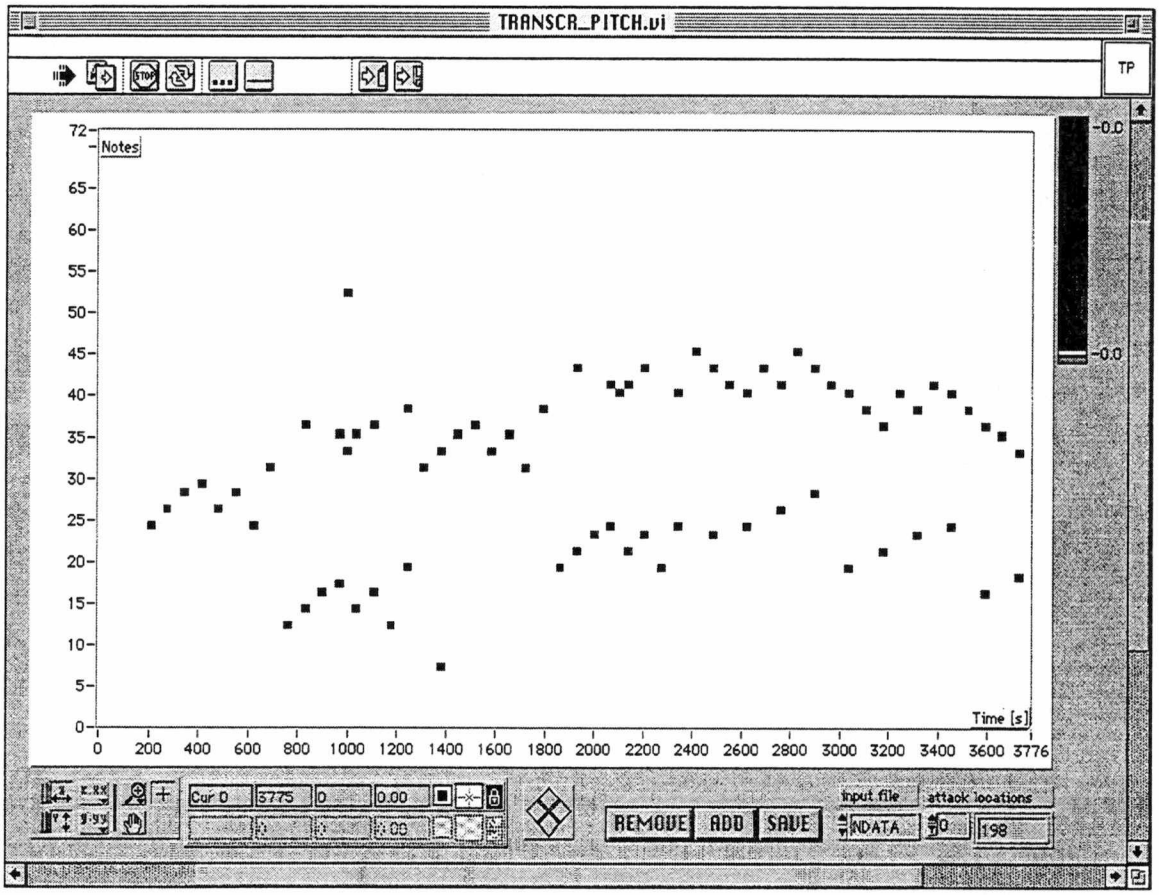


Figure D.1 Front panel of TRANSCR_PITCH.vi.

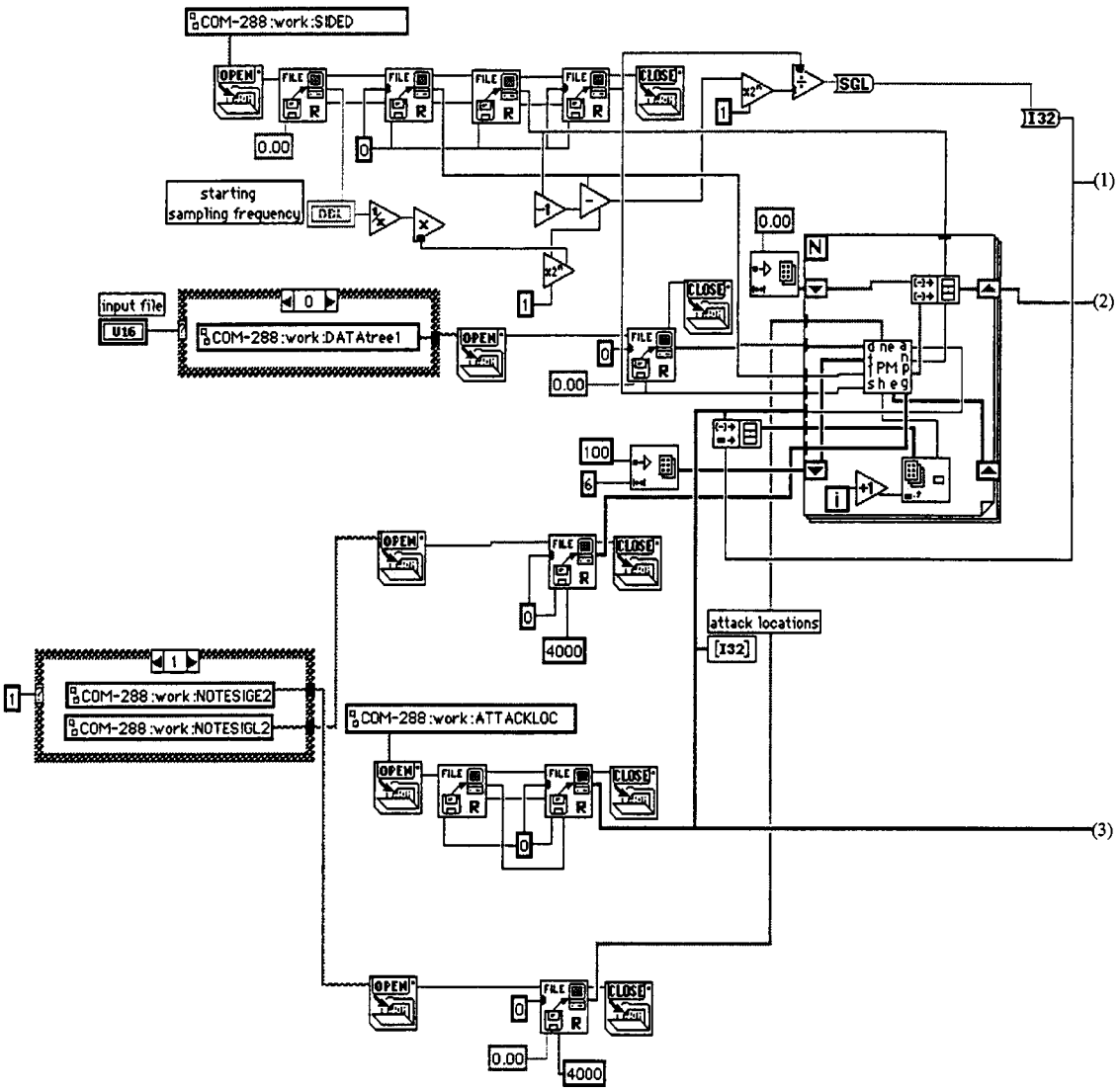


Figure D.2 Diagram of TRANSCR_PITCH.vi.

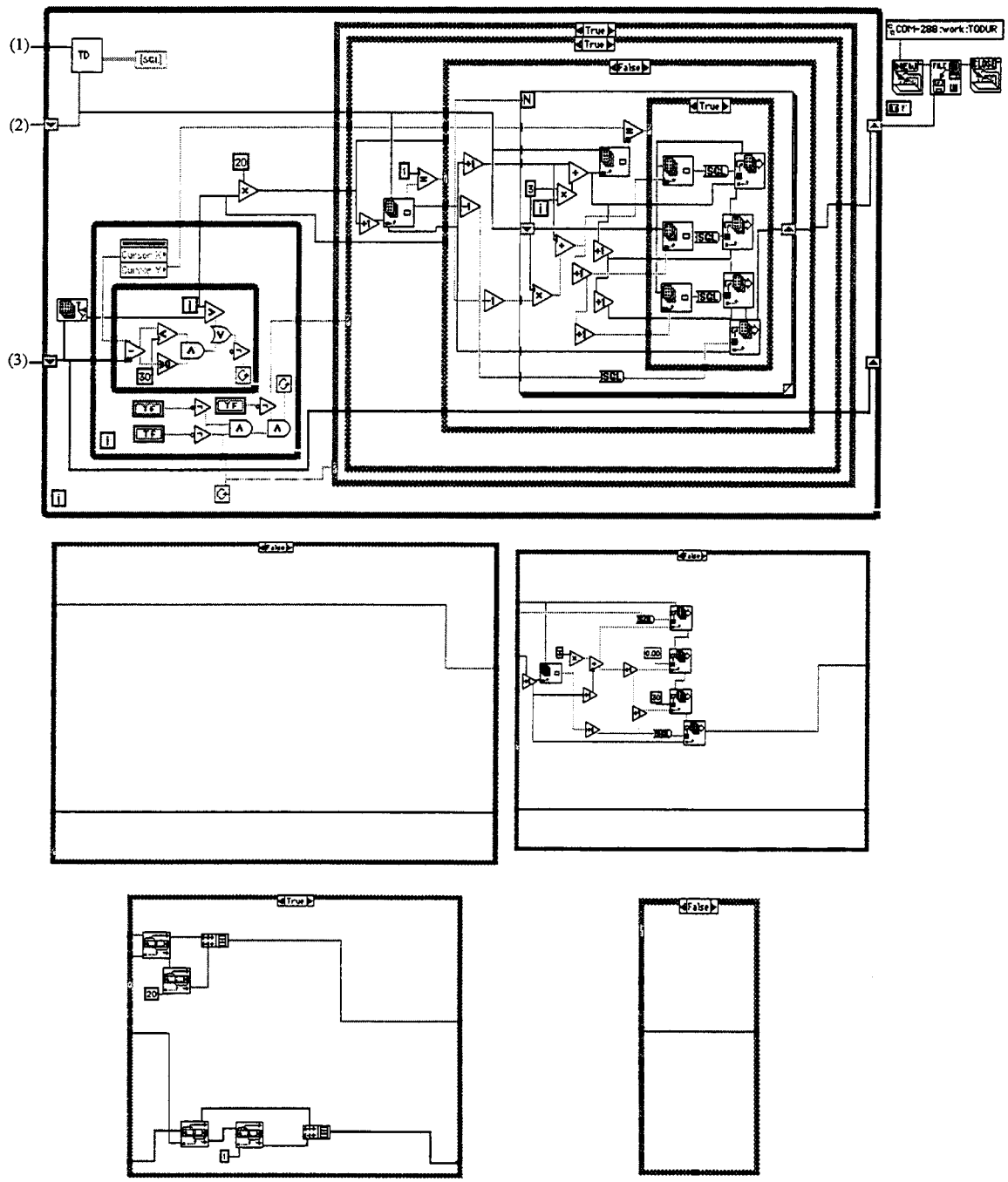


Figure D.2 (continued).

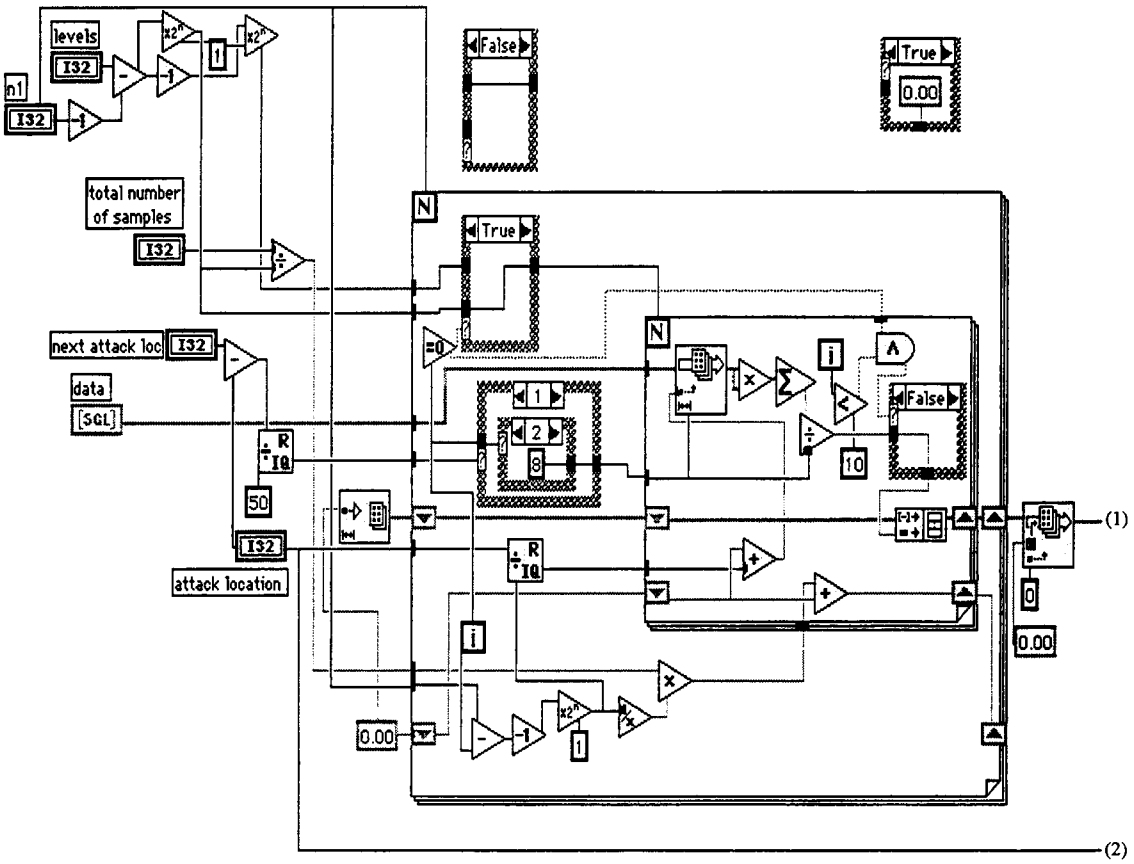


Figure D.3 Diagram of PM.vi.

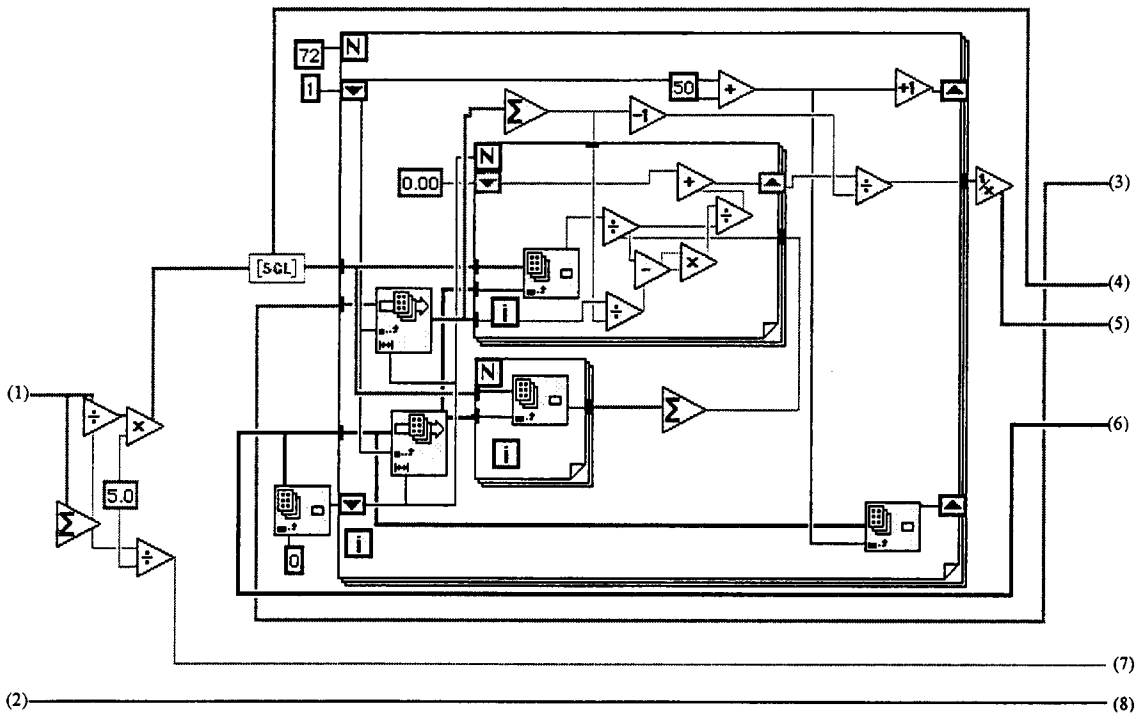


Figure D.3 (continued).

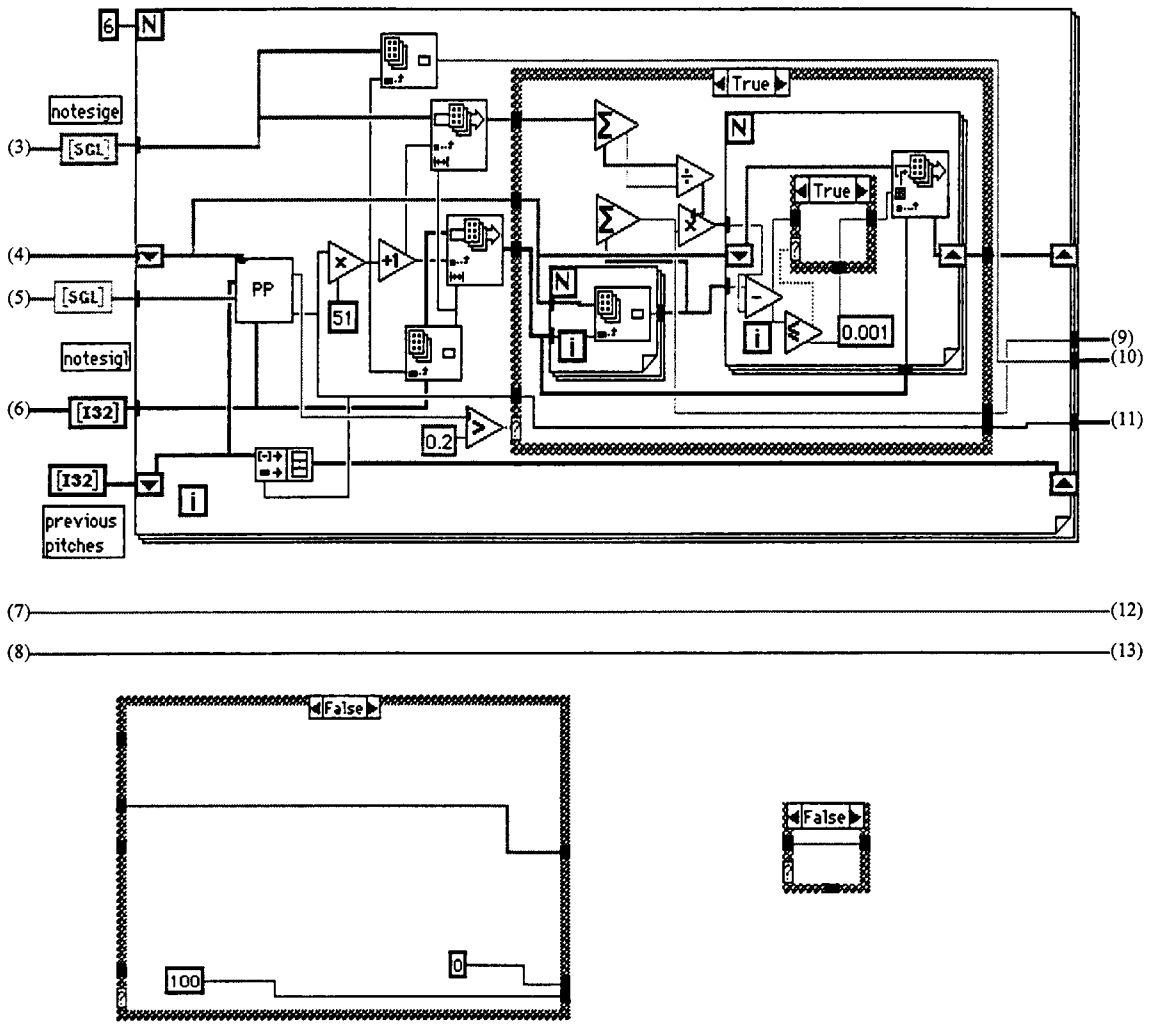


Figure D.3 (continued).

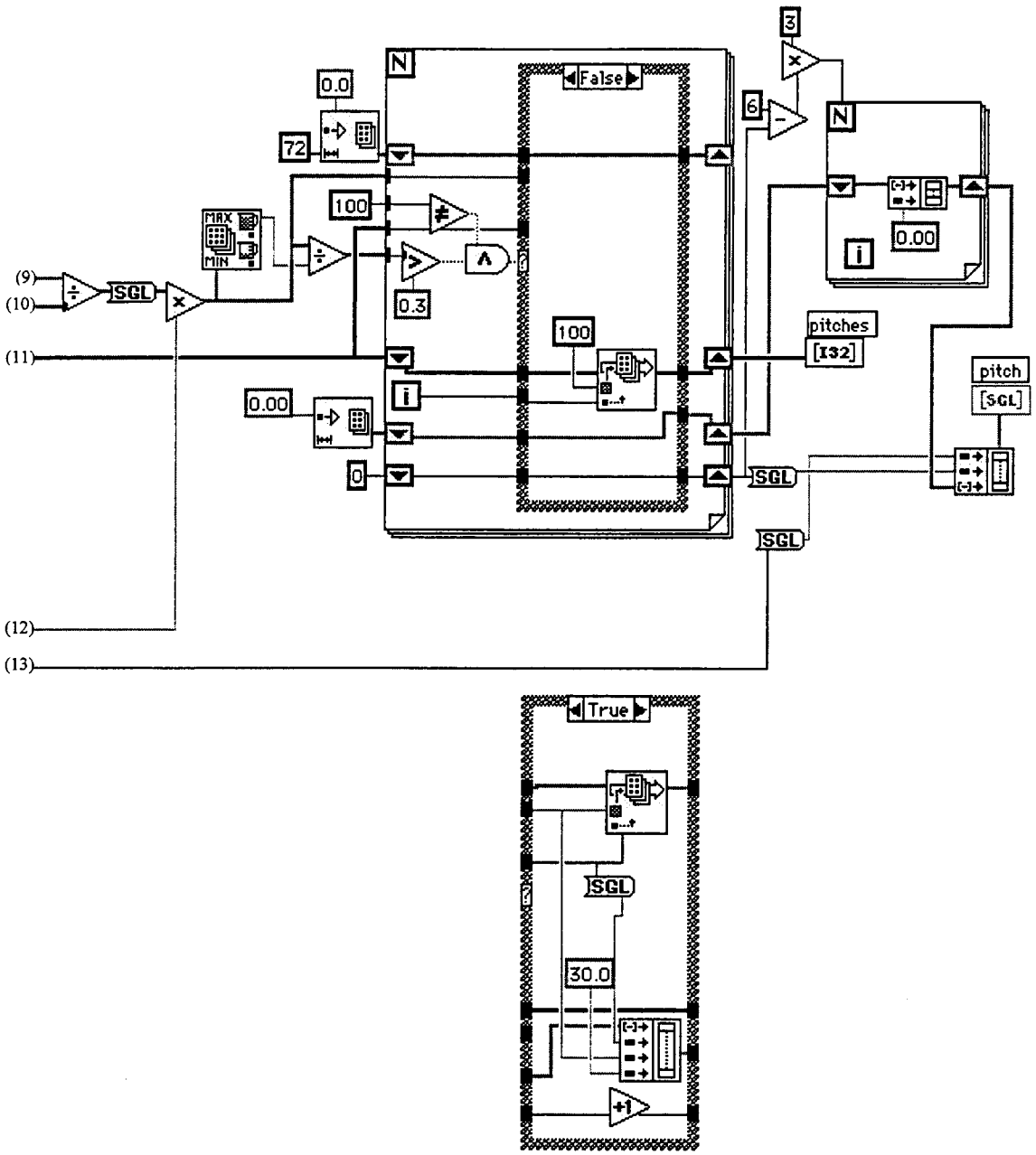


Figure D.3 (continued).

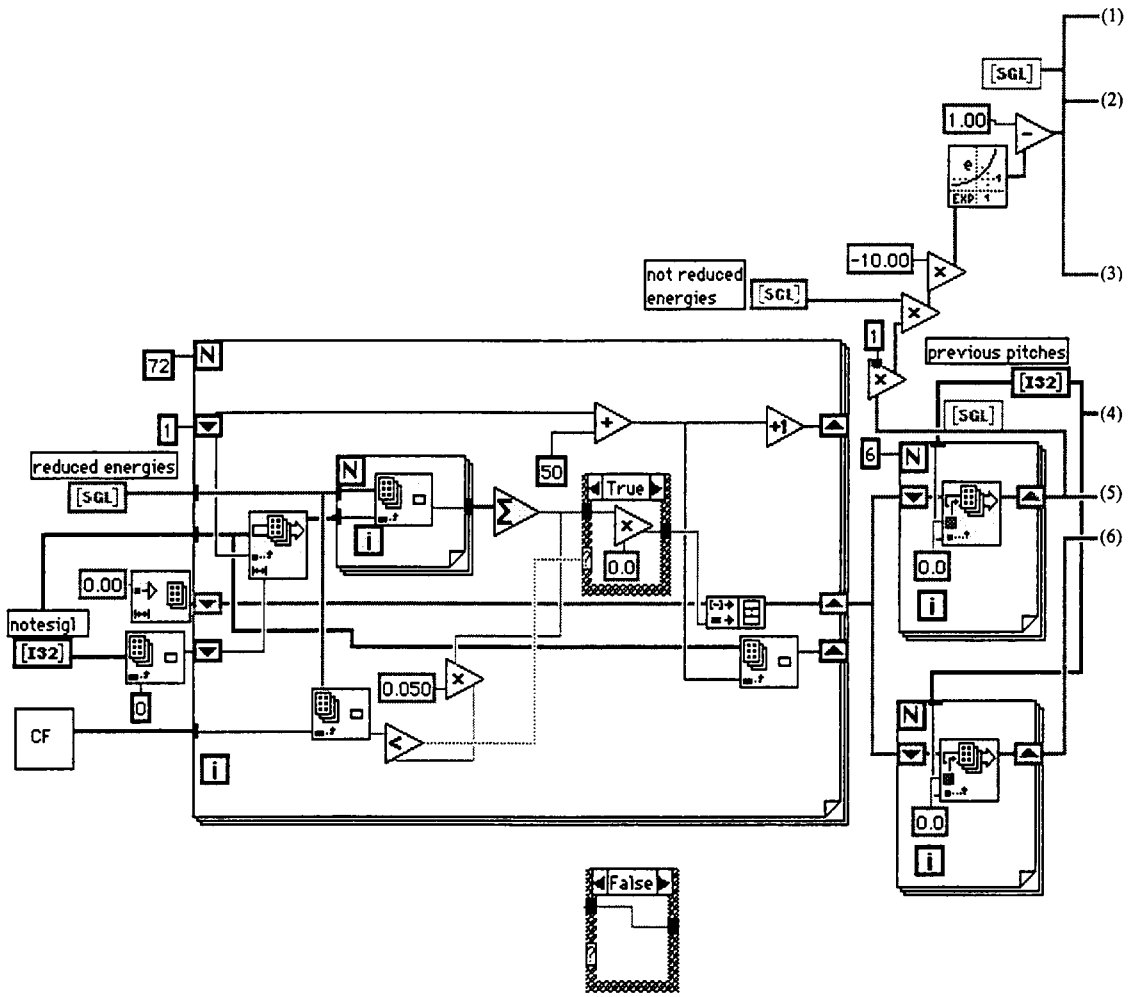


Figure D.4 Diagram of PP.vi.

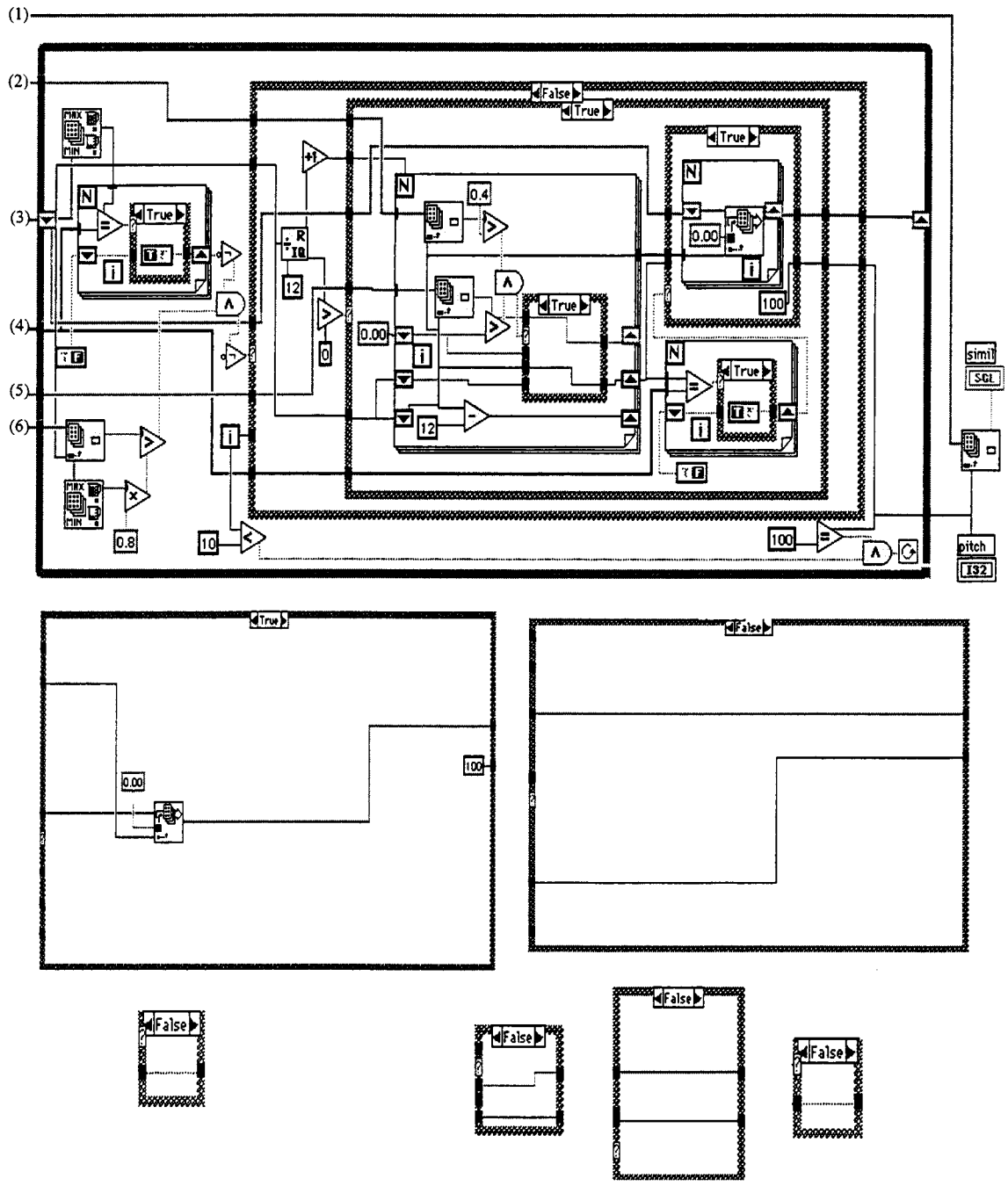


Figure D.4 (continued).

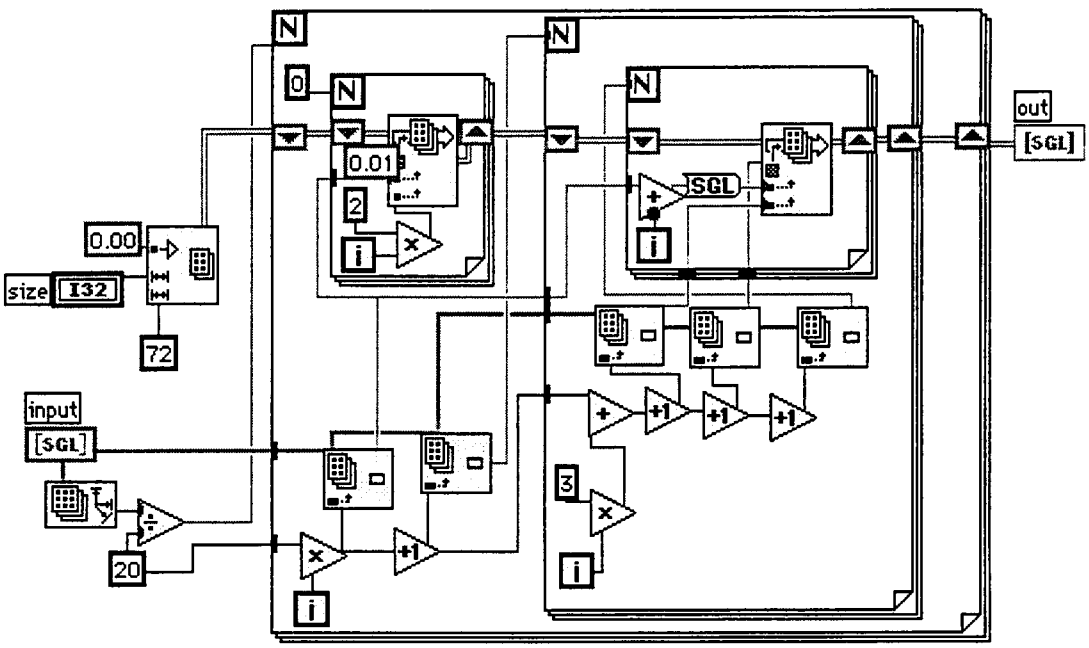


Figure D.5 Diagram of TD.vi

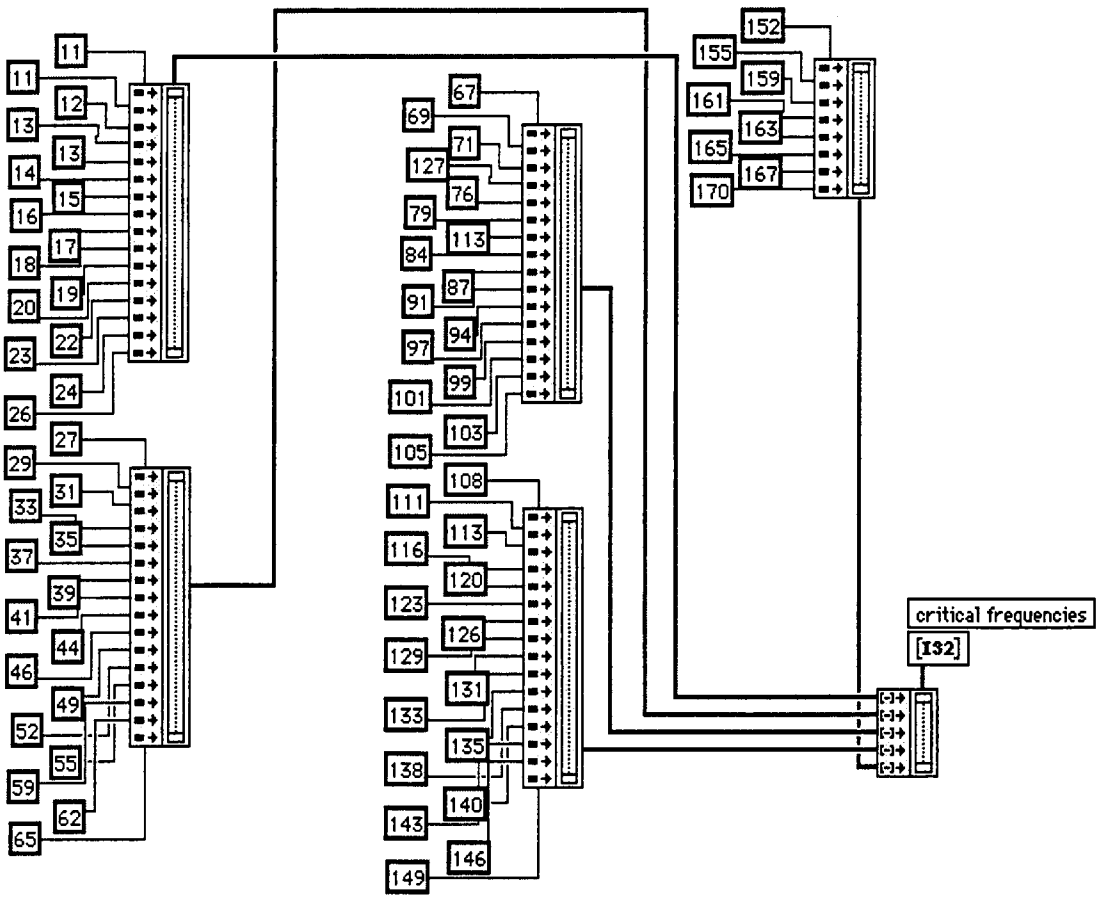


Figure D.6 Diagram of CF.vi.

APPENDIX E

LABVIEW PROGRAM FOR DURATION DETECTION

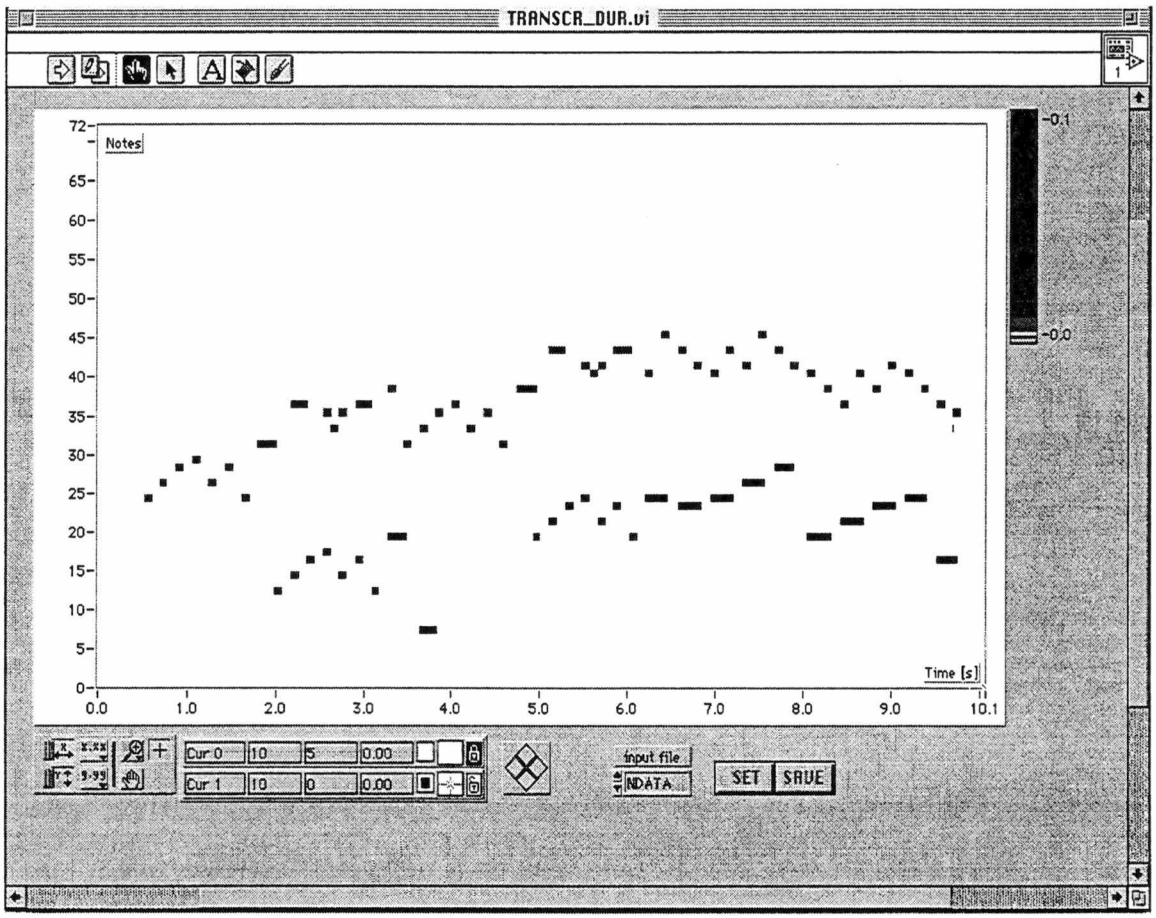


Figure E.1 Front panel of TRANSCR_DUR.vi.

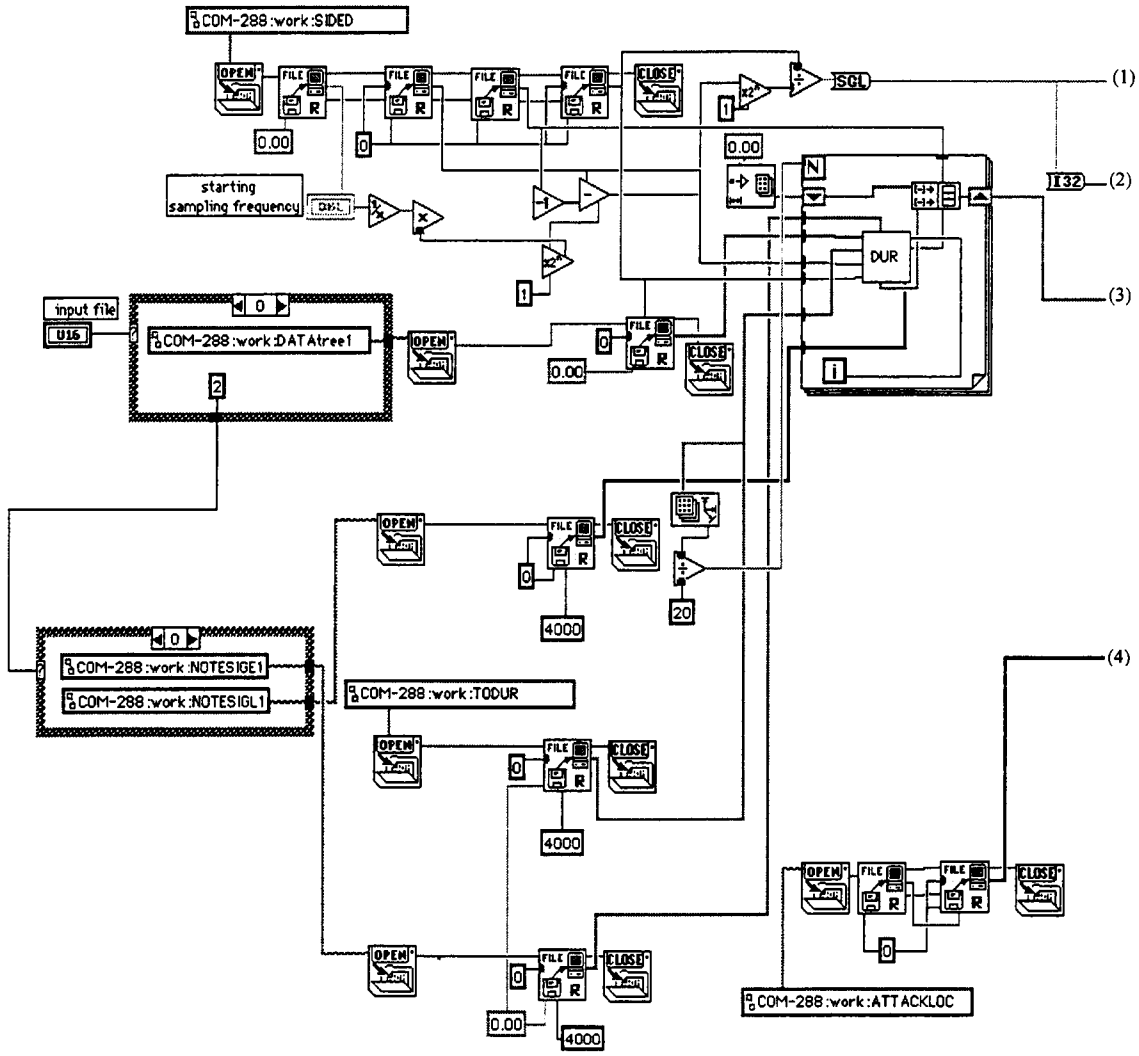


Figure E.2 Diagram of TRANSCR_DUR.vi.

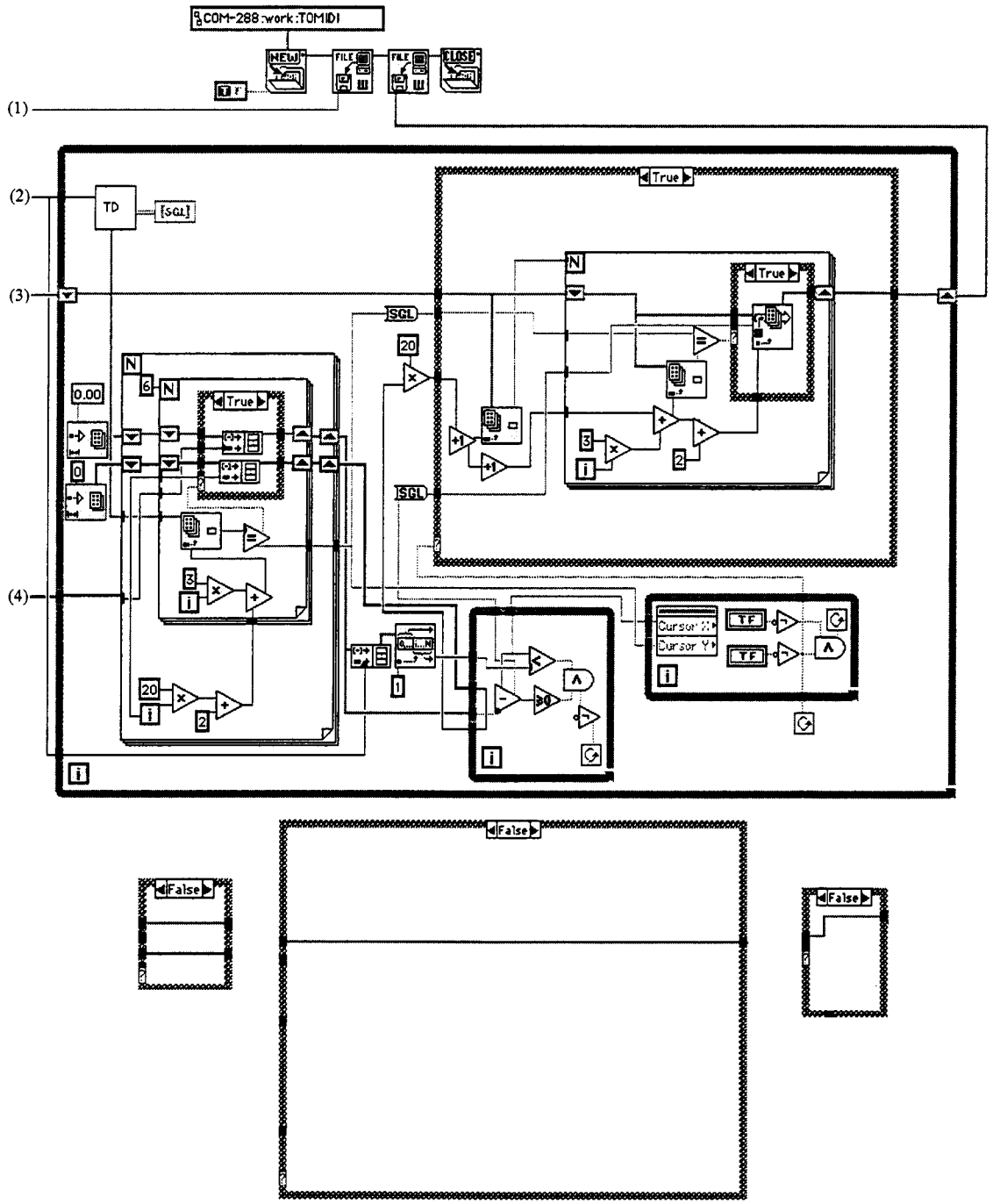


Figure E.2 (continued).

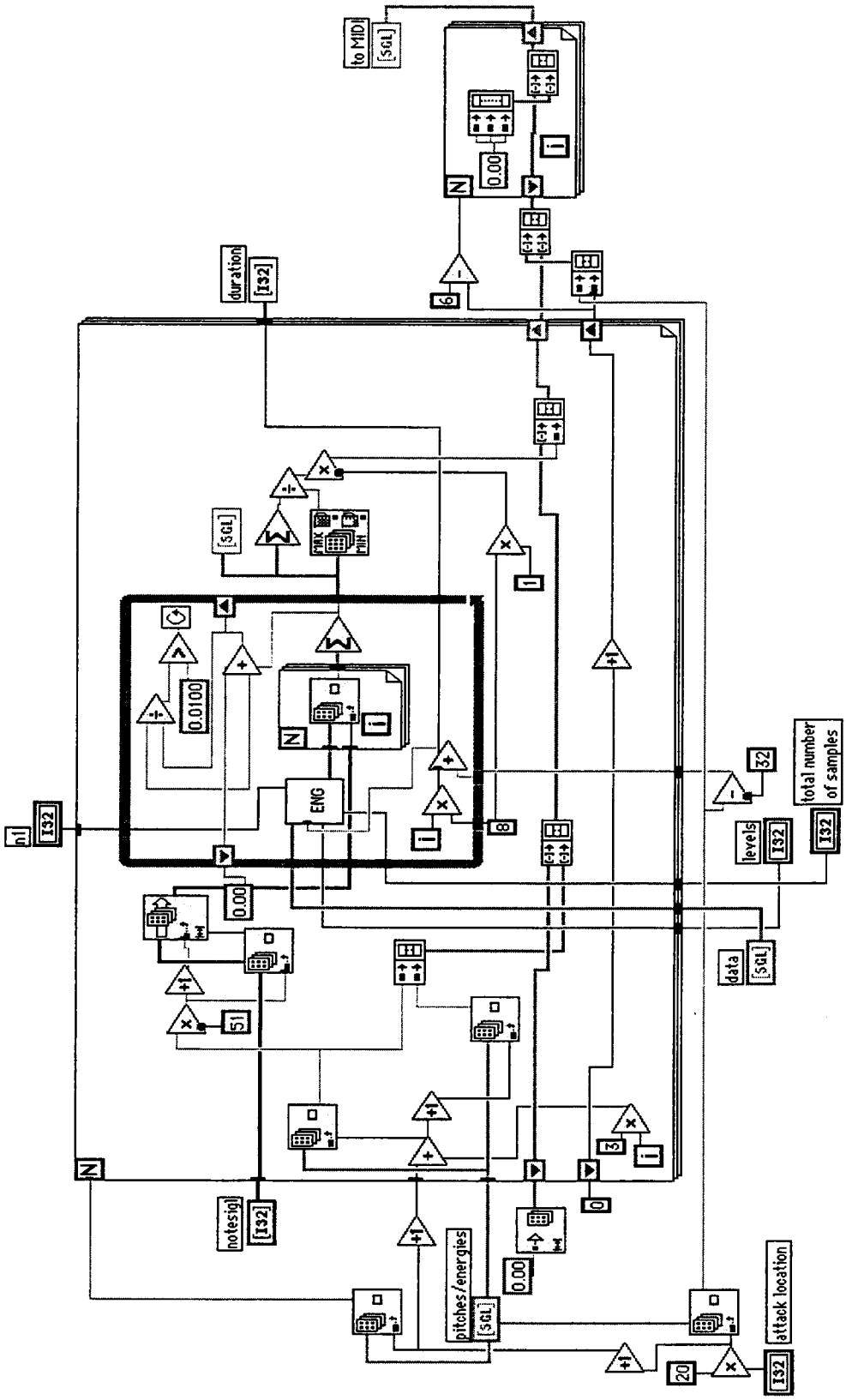


Figure E.3 Diagram of DUR.vi

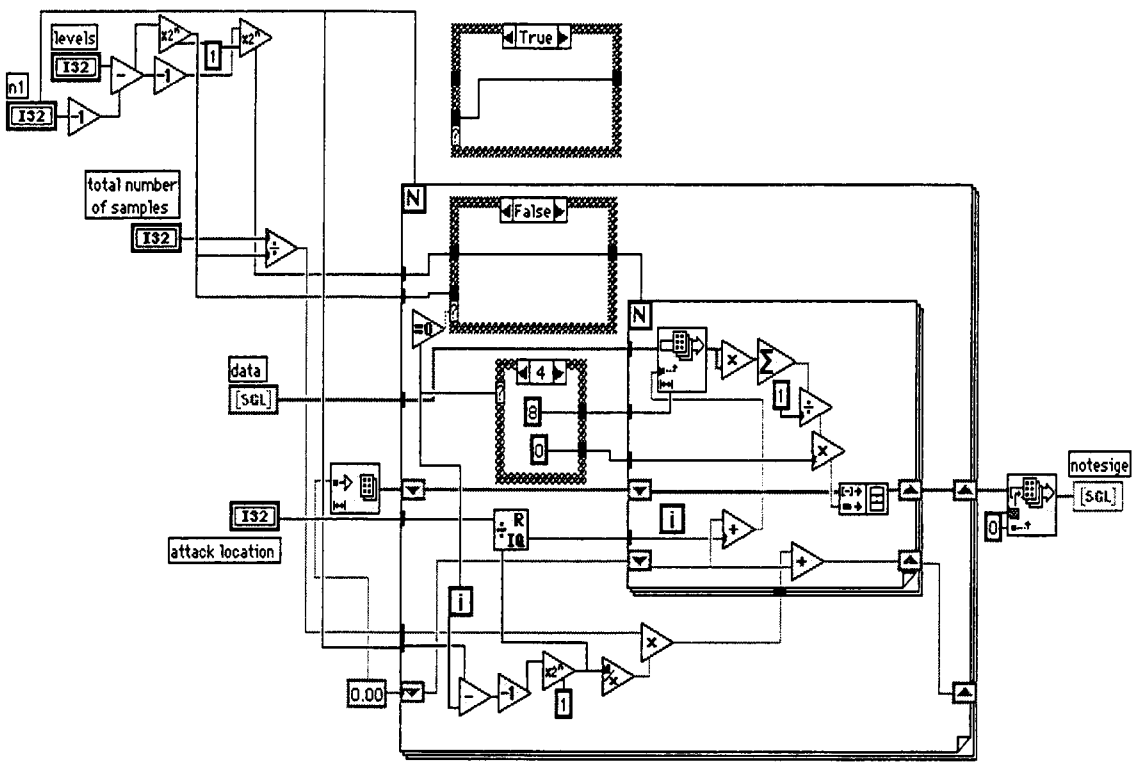


Figure E.4 Diagram of ENG.vi.

APPENDIX F

LABVIEW PROGRAM FOR MIDI CODING

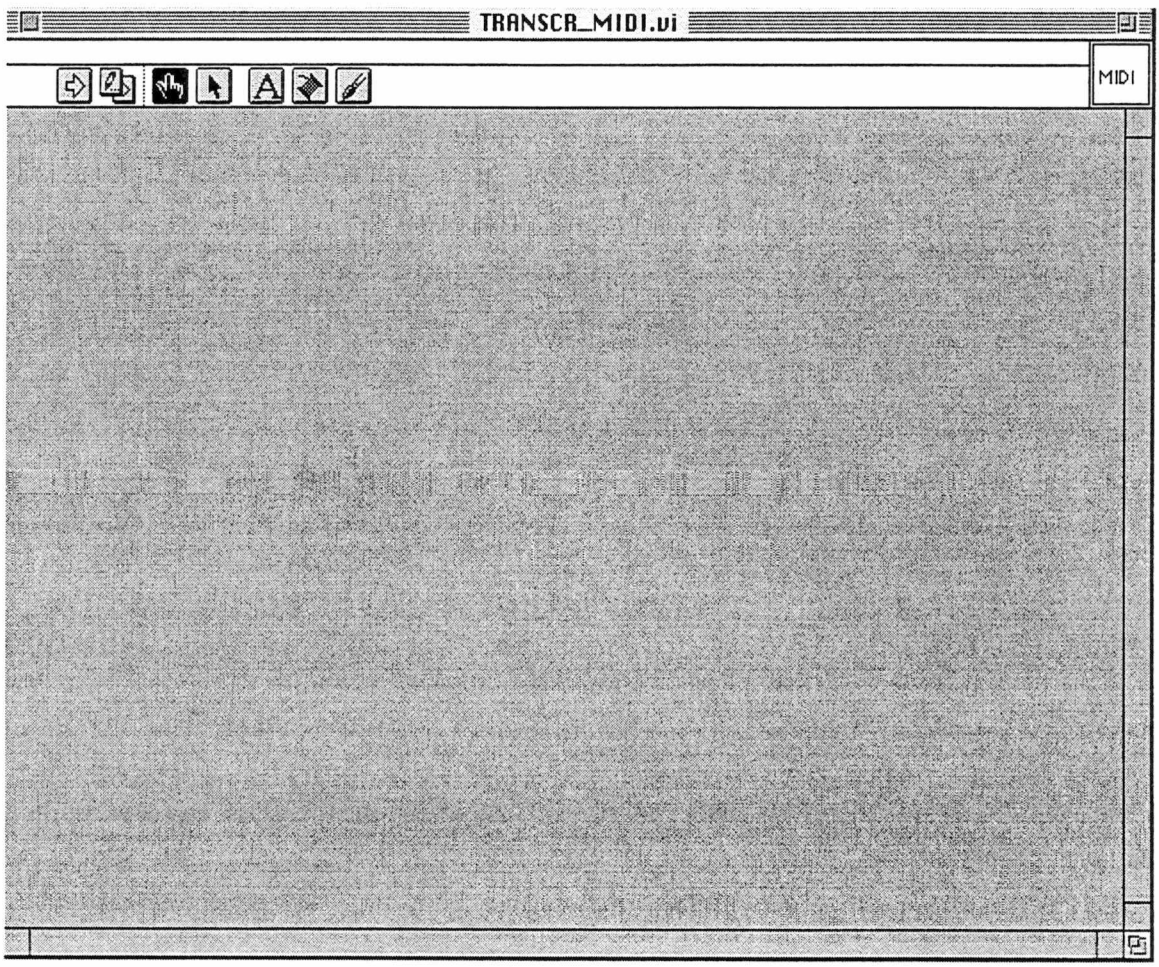


Figure F.1 Front panel of TRANSCR_MIDI.vi.

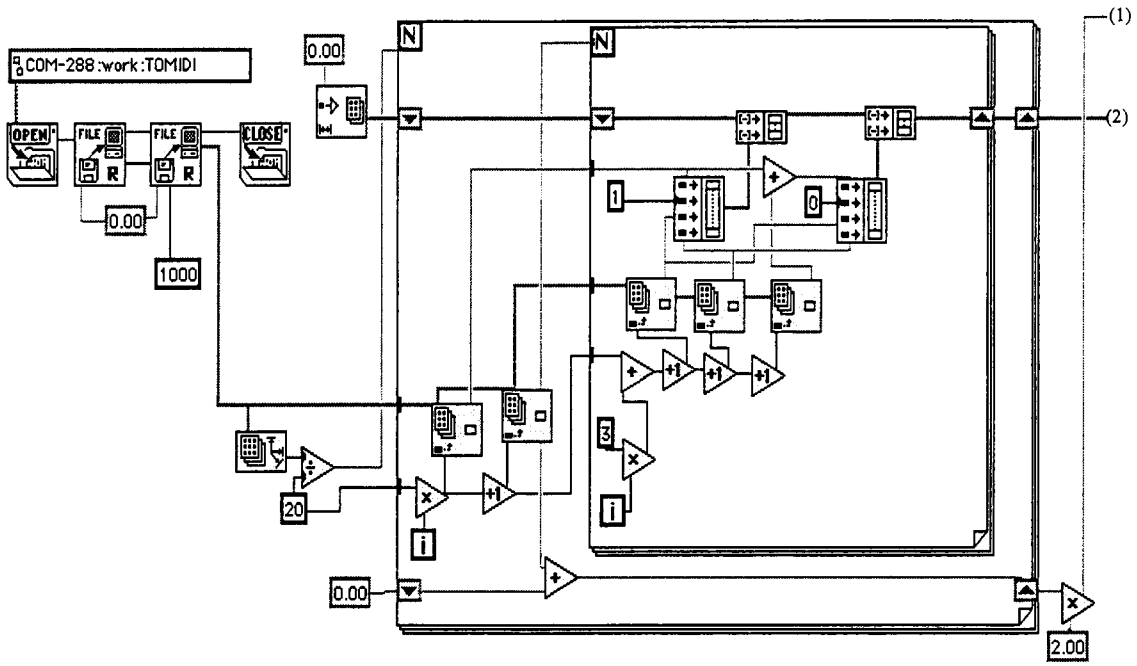


Figure F.2 Diagram of TRANSCR_MIDI.vi.

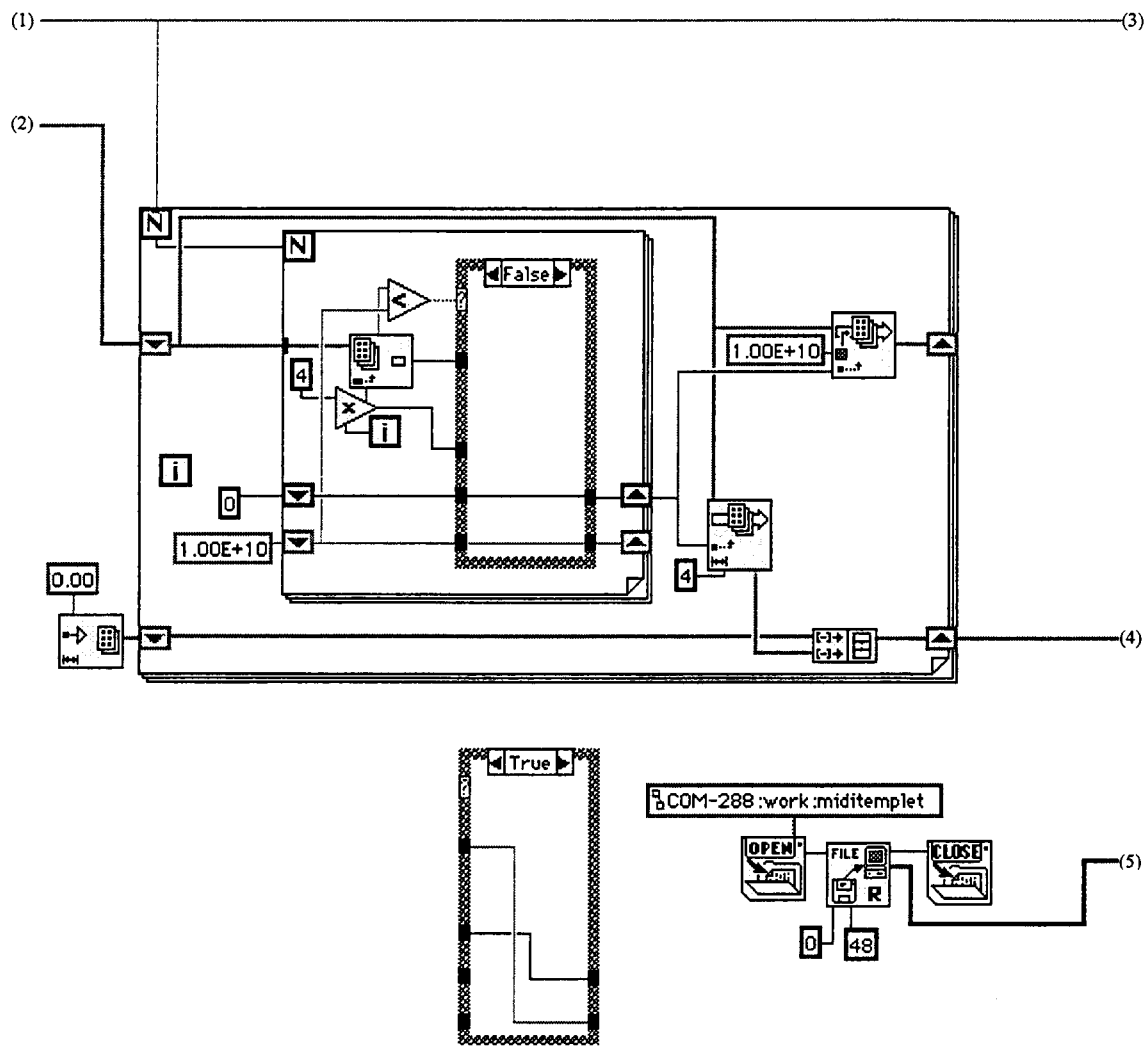


Figure F.2 (continued).

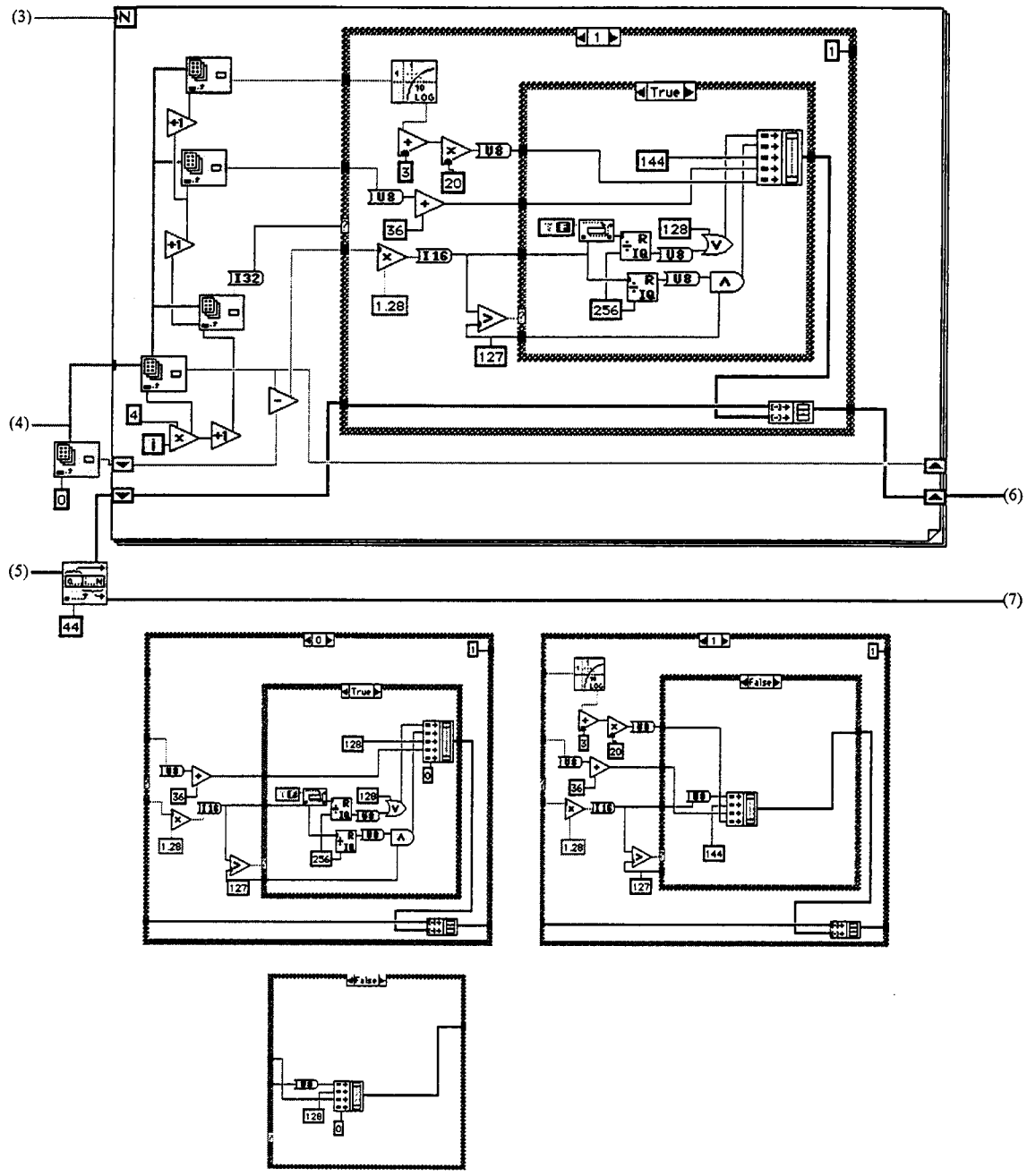


Figure F.2 (continued).

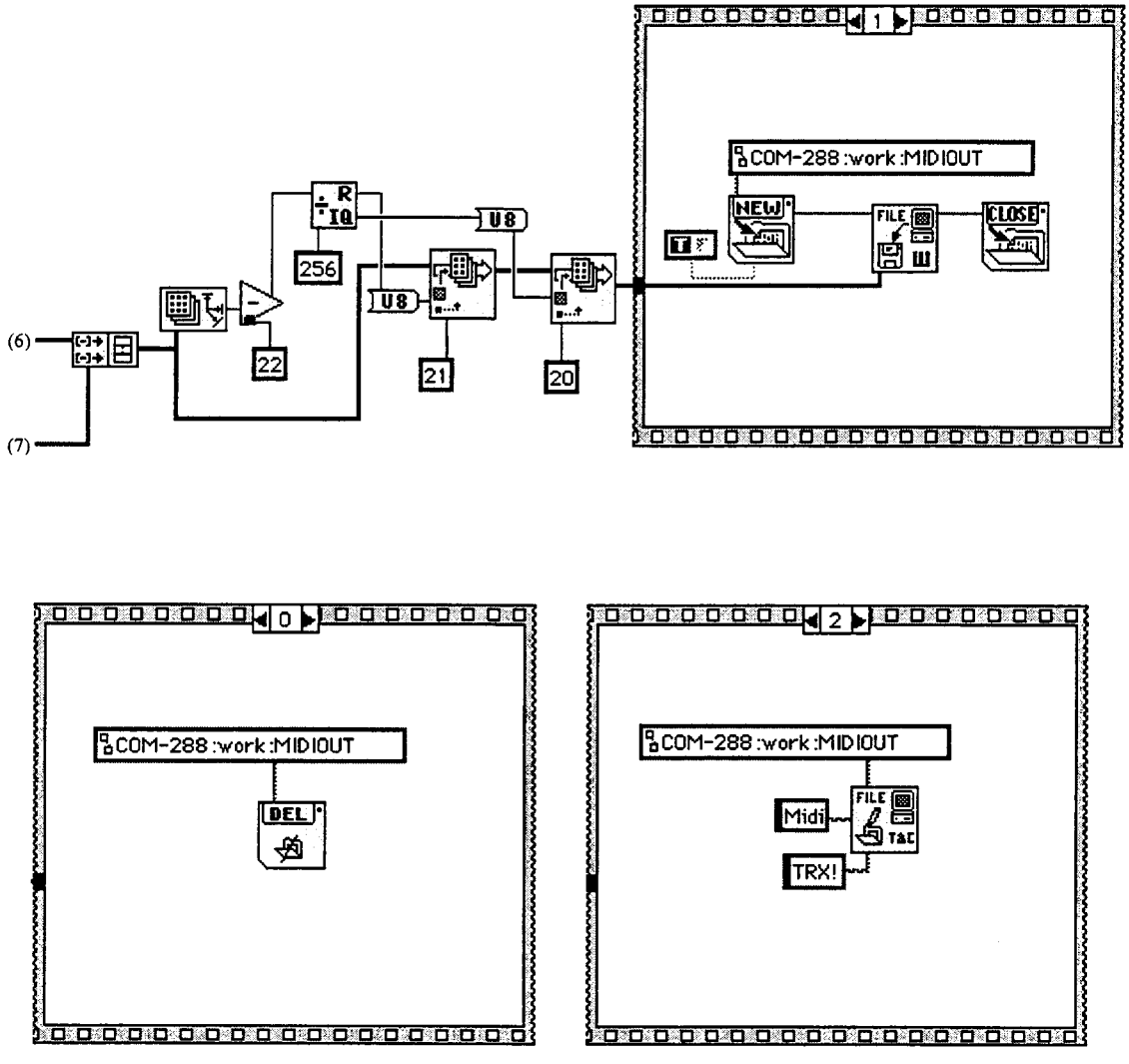


Figure F.2 (continued).

APPENDIX G

LABVIEW PROGRAM OF TIME-FREQUENCY EDITOR

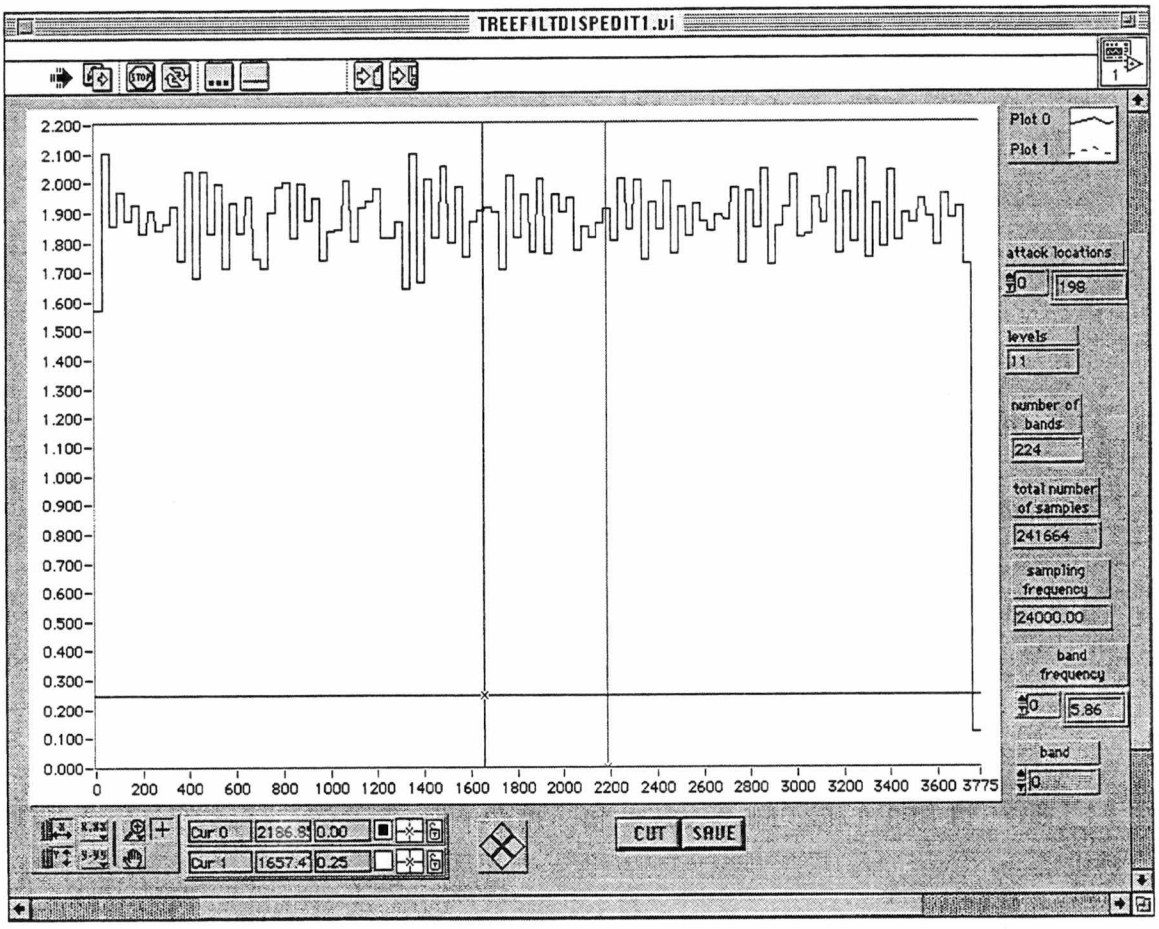


Figure G.1 Front panel of TREEFILTDISPEDIT1.vi.

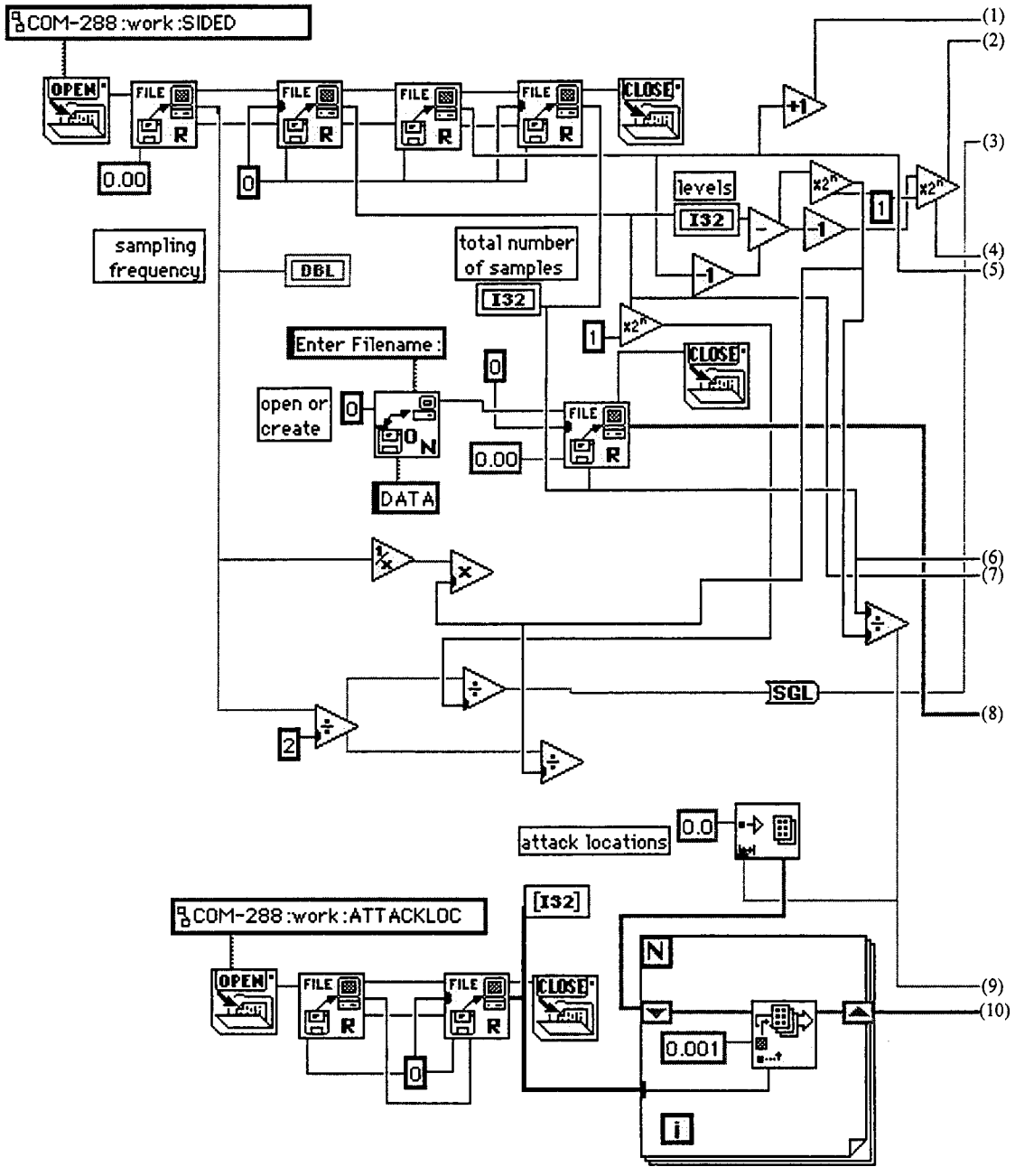


Figure G.2 Diagram of TREEFILTDISPEDIT1.vi.

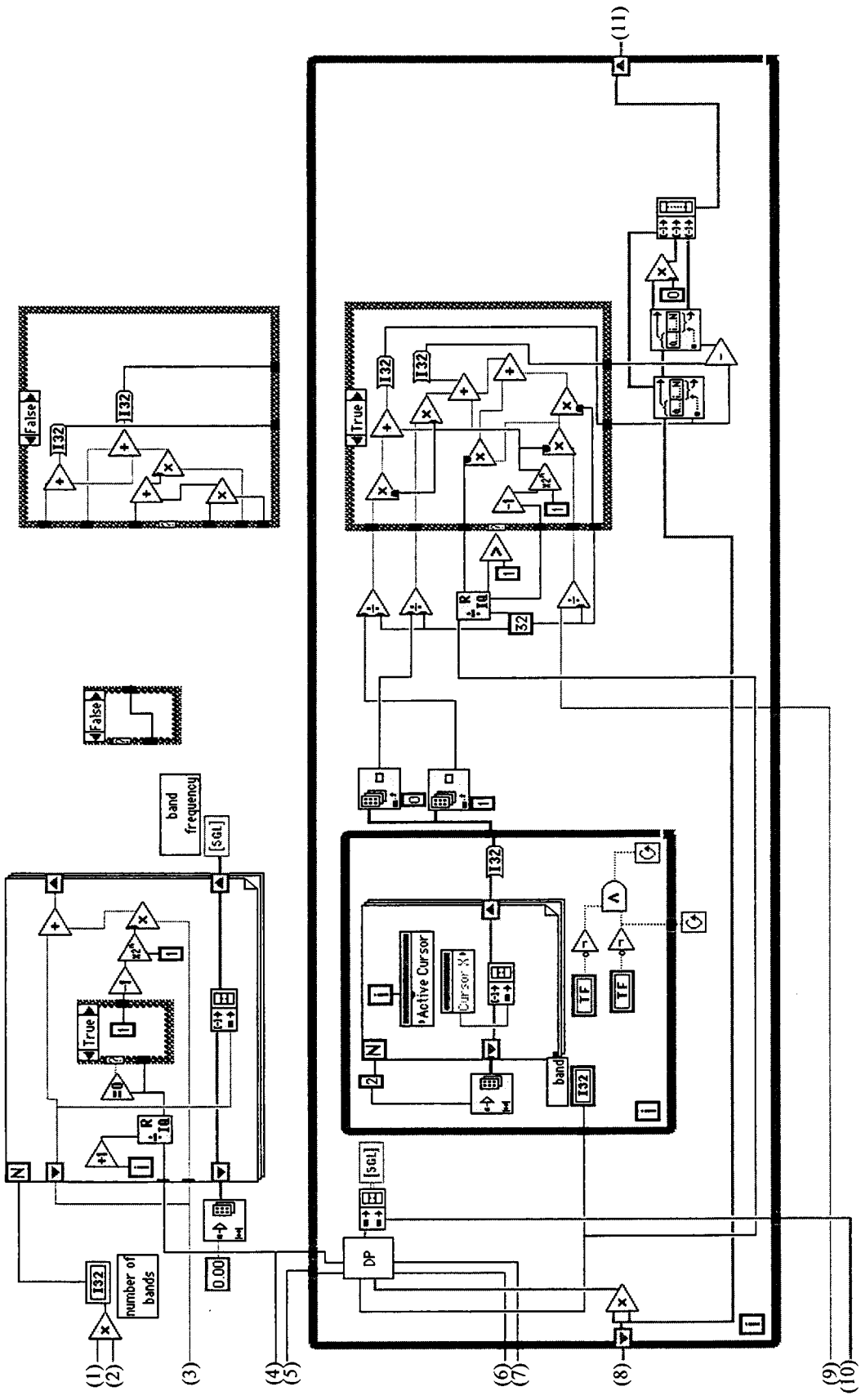


Figure G.2 (continued).

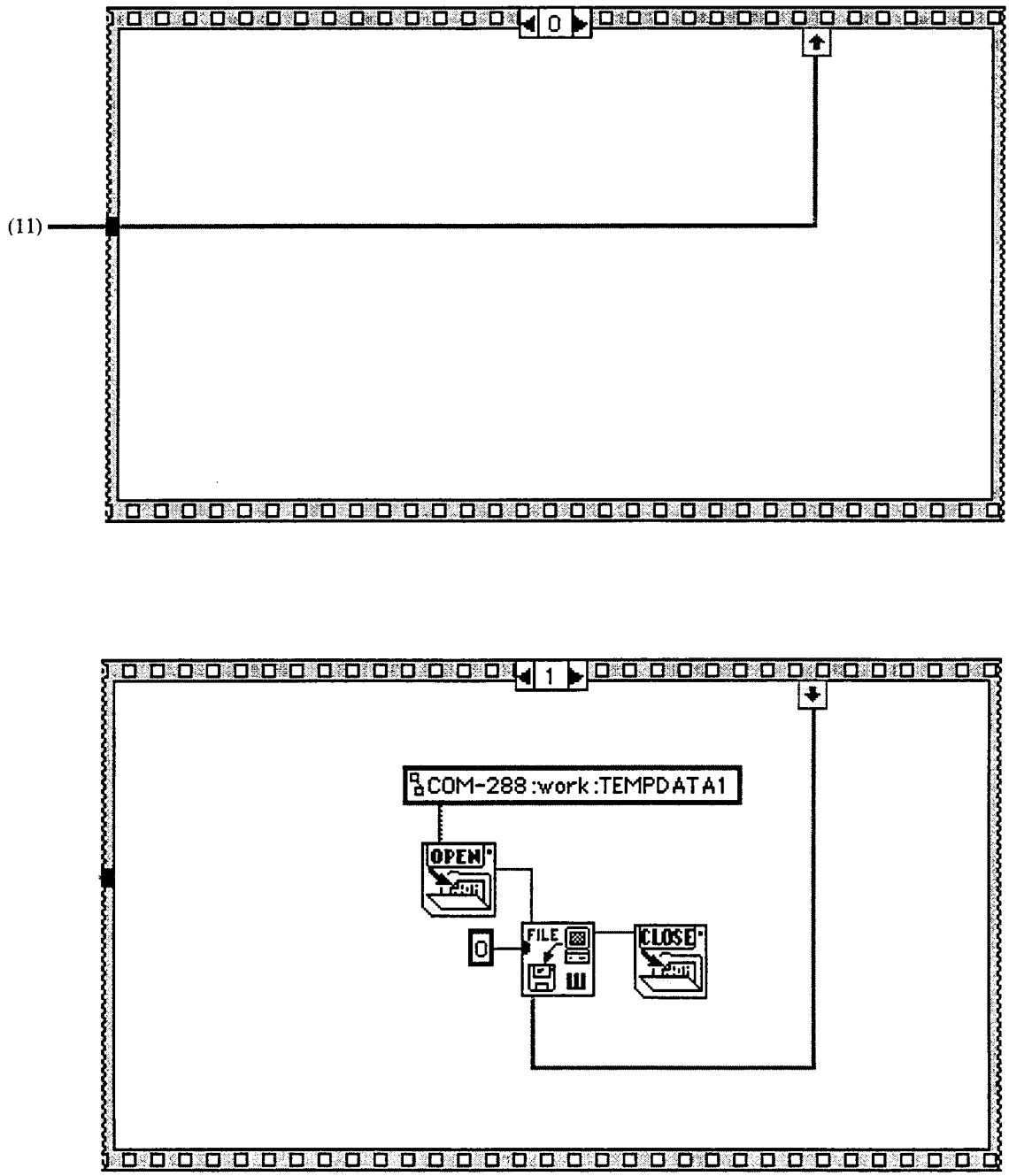


Figure G.2 (continued).

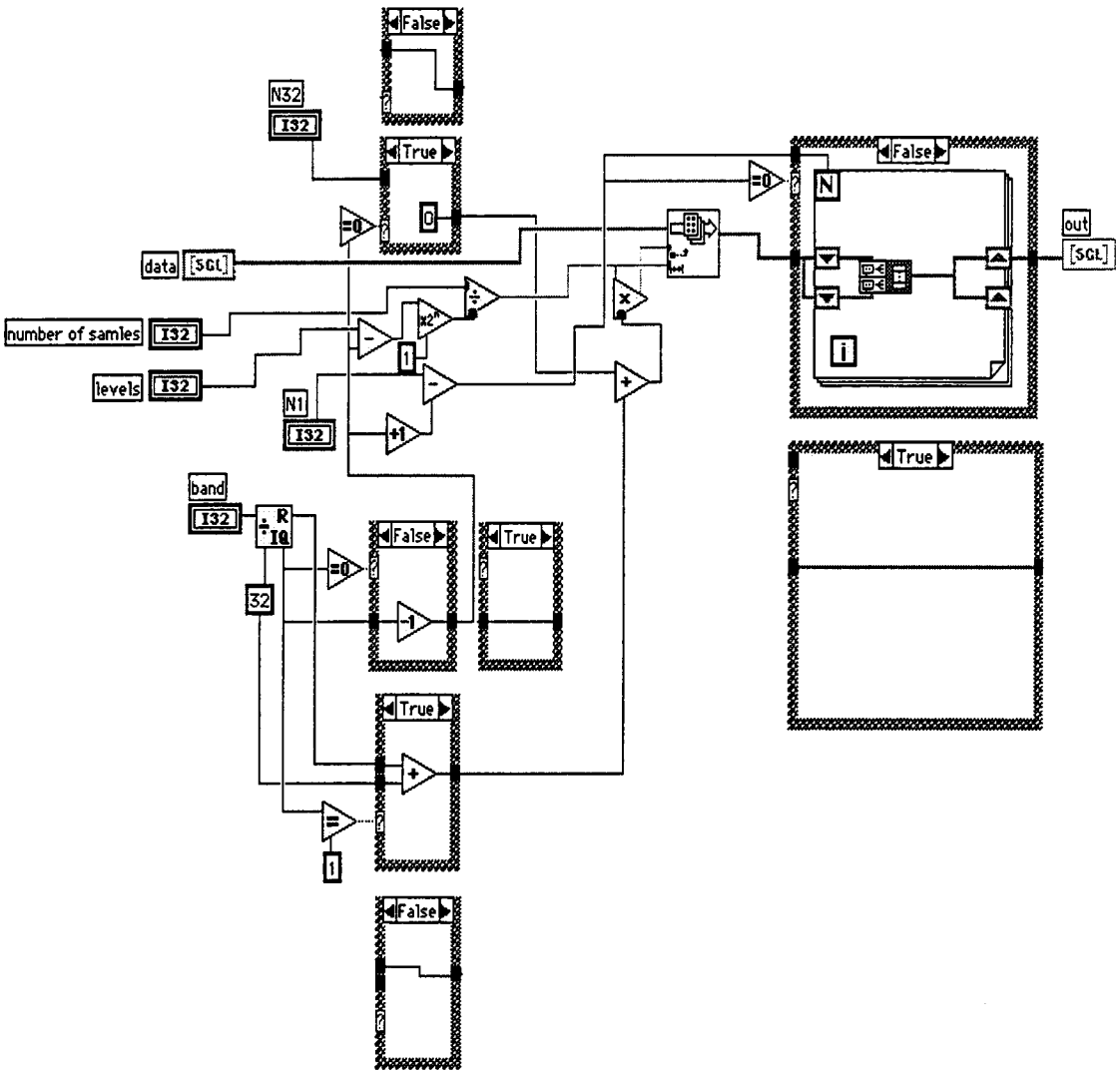


Figure G.3 Diagram of DP.vi.

APPENDIX H

LABVIEW PROGRAM FOR TREE-STRUCTURED FILTER BANK IMPLEMENTATION (SYNTHESIS PART)

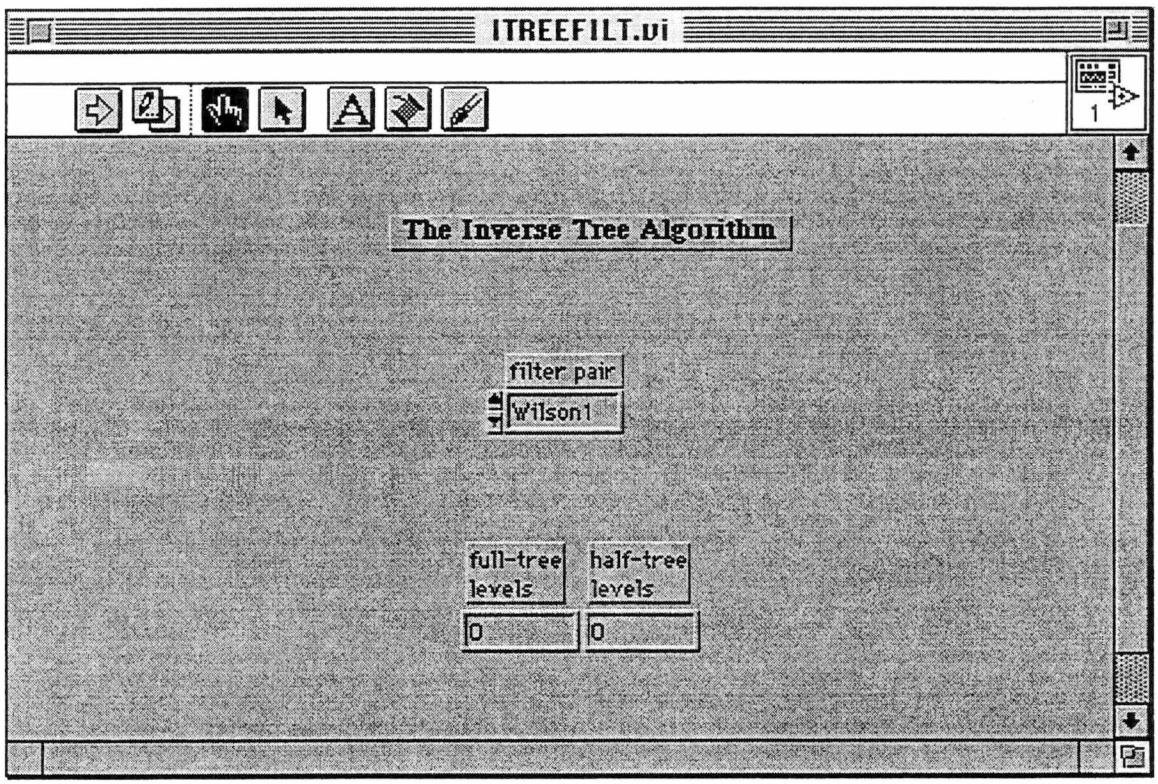


Figure H.1 Front panel of ITREEFILT.vi.

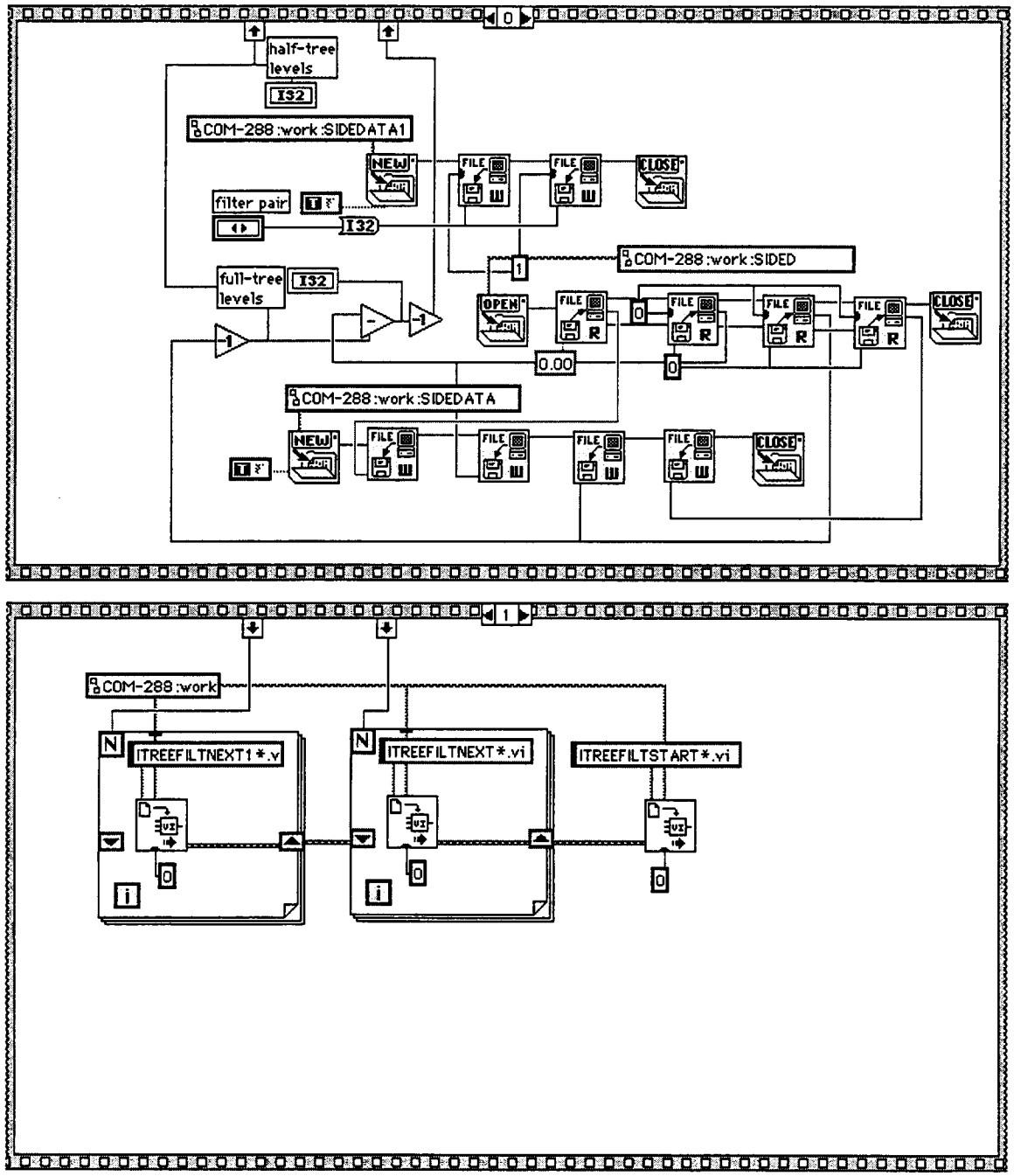


Figure H.2 Diagram of ITREEFILT.vi.

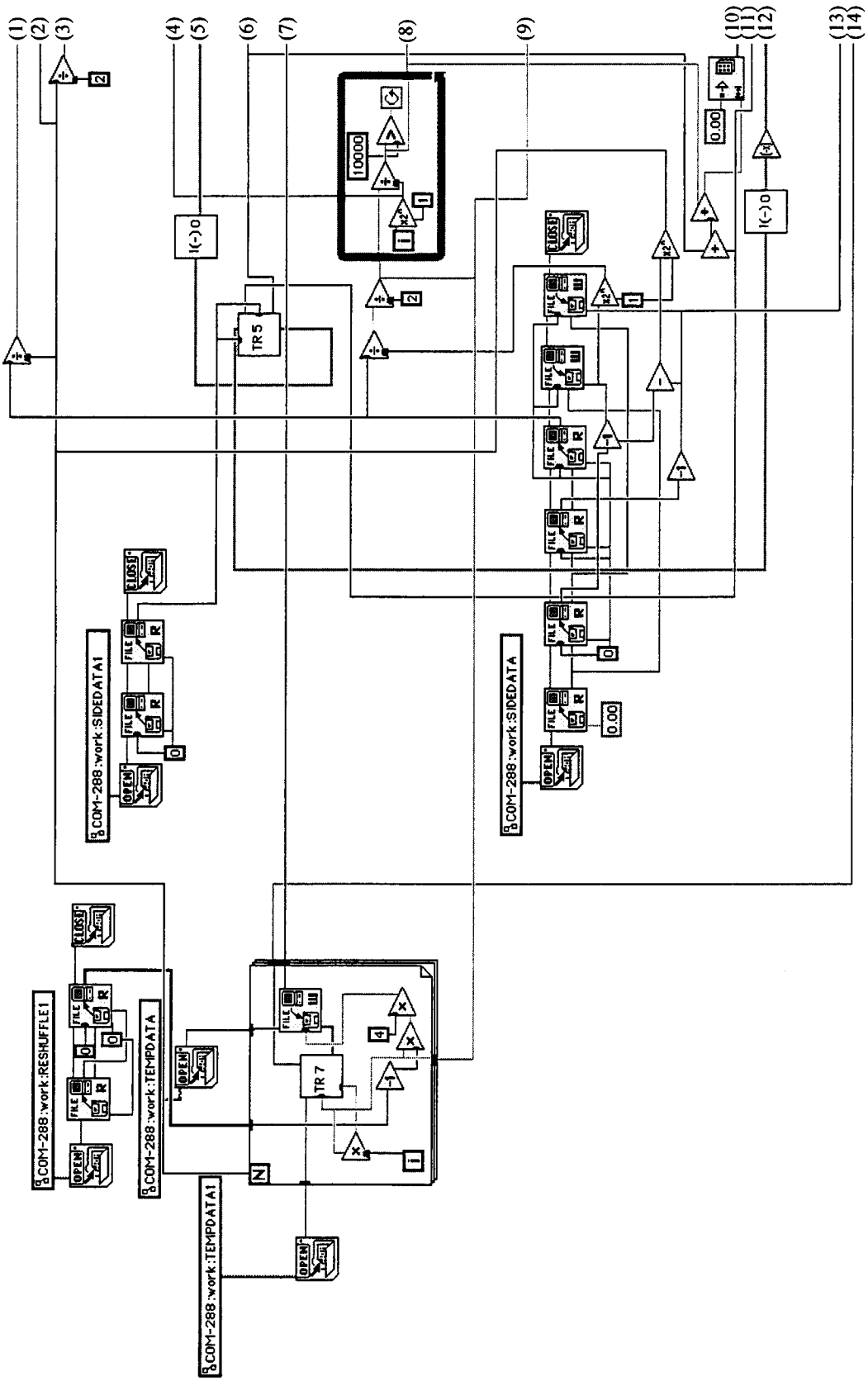


Figure H.3 Diagram of ITREEFILTNEXT1*.vi.

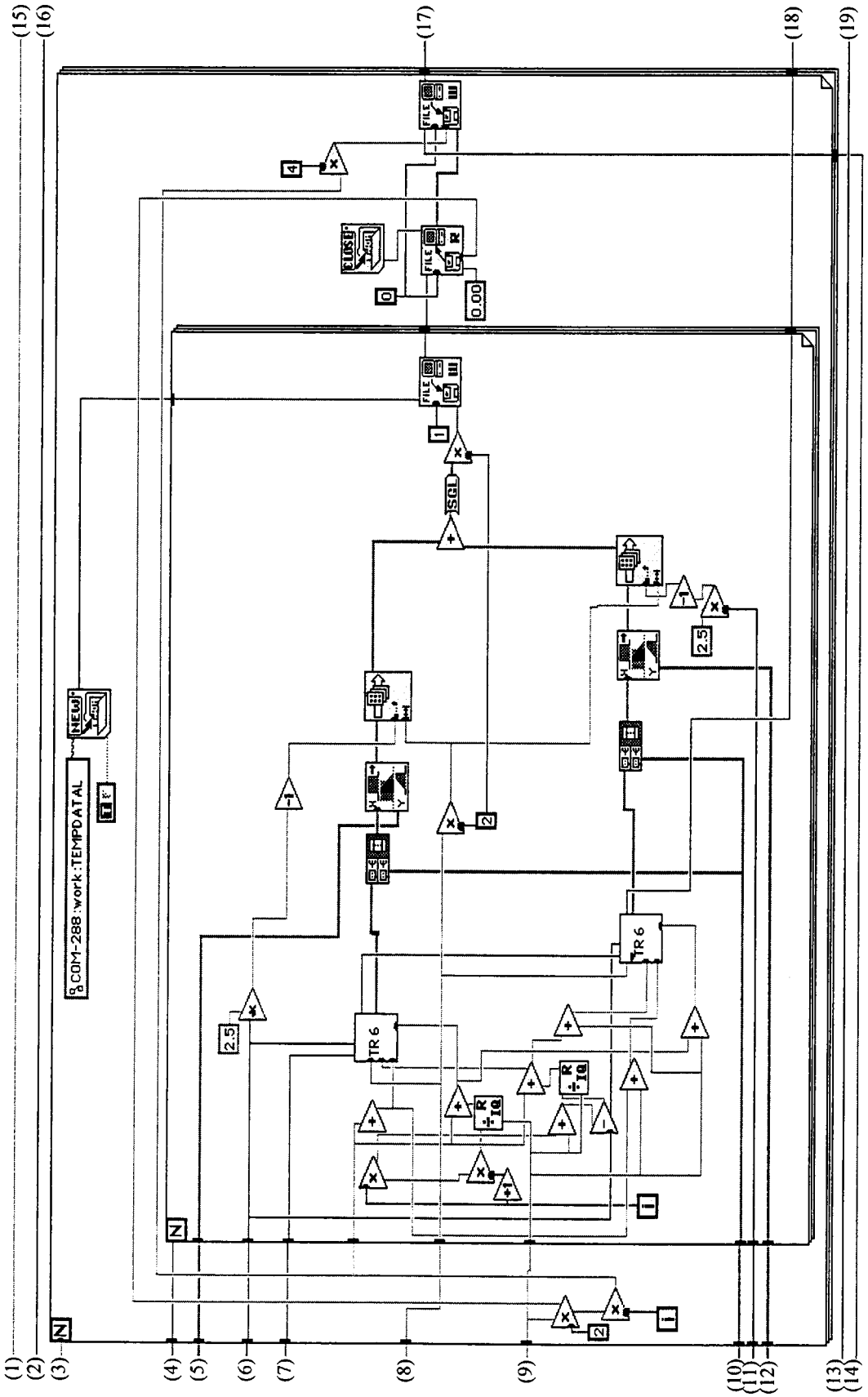


Figure H.3 (continued).

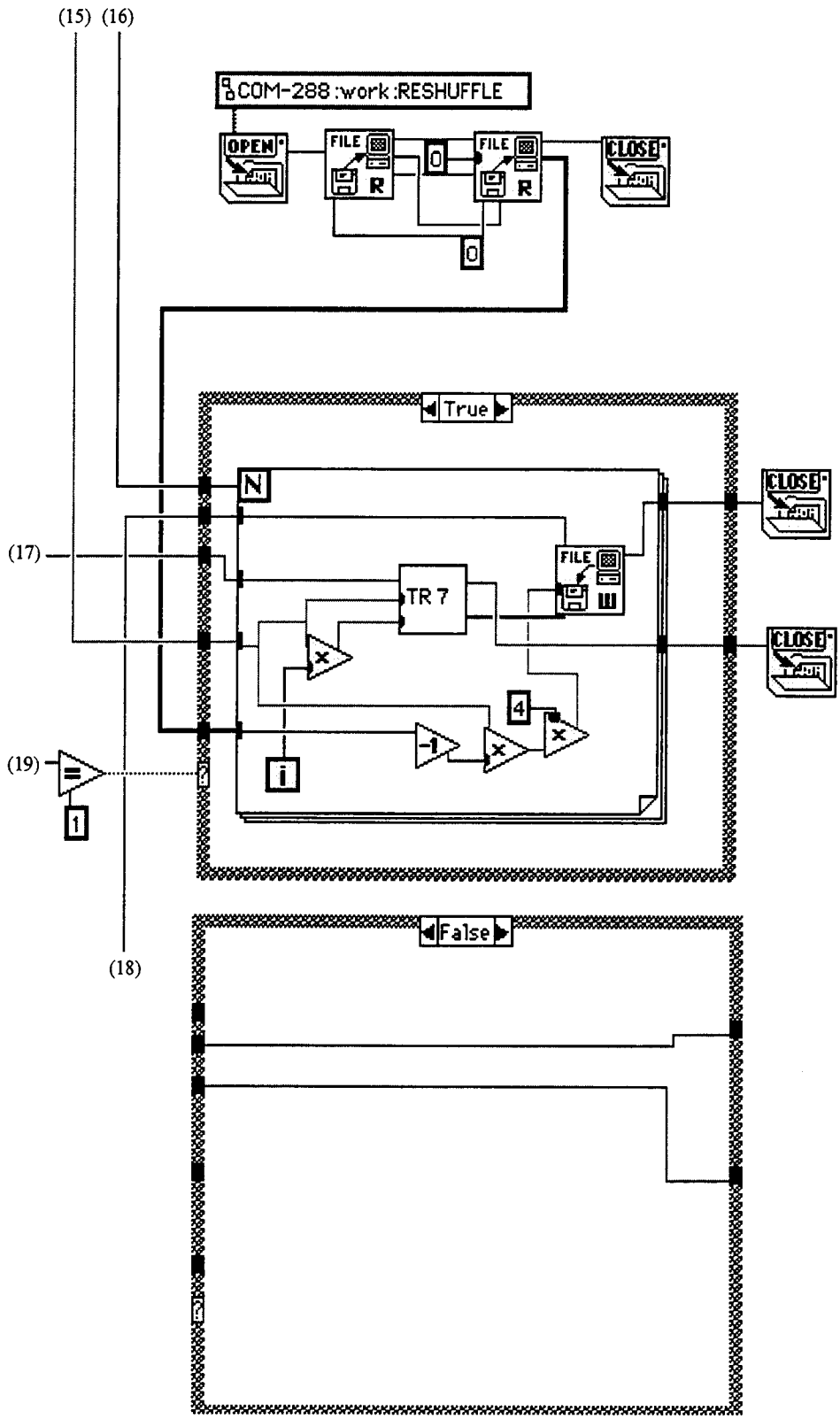


Figure H.3 (continued).

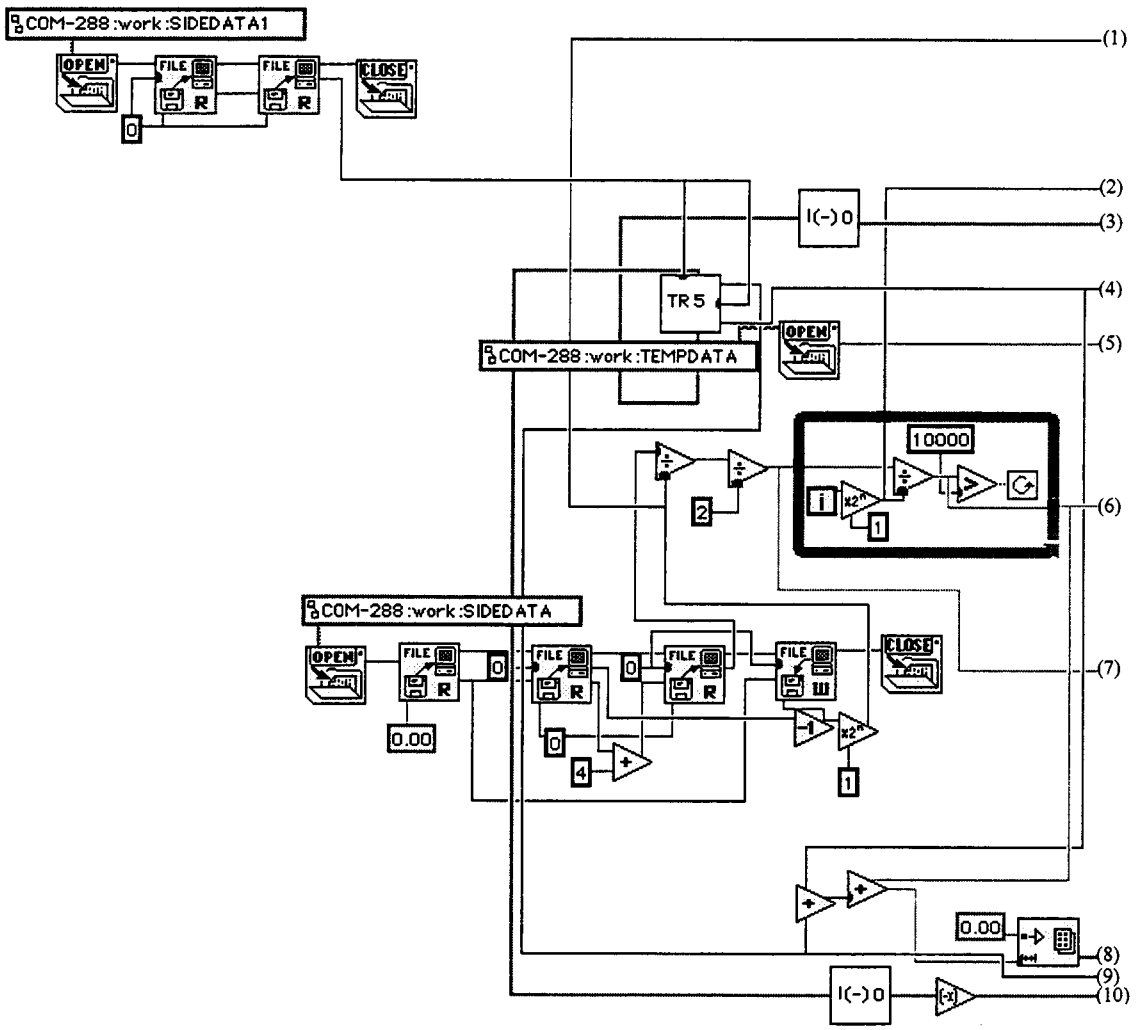


Figure H.4 Diagram of ITREEFILTNEXT*.vi.

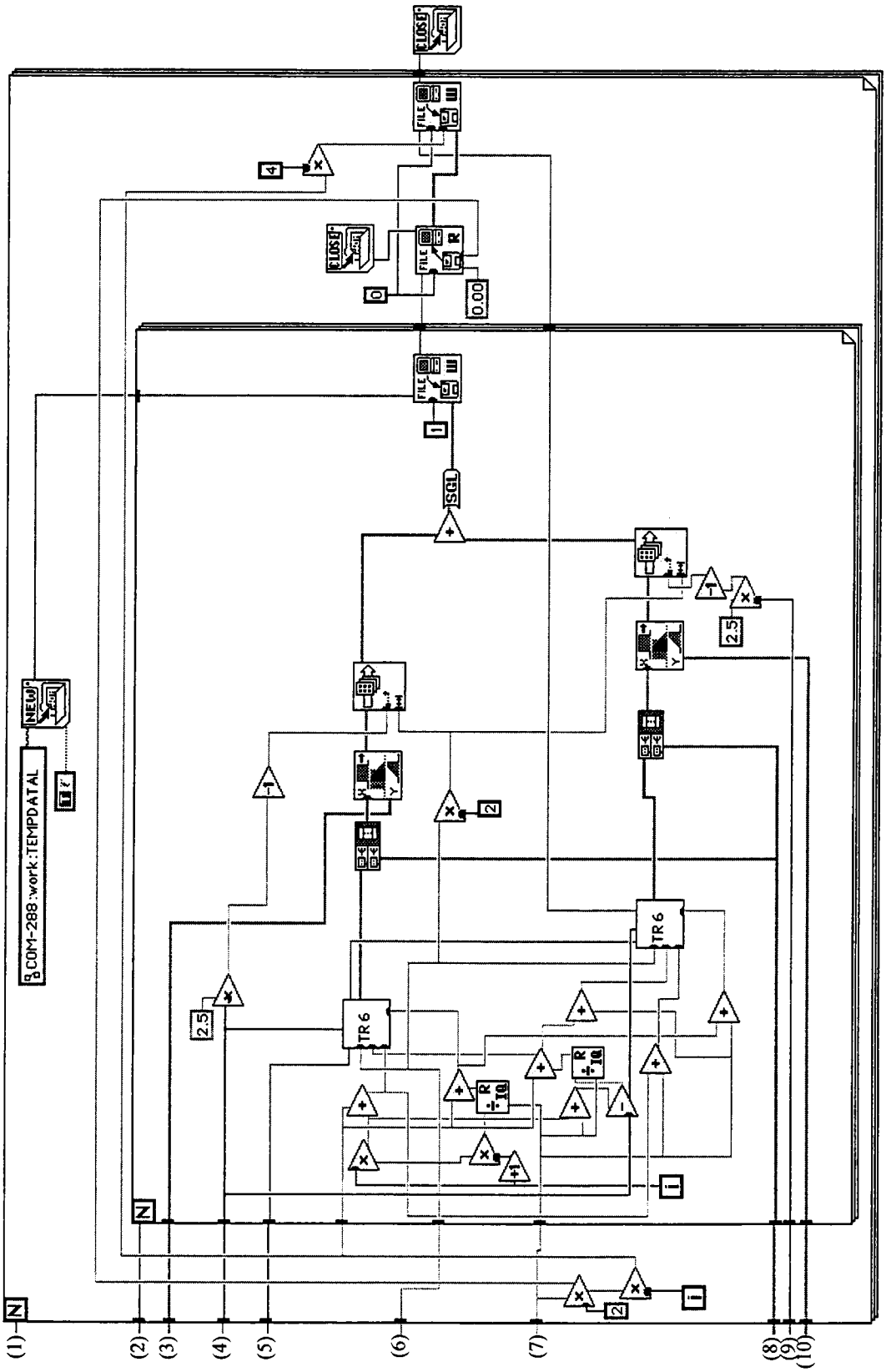


Figure H.4 (continued).

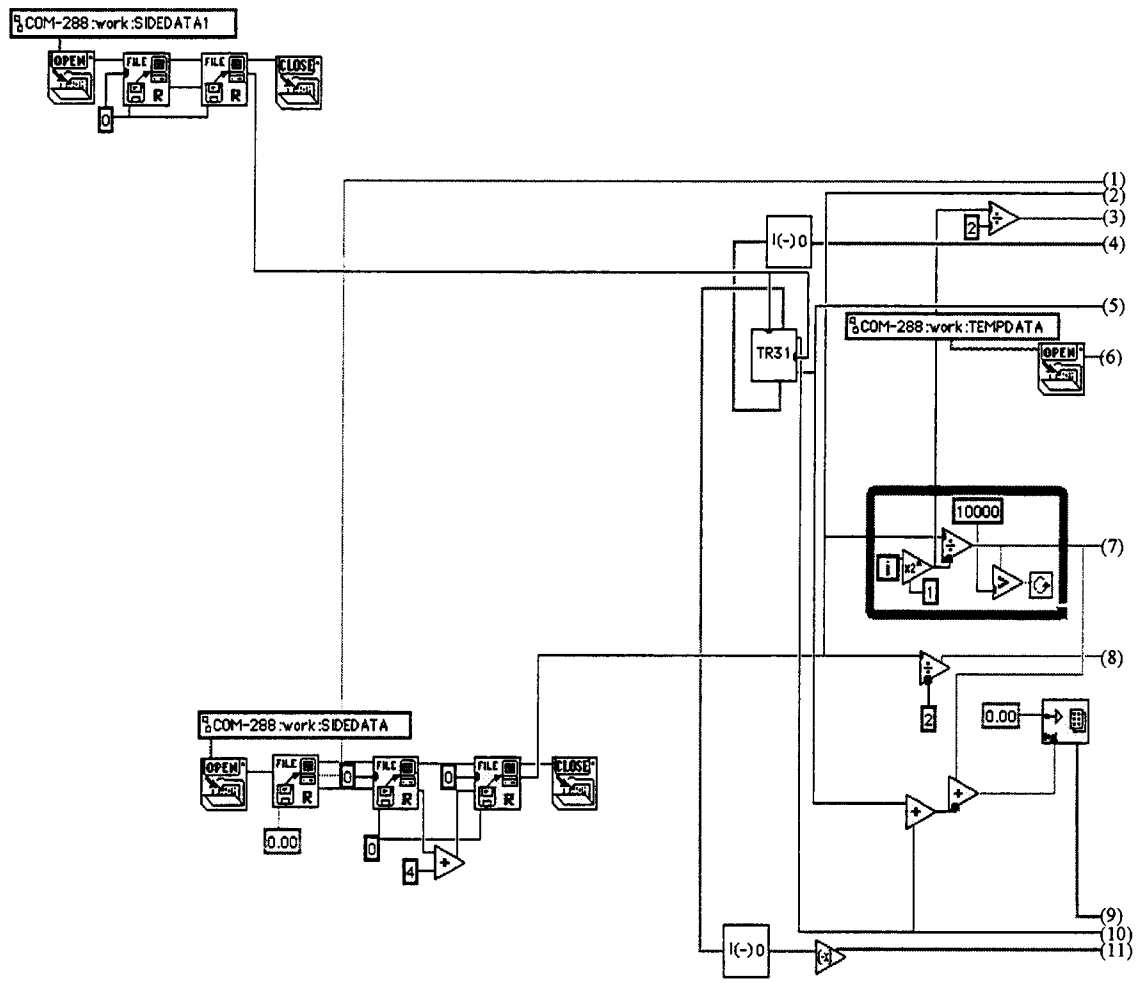


Figure H.5 Diagram of ITREEFILTSTART*.vi.

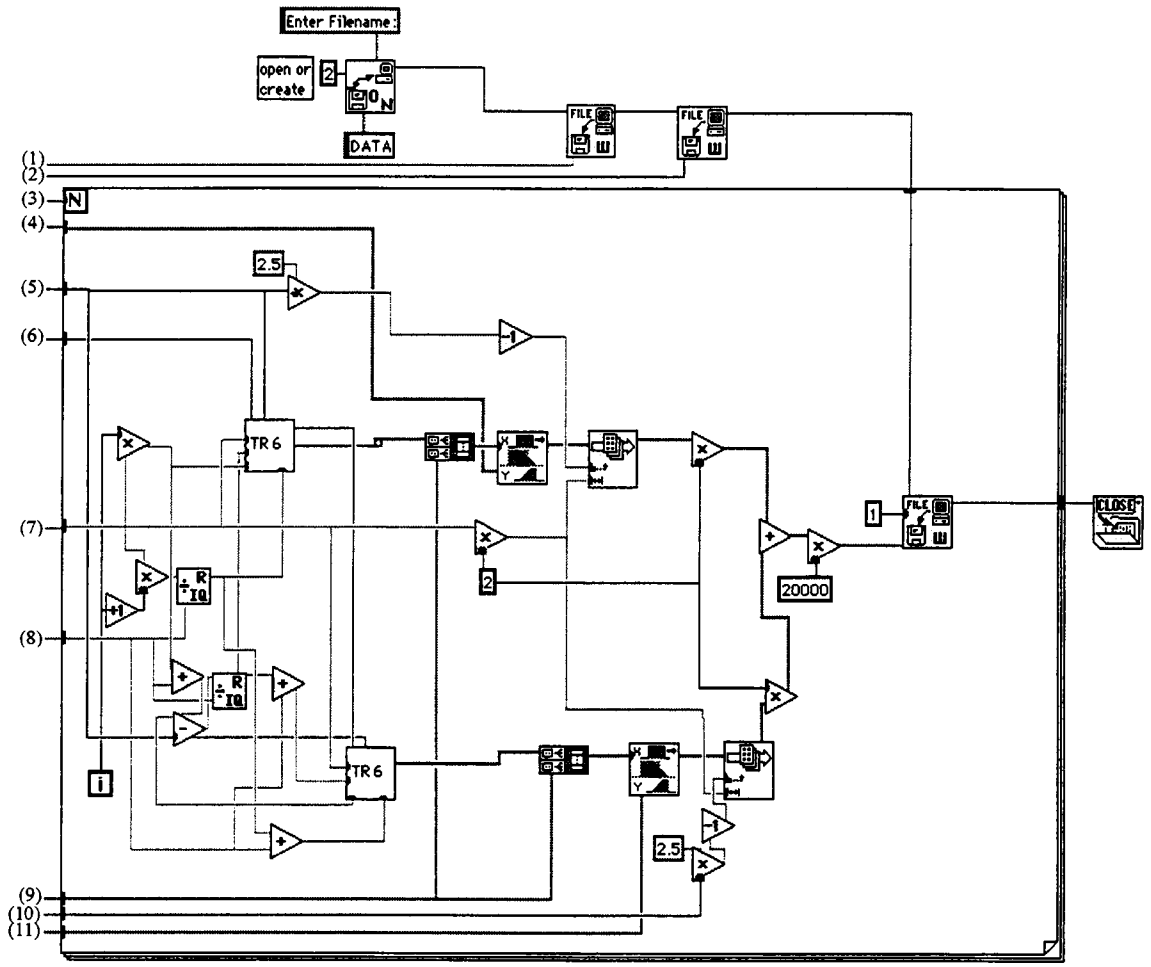


Figure H.5 (continued).

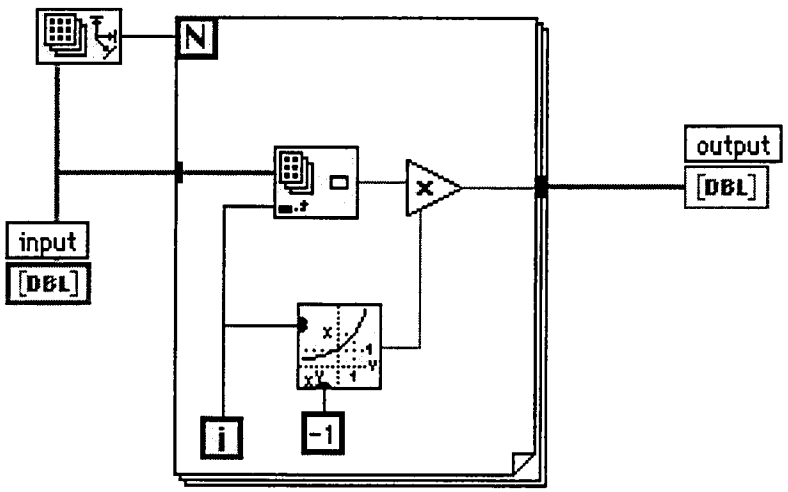


Figure H.6 Diagram of I-O.vi.

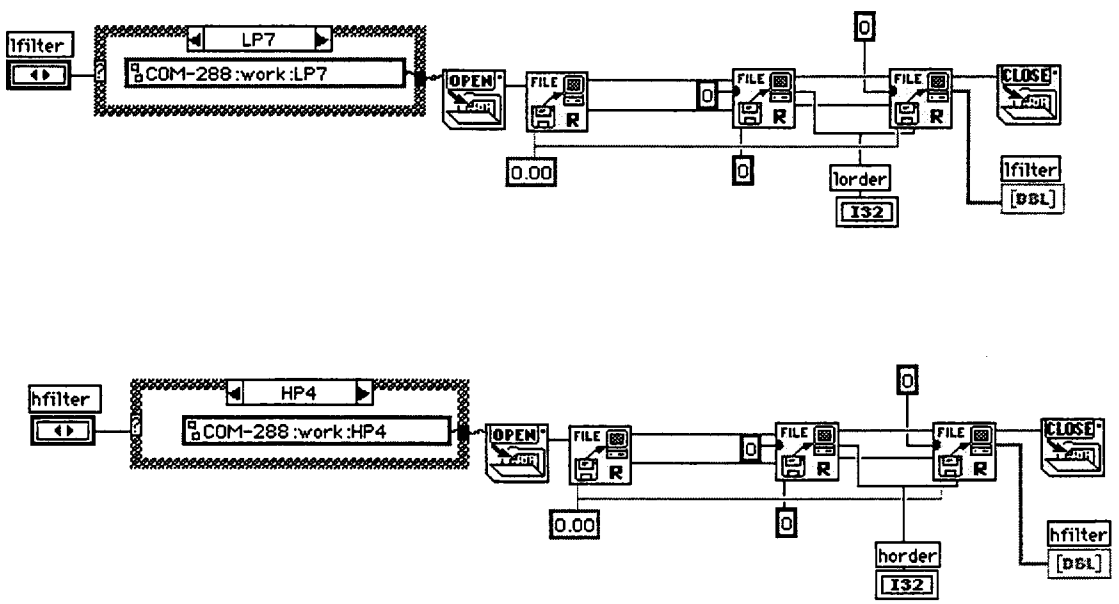


Figure H.7 Diagram of TR31.vi.

VITA

Miljko Bobrek was born in Nova Topola, former Yugoslavia, on June 4, 1954. He graduated from high school in Bosanska Gradiska in 1972 and then entered The University of Banja Luka, Yugoslavia where he received his Bachelor of Science degree in Electrical Engineering in June 1976 and Masters of Science degree in Electrical Engineering in June 1981.

From 1976 to 1989 he was employed at "Rudi Cajavec", Consumer Electronics Company in Banja Luka.

He joined The University of Banja Luka as a research and teaching assistant in 1989, and spent there three years.

He entered his Ph.D. program at The University of Tennessee, Department of Electrical Engineering in January 1993, and graduated with the degree of Doctor of Philosophy in August 1996.