CENTER FOR COMPUTER RESEARCH IN MUSIC AND ACOUSTICS
OCTOBER 1989

Department of Music
Report No. STAN-M-58

A SYSTEM FOR SOUND ANALYSIS/TRANSFORMATION/SYNTHESIS BASED ON A DETERMINISTIC PLUS STOCHASTIC DECOMPOSITION

Xavier Serra

Research sponsored in part by
The System Development Foundation

CCRMA
DEPARTMENT OF MUSIC
Stanford University
Stanford, California 94305

A SYSTEM FOR SOUND ANALYSIS/TRANSFORMATION/SYNTHESIS BASED ON A DETERMINISTIC PLUS STOCHASTIC DECOMPOSITION

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF MUSIC
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

By
Xavier Serra
October 1989

A SYSTEM FOR SOUND ANALYSIS/TRANSFORMATION/SYNTHESIS BASED ON A DETERMINISTIC PLUS STOCHASTIC DECOMPOSITION

Xavier Serra, Ph.D.

Stanford University, 1989

This dissertation introduces a new analysis/synthesis method. It is designed to obtain musically useful intermediate representations for sound transformations. The method's underlying model assumes that a sound is composed of a deterministic component plus a stochastic one. The deterministic component is represented by a series of sinusoids that are described by amplitude and frequency functions. The stochastic component is represented by a series of magnitude-spectrum envelopes that function as a time-varying filter excited by white noise. Together these representations make it possible for a synthesized sound to attain all the perceptual characteristics of the original sound. At the same time the representation is easily modified to create a wide variety of new sounds.

This analysis/synthesis technique is based on the short-time Fourier transform (STFT). From the set of spectra returned by the STFT, the relevant peaks of each spectrum are detected and used as breakpoints in a set of frequency trajectories. The deterministic signal is obtained by synthesizing a sinusoid from each trajectory. Then, in order to obtain the stochastic component, a set of spectra of the deterministic component is computed, and these spectra are subtracted from the spectra of the original sound. The resulting spectral residuals are approximated by a series of envelopes, from which the stochastic signal is generated by performing an inverse-STFT.

The result is a method that is appropriate for the manipulation of sounds. The intermediate representation is very flexible and musically useful in that it offers unlimited possibilities for transformation.

To Eva and Octavi

Acknowledgements

I wish to thank my main advisor Prof. John Chowning for his support throughout the course of this research. Also thanks to Prof. Julius Smith without whom this dissertation could not have even been imagined. His teachings in signal processing, his incredible enthusiasm, and practically infinite supply of ideas made this work possible.

I have to acknowledge Prof. Earl Schubert for always being ready to help with the most diverse problems that I have encountered along the way.

This research would not have been feasible without the particular hardware and software environment on which it was developed. Tovar, the systems programmer at CCRMA, was instrumental in developing and maintaining this environment. In addition, Chris Chafe, Jay Kadis, and Bill Schottstaedt provided very valuable technical support.

I was very fortunate to have the opportunity to collaborate on occasion with Prof.'s Max Mathews, John Pierce, and Bob Schumacher. Over the years I had also very fruitful interactions with the students at CCRMA. I would like to say thanks to Doug Keislar, Richard Karpen, Amnon Wolman, Perry Cook, David Jaffe, and John Strawn.

Contents

Introduction

This dissertation is about sound transformation. The objective is the development of an analysis/synthesis system (Fig. 0.1) that allows the largest possible number of transformations on the analysis data before resynthesis. With such a system a representation is obtained that captures the characteristics of a sound. Then, new sounds are synthesized by performing transformations on the representation and inverting the analysis process.

The system developed in this dissertation is based on modeling the spectral characteristics of a sound. In the frequency domain, a sound is decomposed into deterministic and stochastic components. The sound representation is then obtained by simplifying these two components in order to achieve three objectives. First, we desire a flexible representation, i.e., one easy to transform. Second, the system has to be computationally efficient. Third, the representation is to be judged solely on the quality of the sound it produces. From this point of view, many aspects of the spectral characteristics of the original sound become unimportant, and we wish to exclude them from the representation whenever possible.

The resulting system is an algorithm that is easily implemented in a computer or integrated digital circuit. It is also general in the sense that it can be used for the transformation of a wide variety of sounds.

There are two basic assumptions underlying this research. The first is that any sound can be interchangeably represented in the time domain by a waveform or in the frequency domain by a set of spectra. The second assumption is that most sounds can be decomposed into a deterministic component, which includes the predictable part of the sound (approximated by a sum of time-varying sinusoids), and a stochastic one, which consists of the unpredictable part.

The text is organized as a progression towards the dissertation objective. We go from a general and inflexible sound representation to a less general and more flexible sound

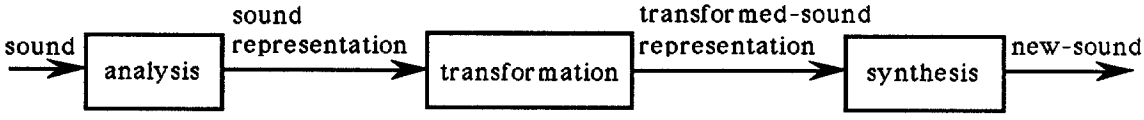

Figure 0.1: Diagram of a general analysis/synthesis system.

representation. Four different systems are presented in chapters 2 to 5, each system building on top of the previous one and carrying the objective of this dissertation a step further.

Chapter 1 summarizes the historical developments relevant to this work. The main ones are: (1) electronic sound-manipulating techniques used in music applications, and (2) speech research oriented towards obtaining alternative representations of the speech signal.

In Chapter 2 the first sound representation is introduced. It is the result of the short-time Fourier transform (STFT), and models a waveform by a set of discrete spectra. From these spectra the original waveform can be recovered, allowing some simple transformations before resynthesis.

Chapter 3 presents a simplification of the short-time Fourier transform that models a sound with a set of sinusoids. These sinusoids are obtained by following the amplitude, frequency and phase of the most prominent peaks over time in the series of spectra returned by the STFT. The sum of all the sinusoids is perceptually equal to the original sound, even though the waveform is not exactly the same. The flexibility of this representation is greater than that of the STFT, thus allowing more transformations of the representations before resynthesis.

A step further is taken in Chapter 4 by assuming that a sound can be decomposed into a deterministic part and a residual. The sound representation is then obtained by restricting the sinusoids of the previous chapter to modelling only the deterministic part of the sound, leaving the rest of the spectral information in the residual component. The sum of the two components is equal to the original sound, and the deterministic component is quite flexible in terms of possible modifications.

The final representation is discussed in Chapter 5. This representation simplifies that of the previous chapter by assuming the residual to be stochastic. Therefore the sound is decomposed into a deterministic and a stochastic component. The representation obtained is flexible and simple, allowing transformations of a variety of sounds. In the absence of modifications, the synthesis from the representation is perceptually very close to the original sound. Compared with the STFT, there is an enormous gain in the flexibility of the representation in exchange for the identity property of the system.

In the appendices some related topics are presented. The first appendix is a description of the hardware and software environment in which this dissertation has been implemented. The next three appendices include variations of the techniques presented in the main chapters that are useful for sound transformation applications. The final appendix is an index of the sound examples that accompany the dissertation.[1]

[1]See Appendix E for instructions on how to obtain a copy of the sound examples.

Chapter 1

Historical Background

1.1 Introduction

The manipulation of pre-recorded sounds has been an important creative resource in electro-acoustic music. This has been accomplished since the early 1950s with tape recorders and other analog devices, and more recently, with the widespread use of digital technology in music, computers have practically replaced the analog media.

The first music compositions using sound manipulations of prerecorded sounds were done in Paris by Pierre Schaeffer in the late 1940s by means of discs, tape recorders, and several kinds of analog devices. Schaeffer used the name "Musique Concrète" to distinguish his new way of creating music, which starts from existing sounds and builds a piece of music out of it, from the traditional music composition, which starts from an abstract idea and only at the end is it materialized into sound. The term Musique Concrète has since then described the music created by transforming prerecorded sounds. Previous to Schaeffer's work some experiments were done in Germany and the U.S.A. with very crude technology that will not be reported here. Comprehensive treatments of Musique Concrète exist elsewhere (Ernst, 1977; Manning, 1985).

With the use of digital computers the capabilities of the tape recorder and the analog devices are extended. Many of the more advanced digital techniques were originally developed in non-musical applications. In particular, many techniques that are currently used in computer music originated in the field of speech research.

This chapter covers the most important historical and current developments that have contributed to the present state of the sound manipulation techniques used in electro-acoustic music and that have influenced this dissertation. The next section summarizes the accomplishments of the traditional Musique Concrète, both with analog and digital means. After that, the speech research which later directly influenced music in the aspect of sound manipulation is described. Then, we summarize the particular musical use of Linear Predictive Coding (LPC), a speech research development, and of analysis-based additive synthesis techniques. Finally, other techniques that have not directly influenced this research, but that are used as analysis-based synthesis techniques, are briefly mentioned.

1.2 Musique Concrète

Supported by the Radiodiffusion Télévision Française (RTF) Pierre Schaeffer used recording techniques to create music out of natural sounds as early as 1948. His equipment was very primitive, consisting of a simple direct disc-cutting lathe and some accompanying analog equipment, but with it he was able to accomplish simple editing and transformations of sounds. The main techniques used were: looping, sound transposition, cutting and juxtaposition of sounds, and filtering. Looping, the repetition of sound over and over, was accomplished by recording a sound into a closed groove of a disk. Sound transposition, which refers to changes in the pitch of a sound, was obtained by varying the speed of the disk player. Cutting sounds was done by operating a volume control inserted between the microphone and the cutter; in this way it was possible to record parts of a sound, for example to leave out the attack of notes. Juxtaposition of sounds was done by simply recording one sound next to the other.

The piece by Pierre Schaeffer and Pierre Henry "Symphonie pour un homme seul" (Schaeffer and Henry, 1949), was done using disc techniques. The *Symphonie* makes use of noises produced by a man, such as breathing, walking, shouting, humming, plus some orchestral and percussion sounds.

1.2.1 Musique Concrète with analog tape

The tape recorder was first used by Schaeffer in 1951. With it appeared new possibilities for sound manipulation, and the process involved in putting together a piece of music was made a bit easier than with discs. The principal techniques developed at the studio of Pierre Schaeffer using analog tape are the following: cutting and splicing, tape loops, change of tape speed, change of direction of tape, tape delay, and combinations of the above. The combinations can be either successions of sounds or simultaneous combinations. On top of the tape manipulations, analog filters and reverberators were also used.

"Cutting and splicing" can be done with discs, but with tape, using the razor blade and the splicing block, the process is more precise.

A tape loop is made by splicing together the two ends of a tape recording so that the recorded passage will repeat continuously during playback.

Change of direction of tape provides a further means of varying the recorded sound. Playing a sound backwards reverses its characteristic envelope and spectral evolution.

Tape delay, also known as tape-recorder feedback, involves the playback and recording of a sound on one machine simultaneously. It adds a reverberation-like effect to the sound.

These tape manipulations and further processing with analog devices such as filters or ring modulators are tools which have been used, and still are used, in electro-acoustic music.

By themselves, or in conjunction with other means of creating sounds, acoustic or electronic, they have proved to be, by listening to their results, very important for music creation. Some of the early compositions involving tape manipulation procedures are: "Vocalise" by Pierre Henry (Henry, 1952), "Williams Mix" by John Cage (Cage, 1952), "Gesang der Jünglinge" by Karlheinz Stockhausen (Stockhausen, 1956), "Thema" by Luciano Berio (Berio, 1958), "Poème Electronique" by Edgard Varèse (Varèse, 1958), "Orient-Occident I" by Iannis Xenakis (Xenakis, 1960), and "Come Out" by Steve Reich (Reich, 1966).

1.2.2 Musique Concrète with computers

With digital computers, equipped with analog to digital (A/D) and digital to analog (D/A) converters and sufficient memory to store the recorded sounds, all the tape techniques can be implemented. However, this was not feasible until the mid seventies, mainly due to memory limitations. Nowadays, elaborate sound editors that allow very precise editing, mixing, and simple processing of waveforms, are available on a wide variety of computers.

Early work manipulating sounds using computers was done at CCRMA by Michael McNabb in his piece "Dreamsong" (McNabb, 1978). The first computer composition from Pierre Schaeffer's center was "Erosphere" (1979) by François Bayle. Other early pieces are: "Arcus" (1978) by York Höller, "Mirages" (1978) and "Songes" (1979) by Jean-Claude Risset (Risset, 1988), and "Mortuos Plango, Vivos Voco" (1980) by Jonathan Harvey (Harvey, 1989).

1.3 Speech Research

A major aim of speech research is to find efficient representations of the speech waveform for transmission and storage. By changing the parameters of the representation, modifications of the speech waveform are accomplished which have proved to be musically useful. In this section the main developments in speech analysis/synthesis with musical potential are presented.

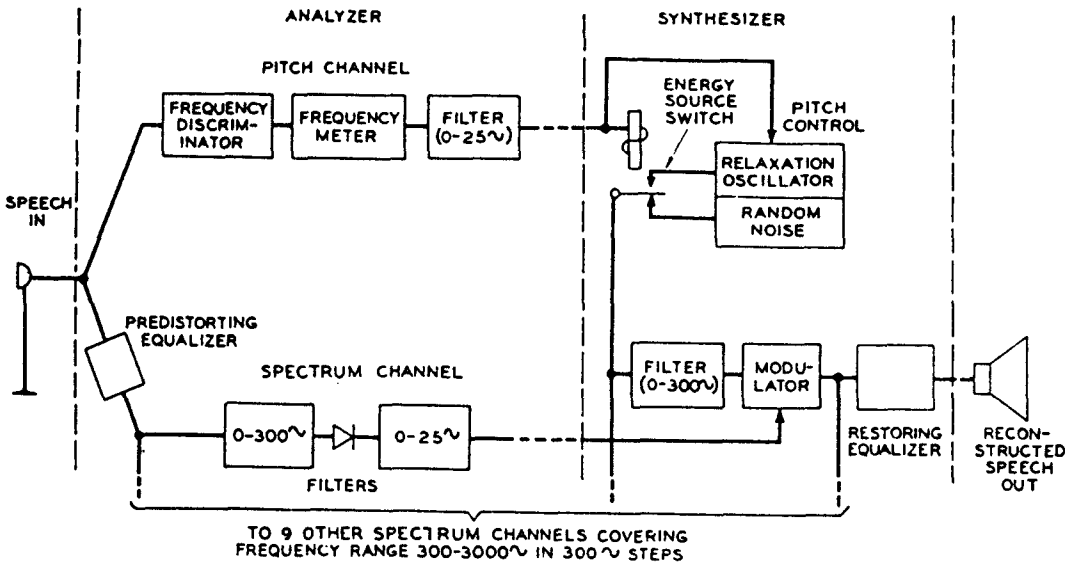

Figure 1.1: Diagram of the system used by Dudley.

1.3.1 The vocoder

The first analysis/synthesis system for speech was invented by Homer Dudley in the 1930s (Dudley, 1939; Schroeder, 1966). Dudley called his system a vocoder "because it operates on the principle of deriving voice codes." The basic idea is to represent the speech waveform by an excitation waveform that represents the sound source, which then goes through a filter that represents the vocal tract. The excitation is a pulse train during vowel sounds and noise during consonants, and the vocal tract response characteristics are represented by a set of "spectrum-analyzing channels."

Figure 1.1 shows the diagram of the system. The output is speech approximately equal in pitch and in spectrum to the original. Dudley's Vocoder has control switches which permit modifications in the operation of the synthesizer, "obtaining interesting effects." Converting voiced into unvoiced speech, by controlling the voiced/unvoiced classification, turns normal speech into a throaty whisper. The pitch of the voice is easily modified by changing the estimate of the pitch period. Any sound can be used as the excitation by using the voice input to control only the bank of filters. This creates the effect of hybridizing the two sounds. If the excitation function is the sound of a locomotive "the demonstrator can make the locomotive puff intelligibly 'We're - start - ing - slow - ly - faster, faster, faster' as the puffs come closer together."

1.3.2 Linear predictive coding (LPC)

In the late 1960s the mathematical technique of linear prediction was applied to the problem of modeling speech behavior (Saito and Itakura, 1966; Atal and Schroeder, 1967), and in 1970 Atal (Atal, 1970) presented the first use of the term linear prediction for speech analysis. The basic idea behind linear predictive analysis is that a speech sample $x(n)$ can be approximated as the linear combination of past speech samples,

$$\hat{x}(n) = -\sum_{k=1}^{p} \alpha_k x(n-k) \tag{1.1}$$

By minimizing the sum of the squared differences (over a finite interval) between the actual samples, $x(n)$, and the linearly predicted ones, $\hat{x}(n)$, a unique set of predictor coefficients, α_k is determined. (The predictor coefficients are the weighting coefficients used in the linear combination.)

The linear prediction formulation is intimately related to the traditional speech model in which speech is modeled as the output of a linear, time-varying filter excited by either quasi-periodic pulses (during voiced speech), or noise (during unvoiced speech). Figure 1.2 shows a diagram of the speech model. The linear prediction method provides a robust, reliable, and accurate method to estimate the parameters that characterize the time-varying system. In the traditional LPC formulation the time-varying filter is an n-pole infinite impulse response (IIR) filter whose poles approximately fall on the formant regions of the analyzed speech sound. Then, there is a pitch detection and a voiced/unvoiced decision algorithm which control the excitation function. With this technique it is possible to manipulate sounds in the same manner as Dudley did with his vocoder.

There have been different formulations of the problem of obtaining the predictor coefficients. The two most relevant ones are called the covariance method (Atal and Hanauer, 1971), and the autocorrelation formulation (Makhoul, 1975; Markel and Gray, 1976). (A more detailed discussion of LPC is given in Appendix C.)

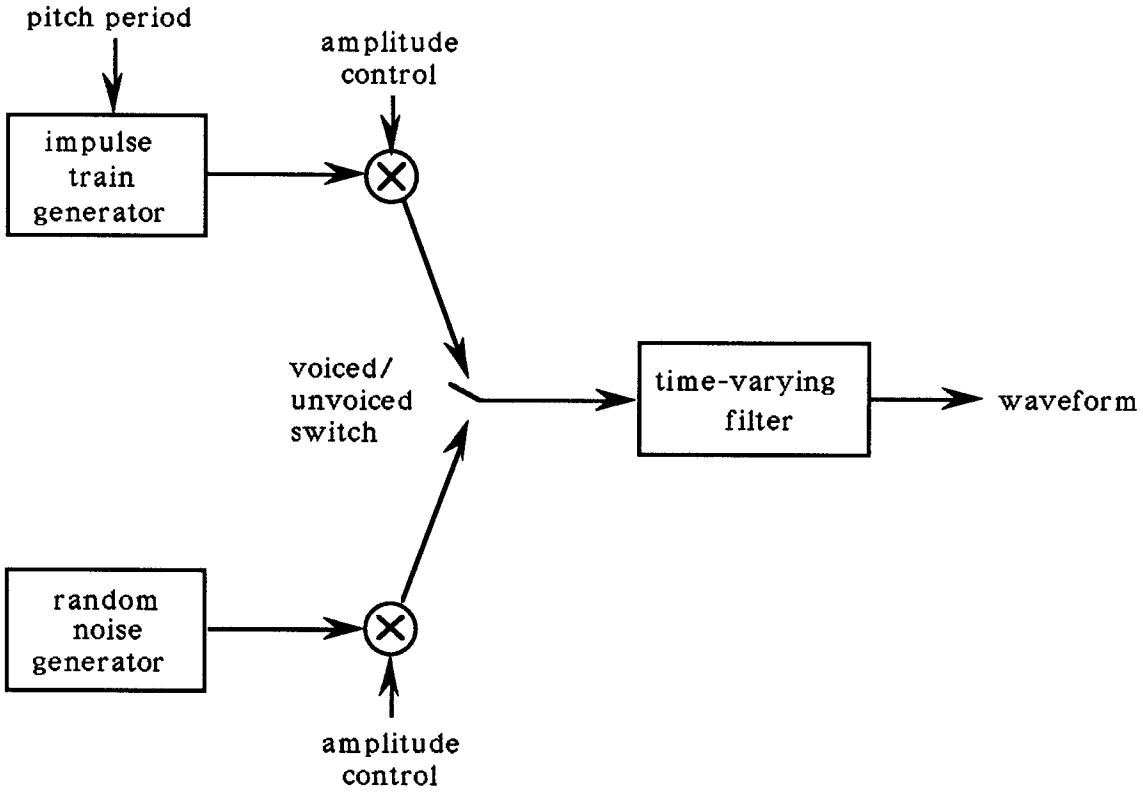

Figure 1.2: Block diagram of simplified model for speech production.

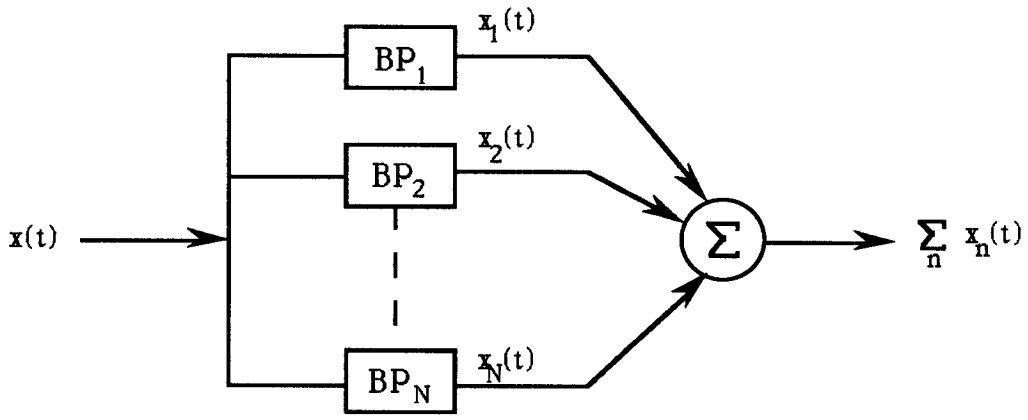

Figure 1.3: Filtering of speech used by Flanagan and Golden.

1.3.3 The phase vocoder

In 1966 Flanagan and Golden (Flanagan and Golden, 1966) proposed another technique for encoding speech, the *phase vocoder*, that represents speech signals by their short-time phase and amplitude spectra. Their starting assumption is that if a speech signal $x(t)$ is passed through a parallel bank of contiguous band-pass filters and then recombined, the signal is not substantially degraded. The operation is illustrated in Fig. 1.3, where $BP_1, \ldots, BP_N$ represent the contiguous filters. The output of the nth filter is $x_n(t)$, and the original signal is approximated as

$$\hat{x}(t) = \sum_{n=1}^{N} x_n(t) \tag{1.2}$$

The impulse response of the nth filter is

$$g_n(t) = h(t)cos(\omega_n t) \tag{1.3}$$

where $h(t)$ is the impulse response of a low-pass filter. The output of the nth filter is the convolution of $x(t)$ with $g_n(t)$,

$$\begin{aligned} x_n(t) &= \int_{-\infty}^{t} x(\lambda) g_n(t-\lambda) d\lambda \\ &= \int_{-\infty}^{t} x(\lambda) h(t-\lambda) \cos[\omega_n (t-\lambda)] d\lambda \\ &= \mathrm{Re}\left[e^{j\omega_n t} \int_{-\infty}^{t} x(\lambda) h(t-\lambda) e^{-j\omega_n \lambda} d\lambda \right] \end{aligned} \tag{1.4}$$

The latter integral is the short-time Fourier transform[1] of the input signal $x(t)$, evaluated at radian frequency ω_n. It is the Fourier transform of that part of $x(t)$ which is "viewed" through the sliding time aperture $h(t)$. If the complex value of the transform is denoted as $X(\omega_n, t)$, its magnitude is the short-time amplitude spectrum $|X(\omega_n, t)|$, and its angle is the short-time phase spectrum $\varphi(\omega_n, t)$. Then

$$x_n(t) = |X(\omega_n, t)| \cos[\omega_n t + \varphi(\omega_n, t)] \tag{1.5}$$

Each $x_n(t)$ may, therefore, be described as the simultaneous amplitude and phase modulation of a carrier $\cos(\omega_n t)$ by the short-time amplitude and phase spectra of $x(t)$, both evaluated at frequency ω_n. However, the phase functions $\varphi(\omega_n, t)$ are generally not bounded, so their derivatives $\dot{\varphi}(\omega_n, t)$, which are more well-behaved, are used instead. Thus, an approximation to $x_n(t)$ is

$$\hat{x}_n(t) = |X(\omega_n, t)| \cos[\omega_n t + \hat{\varphi}(\omega_n, t)] \tag{1.6}$$

where

$$\hat{\varphi}(\omega_n, t) = \int_0^t \dot{\varphi}(\omega_n, \tau) d\tau \tag{1.7}$$

The reconstruction of the original signal is accomplished by summing the outputs of n oscillators modulated in phase and amplitude. Flanagan and Golden showed that the phase vocoder allowed the expansion and compression of the time and frequency scales in a very flexible way.

1.3.4 Recent developments

Research in speech coding is a very active field with constant advances. Here, there is no intention of summarizing the main developments in the field; we will simply mention some of the most recent research that has more or less directly influenced this dissertation.

One important modification on the traditional LPC formulation is what is known as the multipulse LPC (Atal and Remde, 1982). With this formulation the standard pitch-pulse and white-noise excitation is replaced with a sequence of pulses. Their amplitudes and locations are chosen to minimize the perceptual difference between original and synthetic speech signals.

The traditional *phase vocoder* presented by Flanagan has also been extensively modified. A new formulation was introduced by Portnoff (Portnoff, 1976) where most of the computation required is performed by the fast Fourier transform (FFT) algorithm, making the process more efficient while maintaining the properties of the *phase vocoder*. For further

[1]This technique will be discussed in greater detail in the next chapter.

improvements see: Allen 1977; Allen and Rabiner, 1977; Crochiere, 1980; Portnoff, 1980, 1981; Nawab, Quatieri and Lim, 1983; Griffin and Lim, 1984.

More recently, a new speech model has been formulated which is referred to as the Multi-band Excitation Model (Griffin and Lim, 1988). In this model, the short-time spectrum of speech is modeled as the product of an excitation spectrum and a spectral envelope. The spectral envelope is a smoothed version of the speech spectrum and the excitation spectrum is represented by a fundamental frequency, a voiced/unvoiced decision for each harmonic of the fundamental, and the phase of each harmonic declared voiced.

Another recent development is the work done by McAulay and Quatieri on an analysis/synthesis technique based on a sinusoidal model (McAulay and Quatieri, 1986; Quatieri and McAulay, 1986). A very similar method is presented in Chapter 3.

1.4 LPC in Computer Music

Among all the techniques developed for speech purposes, LPC has been used the most in music applications. With LPC a wide variety of speech transformations are possible due to the decomposition of the sound into an excitation function and a time varying filter (Lansky, 1989). Typical modifications are: (1) changing the pitch of the voiced speech while retaining speed and timbre by modifying the pulse train, (2) speed modification by changing the rate at which the filter coefficients are updated, and (3) creating either whisper or tonal voice out of normal speech by using as the excitation either the noise or the pulse train alone.

Charles Dodge was one of the first composers to use LPC in music composition (Dodge, 1975; Dodge, 1985; Dodge and Jerse, 1985; Dodge, 1989). His piece "Speech Songs" done between 1972 and 1973 (Dodge, 1975) can be considered one of the first computer music works to be based directly on the computer analysis of recorded sound. Other early pieces by Dodge using LPC are: "In Celebration," and "The Story of Our Lives" (Dodge, 1975).

An interesting musical usage of LPC is the possibility of hybridizing two sounds (already done by Dudley in the 1930s), commonly referred as cross-synthesis (Petersen, 1976; Moorer, 1979). Instead of the noise or periodic pulse, one sound is used as the excitation function for the LPC filtering process. One of the early compositions using this capability of LPC is Tracy Petersen's "Voices" (Petersen, 1975). An alternative technique for cross-synthesis is presented in appendix B.

J. A. Moorer made an important contribution by discussing some of the particular problems that the use of LPC presents in music, and offered solutions (Moorer, 1977; Moorer, 1979). His pieces "Perfect days" (1977) and "Lions are growing," composed in 1978 (Moorer, 1983) are excellent examples of the musical possibilities of LPC.

LPC can be used for the analysis/synthesis of sounds other than speech, even though the synthesized sounds may be quite different from the original. This is most successful when the sounds to be analyzed have a clear formant structure (Lansky and Steiglitz, 1981).

1.5 Analysis-Based Additive Synthesis

One of the first computer-based analysis-synthesis systems used for musical applications was fashioned by David Luce (Luce, 1963). The use of such a system gave insight into the behavior of musical instruments and its possible perceptual implications. The object of Luce's method was to determine the amplitudes and frequencies of each of the harmonics of a tone as a function of time. The synthesis had the purpose of checking the accuracy of the analysis data. Some improvements on Luce's method were made by Morris David Freedman (Freedman 1965; Freedman 1967; Freedman 1968). Similar work was carried out by James Beauchamp (Beauchamp, 1969) and by Jean-Claude Risset and Max Mathews (Risset and Mathews, 1969).

The current state of the art in additive analysis/synthesis was defined by J. A. Moorer and John Grey in a landmark series of investigations (Moorer, 1975; Grey, 1975; Grey and Moorer, 1977; Grey and Gordon, 1978). They used what they called the heterodyne filter as a means of doing analysis-based additive synthesis.

More recently, the *phase vocoder* has become the standard technique for analyzing musical sounds and performing additive synthesis from the analysis data (Moorer, 1978; Dolson, 1983a, 1983b, 1984, 1985, 1986, 1988; Gordon and Strawn, 1985). It has also been extensively used in music compositions, especially at the studios of UCSD (San Diego) and IRCAM (Paris). With the *phase vocoder*, temporal and spectral features of the sound can be manipulated independently (Dolson, 1989). Possible sound modifications are: (1) time scaling—changing the duration of a sound without altering its pitch, (2) pitch transposition—changing pitch without affecting duration, and (3) time-varying filtering. For example, in "Transfigured Wind" Roger Reynolds uses the *phase vocoder* to stretch flute sounds. However, when the analyzed sounds are harmonic, a wider variety of transformations is possible. For example in "Solera" (Chafe, 1983), Chris Chafe uses the analysis of a clarinet tone as the basis for the whole piece.

1.6 Other Analysis-Based Synthesis Techniques

Apart from LPC and additive synthesis there are other analysis-based synthesis techniques that have been used in music applications and deserve to be mentioned, even though they have not influenced the present research.

Xavier Rodet developed the formant-wave-function synthesis, better known as the CHANT program (Rodet, 1984; Rodet, Potard and Barrière, 1984; Bennett and Rodet, 1989). CHANT is what is known as a synthesis-by-rule technique. Originally designed for the synthesis of the singing voice, this method is based on an acoustical model of the vocal tract and of the singing voice, and directly calculates the amplitude of the waveform of a signal as a function of time. Conceptually, it can be understood as a kind of LPC, where the voice is represented by an excitation going through a set of filters that model the formant evolution.

Another technique originally designed for speech is the VOSIM sound synthesis (Kaegi and Tempelaars, 1978; Janssen and Kaegi, 1986). This synthesis method is based on $\sin^2$ pulses, where the amplitude of the successive pulses decreases uniformly.

A recent development is the use of the wavelet transform for analysis and synthesis of sounds (Kronland-Martinet, 1988). This technique analyzes signals in terms of *wavelets*—functions limited in both the time and frequency domains.

Physical modeling, where a sound is created by emulating the physical behavior of musical instruments, can also be considered a kind of analysis-based synthesis technique (Smith, 1983, 1987; Adrien, Causse and Rodet, 1987; Adrien, Causse and Ducasse, 1988).

Finally, the work by Serra, Rubine and Dannenberg (Serra, Rubine and Dannenberg, 1988) should also be mentioned. Their analysis/synthesis technique is based on the interpolation of spectra over time.

Chapter 2

The Short-Time Fourier Transform

2.1 Introduction

The auditory system functions as a spectrum analyzer, detecting the frequencies present in the incoming sound at every moment in time. In the inner ear, the cochlea can be understood as a set of band-pass filters, each filter passing only frequencies in a very narrow range. This is what a spectrum analyzer does. Spectral representations are so widely used in sound applications because they mimic the behavior of the ear. These representations, also known as Fourier representations, are well studied (Rabiner and Gold, 1975; Oppenheim and Schafer, 1975) and are used in many scientific and technical applications. In particular, in computer music they have been extensively used for the analysis and synthesis of sounds (Moorer, 1977; Cann, 1979; Dodge and Jerse, 1985).

In this chapter we present a particular Fourier-based analysis and synthesis system, called the short-time Fourier transform, STFT, (Allen, 1977; Allen and Rabiner, 1977). This is a very general technique useful in the study of time-varying signals, such as musical sounds, that can be used as the basis for more specialized techniques. In the following chapters the STFT is used as the basis for several analysis/synthesis systems.

The spectral analysis performed by the auditory system differs in some respects from the standard Fourier analysis methods, as well as from the STFT. It is important to know the divergences so that they can be corrected, or at least taken into account when interpreting the results of the analysis. The most important difference is that the auditory system obtains a log-scale spectrum whereas traditional Fourier analysis computes the spectrum with a linear scale. In addition, other characteristics of the auditory system such as masking both in time and in frequency domains,[1] and perception of amplitude in relation with frequency (Plomp, 1966; Yost and Nielsen, 1977; Roederer, 1979), are not considered in the standard spectrum analysis implementations.

[1]Time domain masking refers to the masking of one event by another one that precedes or follows it in time. Frequency domain masking refers to the masking of one frequency component by another one.

The purpose of this dissertation is a musical one. Perceptual considerations are taken into account whenever they bring improvements to the musical results and do not excessively complicate the technique. This is a trade-off which is not always easy to make and has to be weighted in every situation.

This chapter presents the STFT as it will be used by the analysis/synthesis techniques developed in the following chapters. Emphasis is given to aspects which are relevant for those techniques. Before introducing the STFT, we present the Fourier transform, the basis of all spectral representations. After the Fourier transform, we introduce the model on which the STFT is based. Then, an analysis/synthesis system based on the STFT is presented, first the analysis part and then the synthesis part, followed by a summary of the system. The chapter ends with an example that demonstrates the use of the STFT and a section with conclusions.

2.2 The Fourier Transform

The Fourier transform is a method that allows the conversion of a time domain waveform into its frequency spectrum, or, in other words, the decomposition of a waveform into a number sinusoidal components.

The most common definition of the Fourier transform is

$$X(\omega) \triangleq \int_{-\infty}^{+\infty} x(t) e^{-j\omega t} dt \tag{2.1}$$

where t is the continuous time index in seconds and ω is the continuous frequency index expressed in radians per second.

Given a continuous waveform $x(t)$, the Fourier transform returns $X(\omega)$, a complex number of the form $(a + ib)$ for every value of ω. The magnitude of the complex number, $\sqrt{a^2 + b^2}$, specifies the amplitude of a sinusoidal component of $x(t)$ with frequency ω, and the angle, $\tan^{-1}(b/a)$, specifies the phase of that same sinusoidal component. It is customary to interpret $X(\omega)$ as the whole frequency spectrum, that is, the values for all ω.

This transform is widely used and well understood (see, for example, Bracewell, 1978). For the purpose of this thesis it is not necessary to go into its properties. It is sufficient to say that $X(\omega)$ is a periodic function of ω with period 2π and that it is possible to recover the original function $x(t)$ by means of the inverse Fourier transform,

$$x(t) \triangleq \frac{1}{2\pi} \int_{-\infty}^{+\infty} X(\omega) e^{j\omega t} d\omega \tag{2.2}$$

That is, given the spectrum $X(\omega)$, the inverse Fourier transform returns the waveform $x(t)$ from which it was obtained. Thus, the Fourier transform is an identity system.

In the discrete world of computers, sounds exist as sampled waveforms and the continuous Fourier transform becomes the discrete Fourier transform (DFT). If $x(n)$ is a signal N samples long, then its DFT is

$$X(k) \triangleq \sum_{n=-N/2}^{N/2-1} x(n)e^{-j\omega_k n}, \qquad \omega_k = 2\pi k/N, \quad \text{N even}, \quad k = 0, 1, \ldots, N-1 \tag{2.3}$$

where ω is the discrete radian frequency, n is the discrete time index in samples, and k the discrete frequency index in bins (frequency domain samples are referred as bins). $X(k)$ is the discrete spectrum. The relation between radian frequency and frequency in Hertz is given by

$$f = f_s\omega/2\pi \tag{2.4}$$

where f the frequency in Hz, f_s is the sampling rate and ω the radian frequency.

Since the frequency index k is discrete, the DFT assumes that $x(n)$ can be represented by a finite number of sinusoids, therefore the signal $x(n)$ is bandlimited in frequency. The frequencies of the sinusoids are equally spaced between 0 Hz and the sampling rate f_s, or in radians between 0 and 2π. Thus, the DFT takes a sequence of length N (the time signal) and produces another sequence of length N (the frequency spectrum).

The corresponding inverse-DFT is

$$x(n) \triangleq \frac{1}{N} \sum_{k=-N/2}^{N/2-1} X(k)e^{j\omega_k n} \tag{2.5}$$

That is, given the discrete spectrum $X(k)$, the inverse-DFT returns the discrete waveform $x(n)$ from which it was obtained.

The prevalence of the DFT in so many different applications is due to the existence of a very fast algorithm for its computation called the fast Fourier transform (FFT). The traditional implementation of the FFT requires the length of the signal, N, to be a power of 2. By making this restriction the computation time is reduced from one proportional to N^2 for the DFT to one proportional to $N \log N$ for the FFT.

2.3 The Short-Time Fourier Model

Every useful analysis/synthesis system is based on an underlying mathematical model. The STFT is no exception. The model underlying the STFT technique presented in this chapter can be formulated as

$$s(n) = \sum_{l=0}^{L-1} \frac{1}{2\pi} \left(\int_{-\pi}^{\pi} X_l(\omega) e^{j\omega(n-lH)} d\omega \right) \tag{2.6}$$

were $X_l(\omega)$ is the continuous spectrum of a signal $x(n)$ at frame l. That is, the signal $s(n)$ is modeled as the sum of a series of inverse Fourier transforms.

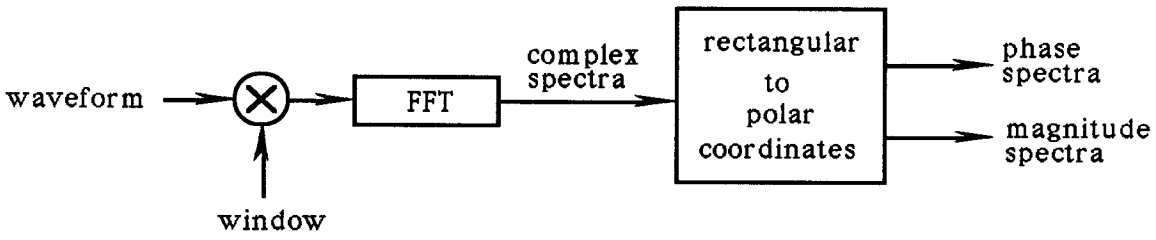

Figure 2.1: General diagram of the analysis part of the STFT.

2.4 Short-Time Fourier Transform

In practice, musical sounds are non-periodic and time-varying waveforms, characteristics for which the Fourier transform is not appropriate. The alternative is to use the short-time Fourier transform. A diagram of the transform is shown in figure 2.1 and figure 2.2 includes a graphic example. In the discrete case it may be defined as

$$X_l(k) \triangleq \sum_{n=0}^{N-1} w(n)x(n+lH)e^{-j\omega_k n}, \qquad l = 0, 1, \ldots \tag{2.7}$$

where $w(n)$ is a real "window" that determines the portion of the input signal $x(n)$ that receives emphasis at a particular frame l. H is the *hop-size*, or time advance, of the window.

The spectrum $X_l(k)$ is a function of two variables, the frame number l and the discrete frequency index k. Depending on which one is considered fixed, the spectrum $X_l(k)$ has two different interpretations (Allen and Rabiner, 1977). For the purpose of this thesis only the interpretation with l fixed, called overlap-add decomposition, will be considered. In this case, $X_l(k)$ is the normal discrete Fourier transform, the spectrum, of the sequence $w(n)x(n+lH)$, $0 \leq n < N-1$. The STFT computes a spectrum at every frame l, advancing with a *hop-size* of H so as to "slide" the window $w(n)$ along the sequence $x(n)$.

The other interpretation of $X_l(k)$ is as a function of l, for a fixed frequency index k. In this filter-bank or sinusoidal interpretation the values for each frequency bin k (a complex number at every l) describe a sinusoid. (In the next chapter a system is developed which uses a sinusoidal interpretation, but since it has major differences with the filter-bank interpretation it may be confusing to relate them.)

In the overlap-add interpretation the process of windowing not only selects the portion of signal to be processed but by tapering the ends of the analyzed data it also converts the frequency spectrum into a smooth function (except when a rectangular window is used). This allows the analysis of non-periodic sounds, that is sounds which have partials at non-integer multiples of the fundamental frequency. In a smooth spectrum the peaks need not be centered on the discrete samples and every possible peak can theoretically be detected by doing the appropriate interpolation, as will be seen later.

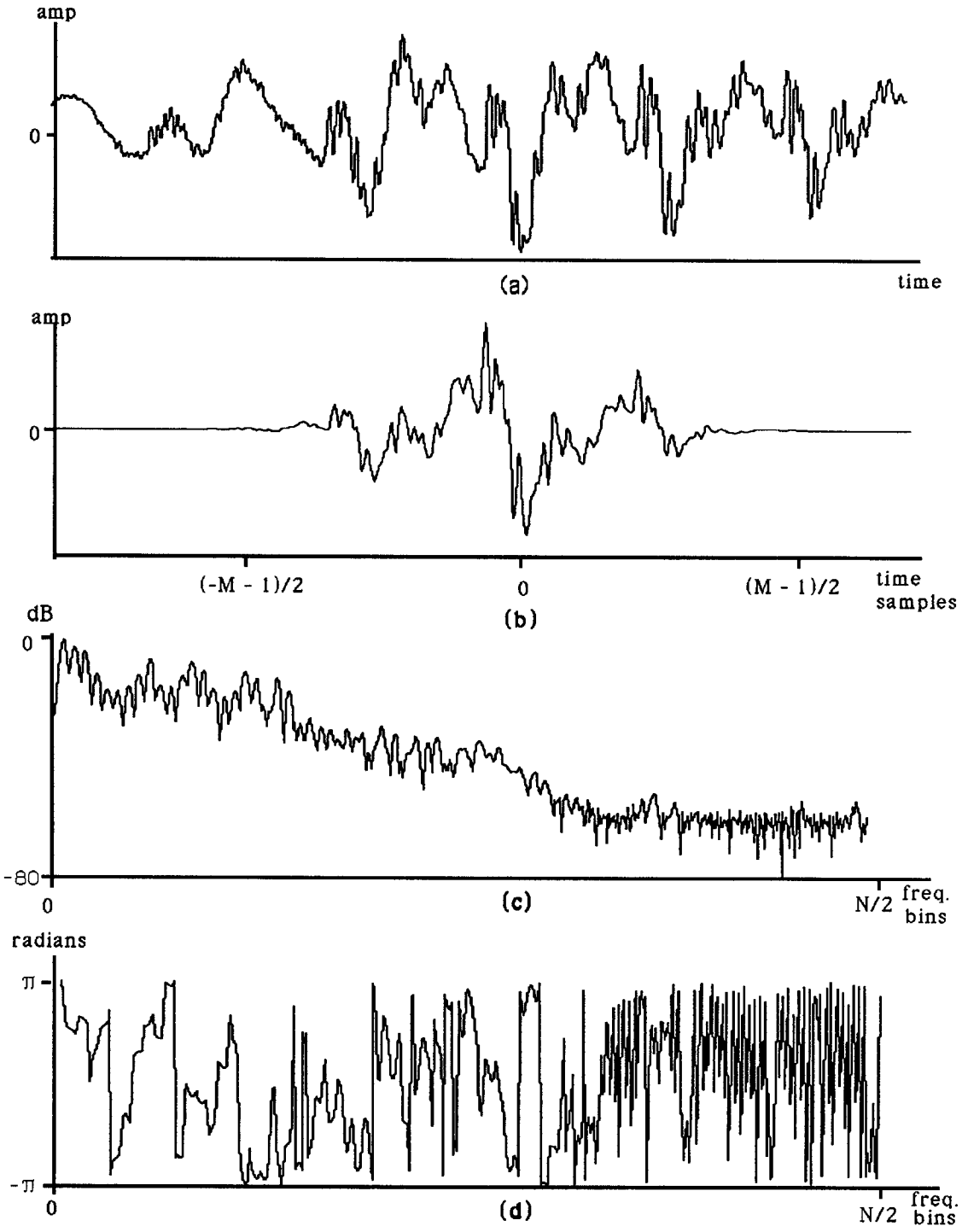

Figure 2.2: STFT example: (a) speech waveform, (b) windowed waveform, (c) magnitude spectrum from the windowed waveform, (d) corresponding phase spectrum.

The output of the STFT is a series of spectra, one for every frame l of the input waveform. Each spectrum $X_l(k)$ is a complex valued function, but a more useful representation is in terms of magnitude and phase, obtained by

$$\begin{aligned} A_l(k) &\triangleq |X_l(k)| \triangleq \sqrt{a^2(k) + b^2(k)} \\ \Theta_l(k) &\triangleq \angle X_l(k) \triangleq \tan^{-1}[b(k)/a(k)] \qquad \text{(radians)} \end{aligned} \tag{2.8}$$

where $|X_l(k)|$ is the magnitude, $\angle X_l(k)$ is the phase, and $a(k)$ and $b(k)$ are the real and imaginary parts of the complex value returned by the Fourier transform,

$$\begin{aligned} a(k) &\triangleq \text{Re}\{X_l(k)\} \\ b(k) &\triangleq \text{Im}\{X_l(k)\} \end{aligned} \tag{2.9}$$

The most common version of the STFT used in computer music is the phase-vocoder (Flanagan, 1966; Dolson, 1986). This particular implementation of the STFT goes one step further by converting the phase values into instantaneous frequencies. If the phase at frequency bin k and frame l is given by $\theta_l(k)$, then the instantaneous frequency at the same bin is

$$\hat{f}_l(k) \triangleq \frac{\theta_l(k) - \theta_{l-1}(k)}{2\pi HT} \qquad \text{(Hz)} \tag{2.10}$$

where H is the hop-size of the window, T $(= 1/f_s)$ the sample period (in seconds), and $\hat{f}$ is the estimated instantaneous frequency. The phase is discarded and redefined as the integral of the instantaneous frequency when needed,

$$\hat{\theta}_m(k) \triangleq \hat{\theta}_{m-1}(k) + 2\pi T \hat{f}_m(k) \tag{2.11}$$

where $\hat{f}_m(k)$ is the linear interpolated frequency from $\hat{f}_{l-1}(k)$ to $\hat{f}_l(k)$.

The phase-vocoder is a more flexible representation than the standard STFT. It provides for time scaling and frequency transposition of sounds. However, only the standard STFT implementation is discussed in this chapter since it is the one used as the intermediate step for the techniques presented in the following chapters.

There are several issues in the calculation of the STFT that require particular attention: (1) choice of the analysis window, (2) computation of the DFT, and (3) hop size of the window. They are discussed below in that order.

2.4.1 Analysis window

The first step in computing the STFT is the windowing of the waveform. The choice of the analysis window is important. It determines the trade-off of time versus frequency resolution which affects the smoothness of the spectrum and the detectability of different sinusoidal components. The most commonly used windows are called Rectangular, Hamming, Hanning, Kaiser, Blackman, and Blackman-Harris. Harris (Harris, 1978) gives a good discussion of these windows and many others.

To understand the effect of the window, let us look at what happens to a sinusoid when its Fourier transform is computed. A complex sinusoid of the form

$$x(n) = Ae^{j\omega_x n} \tag{2.12}$$

where A is the amplitude and ω_x is the radian frequency, when multiplied by a length M window transforms to

$$\begin{aligned} X_w(\omega) &= \sum_{n=-\infty}^{\infty} x(n)w(n)e^{-j\omega n} \\ &= A\sum_{n=-M/2}^{(M/2)-1} w(n)e^{-j(\omega-\omega_x)n} \\ &= AW(\omega-\omega_x) \end{aligned} \tag{2.13}$$

where $W(\omega)$ is the transform of the analysis window. Thus, the transform of a windowed sinusoid, whether isolated or part of a complex tone, is the transform of the window scaled by the amplitude of the sinusoid and centered at the sinusoid's frequency (Fig. 2.3).

All the standard windows are real and symmetric and have a frequency spectrum with a sinc-like shape[2] (Fig. 2.3). For the purpose of this thesis, and in general for any sound analysis/synthesis application, the choice is mainly determined by two of the spectrum's characteristics: (1) the width of the main lobe, defined as the number of bins (DFT-sample points in a spectrum computed by a DFT of the size of the window length) between the two zero crossings,[3] and (2) the highest side-lobe level, which measures how many dB down the highest side lobe is from the main lobe. Ideally, we want a narrow main lobe (i.e., good resolution) and a very low side-lobe level (i.e., no cross-talk between DFT channels). The choice of window determines this trade-off. For example, the rectangular window has the narrowest main lobe, 2 bins ($= 2f_s/M$Hz), but the first side-lobe is very high, -13dB relative to the main-lobe peak. The Hamming window has a wider main lobe, 4 bins, and

[2] A *sinc* function is defined as $\sin(x)/x$. Most of the standard windows can be expressed as a sum of these functions.

[3] When the analysis-window size is equal to the DTF-size, a bin corresponds to f_s/MHz, where f_s is the sampling rate and M the window length.

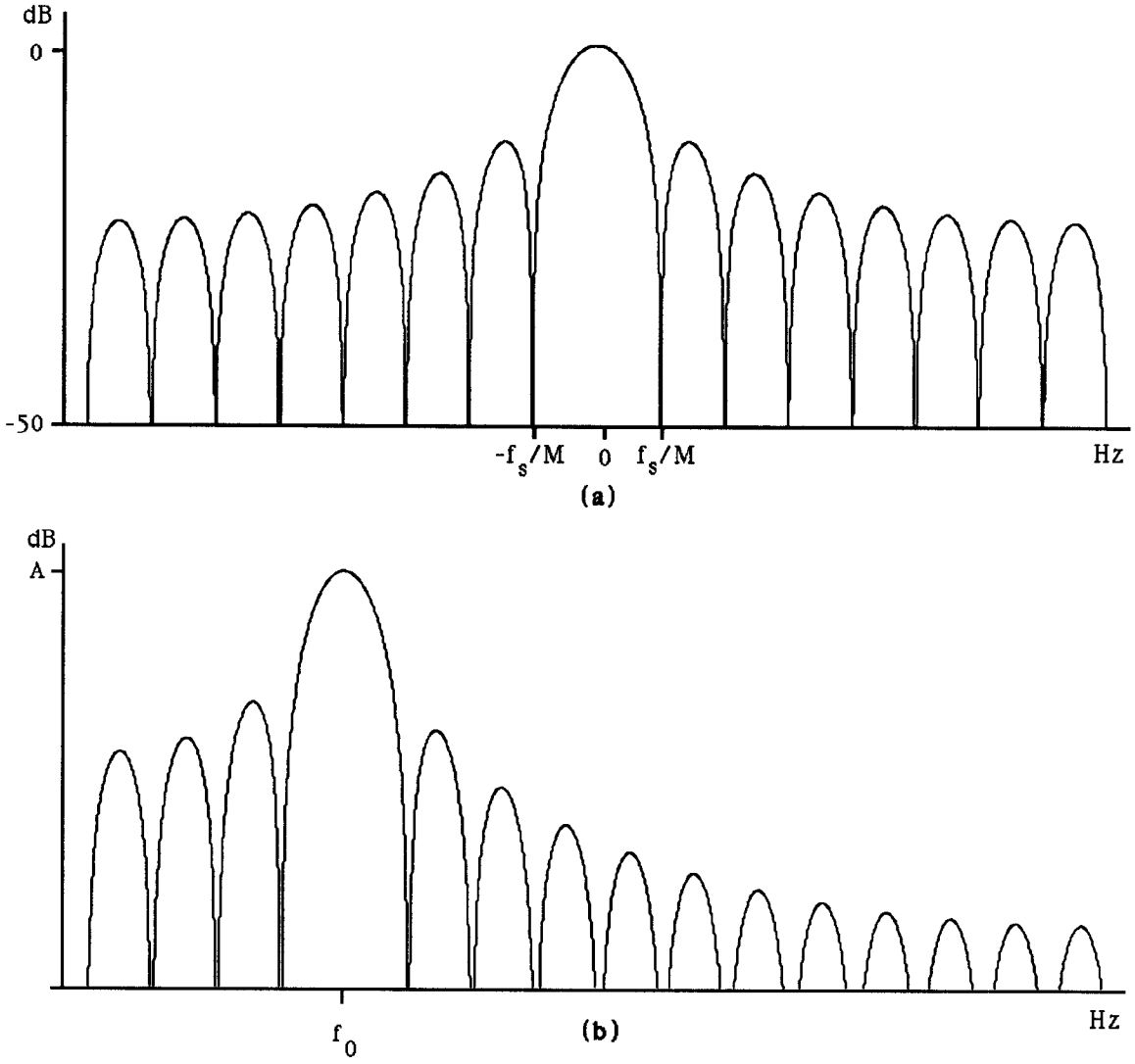

Figure 2.3: (a) Magnitude spectrum of a rectangular window, f_s = sampling rate, M = analysis-window size, (b) magnitude spectrum of a sinusoid with frequency f_0 and amplitude A, windowed using a rectangular window.

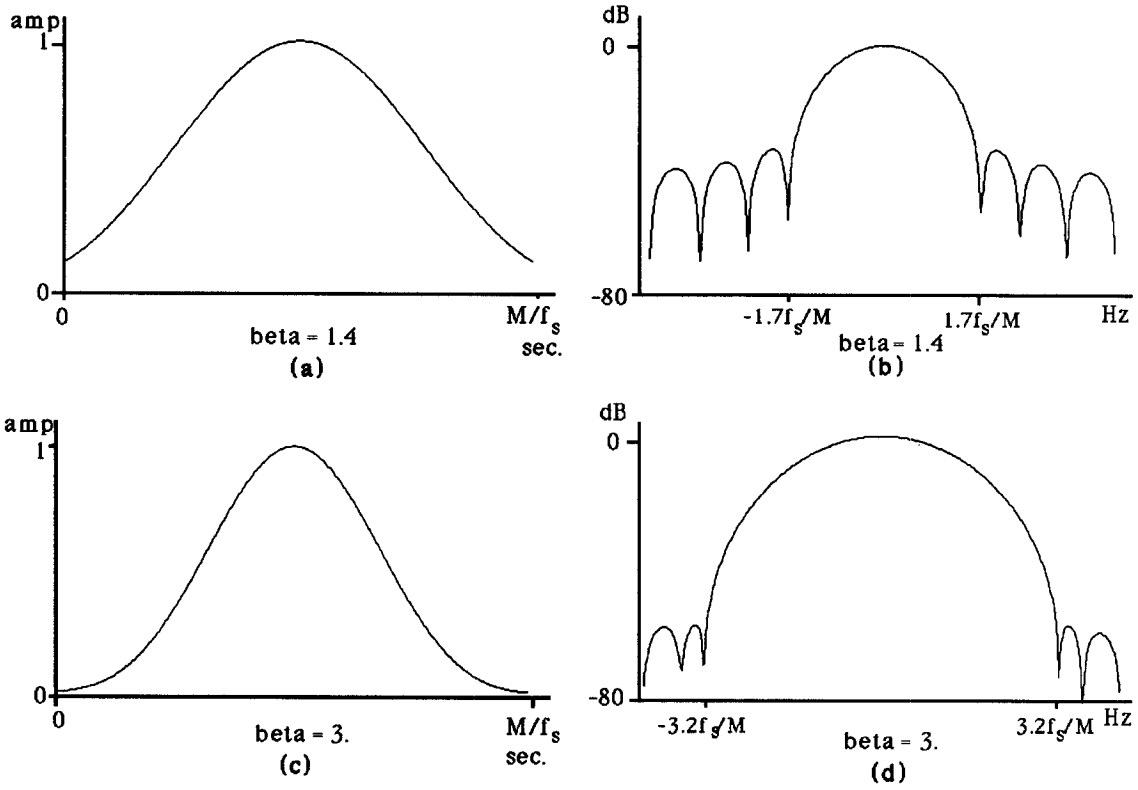

Figure 2.4: (a) Kaiser window M samples long, with $\beta = 1.4$, (b) magnitude spectrum of window, (c) Kaiser window with $\beta = 3$, (d) magnitude spectrum of window.

the highest side-lobe is 43dB down. A very different window, the Kaiser, allows control of the trade-off between the main-lobe width and the highest side-lobe level. If a narrower main-lobe width is desired then the side-lobe level will be higher, and vice versa. Since control of this trade-off is valuable, the Kaiser window is a good general-purpose choice. Figure 2.4 shows the Kaiser window and its magnitude spectrum, and Figure 2.5 shows the values of this trade-off as a function of the control parameter β (Kaiser, 1974).

Another good general purpose window for sound applications is the Blackman-Harris window (Harris, 1978; Nuttall, 1981). This window has four different settings for the main-lobe width versus side-lobe level trade-off. However, in this dissertation the precise control offered by the Kaiser window has proved very valuable and it is used throughout.

Let us look at this problem in actual practice. To "resolve" two sinusoids separated in frequency by ΔHz, we need (in noisy conditions) two clearly discernible main lobes, i.e.,

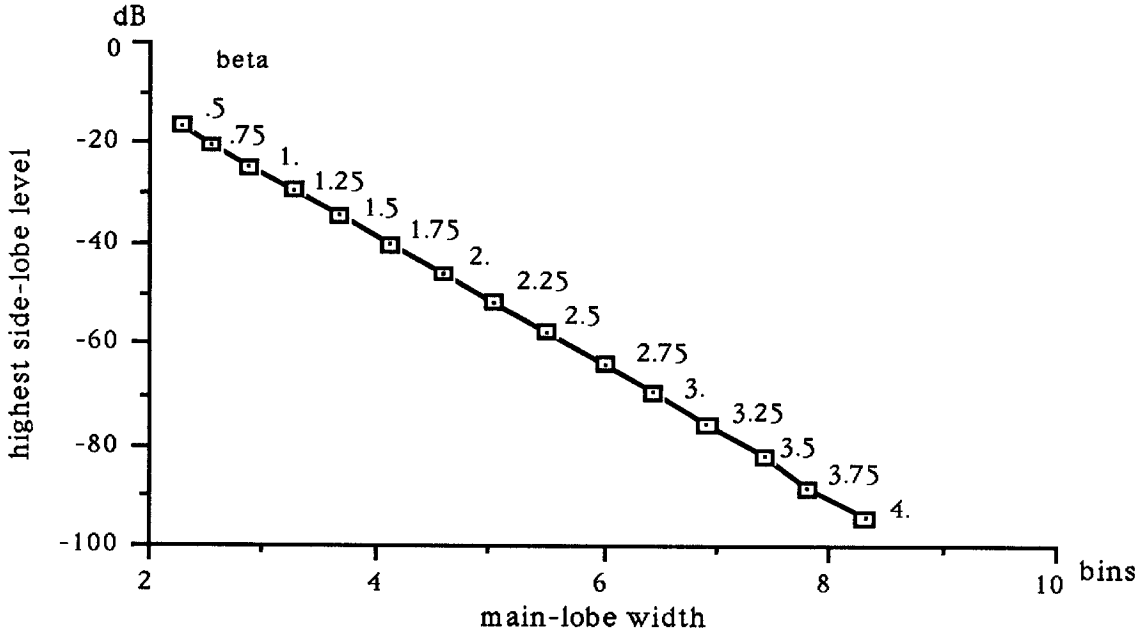

Figure 2.5: Table for the Kaiser window, showing the trade-off between main-lobe width and highest side-lobe level as a function of the control parameter β. The main-lobe width is expressed as the distance between the zero crossings.

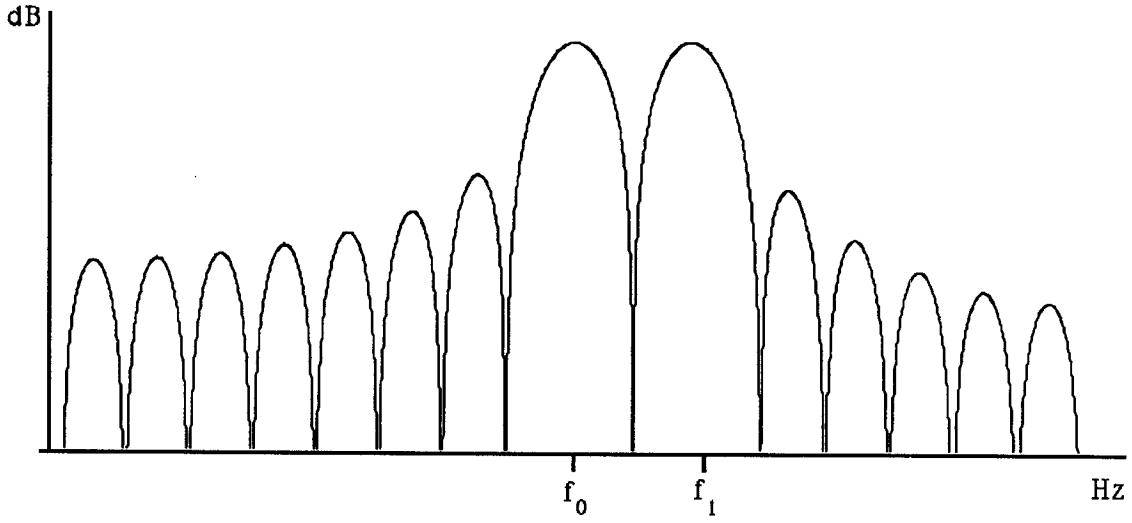

Figure 2.6: Magnitude spectrum of two sine waves with frequencies f_0 and f_1.

they should look something like Fig. 2.6. To obtain the separation shown (main lobes meet at zero-crossings), we require a main-lobe bandwidth in Hz, B_f, such that

$$B_f \leq \Delta \tag{2.14}$$

or, if

$$\begin{aligned} B_f &= B_s f_s / M \\ \Delta &= |f_{k+1} - f_k| \end{aligned} \tag{2.15}$$

where B_s is the main-lobe bandwidth (in samples), f_s the sampling rate, M is the window length, and f_k, f_{k+1} are the frequencies of the sinusoids, we need

$$M \geq B_s \frac{f_s}{\Delta} = B_s \frac{f_s}{|f_{k+1} - f_k|} \tag{2.16}$$

If f_k and f_{k+1} are successive harmonics of a fundamental frequency f_1, then $f_1 = f_{k+1} - f_k = \Delta$. Thus, harmonic resolution requires $B_f \leq f_1$ and thus $M \geq B_s f_s / f_1$. Note that $f_s/f_1 = T_1/T = P$, the period, in samples. Hence,

$$M \geq B_s P \tag{2.17}$$

Thus, with a Hamming window, with main-lobe bandwidth $B_s = 4$ bins, we want at least four periods of a harmonic signal under the window. From Fig. 2.5 we can see that if we want the same side-lobe level for the Kaiser window that the Hamming window has, β

has to be 1.75 and the number of periods under the window has to be a little bigger than 4. Thus, the Kaiser window can be set to approximately equal the Hamming window.

It is not always the case that four periods of a waveform can be taken to compute a single spectrum in the STFT. In some situations the waveform is not stable enough and a shorter window is needed. Thus, some compromises have to be made in order to obtain both frequency and time resolution. With the Kaiser window this is done very simply. For example, when a sound is very stable the window length M can be taken to be quite long, 4 or more periods, and the β parameter big, 2.5 or more, therefore obtaining a very good frequency resolution and good side-lobe rejection. When the sound is less stable, and the window has to be shorter, it is possible to maintain the frequency resolution by decreasing β and thus compromising on the side-lobe rejection.

In other situations the sound is not periodic and the four-periods rule cannot be applied either. In these situations the length of the window is determined by the closest two frequencies that we want to separate.

If the waveform to be analyzed has pronounced time-varying characteristics, either in terms of the frequencies present or of the instantaneous amplitude, it may be necessary to use a variable window size throughout the sound. Such a time-varying analysis causes problems in an overlap-add synthesis method, as will be shown later, but not in the sinusoidal representation introduced in the next chapter.[4]

A final point to be made about windows is the choice between odd and even length. A window of odd length can be centered around the middle sample, while one of even length does not have a mid-point sample. If one end point is deleted, an odd-length window can be overlapped and added so as to sum to a constant in the same way that the even length window does, a point which, as discussed in section 2.4.3, is important. For purposes of phase detection a constant-phase spectrum[5] is preferred, and this is obtained most naturally by using a symmetric window with the middle sample at the time origin.

---

[4]Though possible, a time-varying window-size has been unnecessary in the analysis/synthesis systems presented in the next chapters.

[5]A constant-phase spectrum is defined as the spectrum produced by a zero-phase window, that is, a spectrum in which the windowing process has not modified the phases of the analyzed waveform.

2.4.2 Computation of the DFT

Once a section of the waveform has been windowed (as shown in Figure 2.2), the next step is to compute the spectrum using the DFT. For practical purposes the FFT (fast Fourier transform) should be used whenever possible. But this requires the length of the analyzed signal to be a power of two, and in theory this should rarely be possible since the window length is controlled very precisely depending on the necessary resolution at every particular portion of a sound. However in practice this can be overcome by taking the appropriate window length and simply filling with zeros the rest of the length required by the FFT. This not only allows the use of the FFT algorithm, but it computes a smoother spectrum.

The FFT size N is normally chosen to be the first power of two that is at least twice the window length M, with the difference $N - M$ filled with zeros ("zero-padded"), since zero-padding in the time domain corresponds to interpolation in the frequency domain, and interpolating the spectrum has various benefits. First, it is easier to find spectral peaks which are not exact bin frequencies when the spectrum is more densely sampled (this is very important for the applications that will be developed in the next chapters). Second, plots of the magnitude of the more smoothly sampled spectrum are less likely to confuse the untrained eye. (Only signals truly periodic in M samples should not be zero-padded. They should also be windowed only by the Rectangular window.) Third, for overlap-add synthesis from spectral modifications (discussed below) the zero-padding allows for multiplicative modification in the frequency domain (convolutional modification in the time domain) without time-aliasing[6] in the inverse FFT. The length of the allowed convolution in the time domain (the impulse response of the effective digital filter) equals the number of extra zeros (plus one) in the zero padding.

If B_s is the number of samples in the main lobe when the zero-padding factor is 1 ($N = M$), then a zero-padding factor of N/M gives $B_s N/M$ samples for the same main lobe (and same main-lobe bandwidth). The zero-padding (interpolation) factor N/M should be large enough to enable an accurate estimation of the true maximum of the main lobe. That is, since the window length is not an exact number of periods for every frequency, the center of the spectral peaks does not correspond to the frequency bins, but by an appropriate zero-padding factor the center of the peak can be found. It has been determined by computational search that, for a rectangularly windowed sinusoid (as shown in Figure 2.3), quadratic frequency interpolation (using the three highest bins) yields at least 0.1% (of the distance from the sinc peak to the first zero-crossing) accuracy if the zero-padding factor N/M is 5 or higher.[7]

---

[6]Time-aliasing is defined as the overlap of samples.

[7]This peak interpolation strategy is discussed in detail in the next chapter.

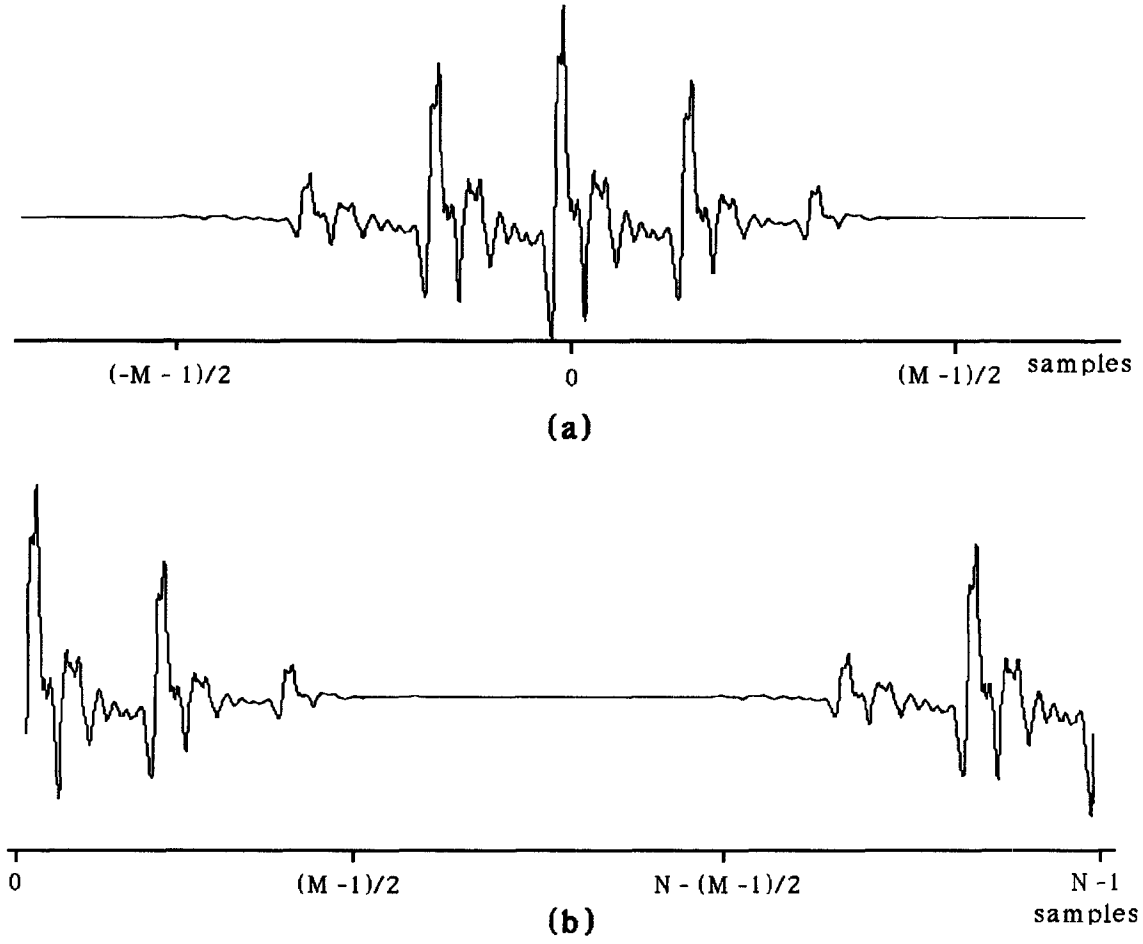

Figure 2.7: (a) Windowed waveform, (b) same waveform as it is stored in the FFT buffer for a constant-phase spectrum.

As mentioned in the previous section, we facilitate phase detection by using a zero-phase window, i.e., the windowed data (using an odd length window) is centered about the time origin. A zero-centered data frame appears in the length N FFT input buffer as shown in Figure 2.7. The first $(M-1)/2$ samples of the windowed data, the "negative-time" portion, are stored at the end of the buffer, from sample $N-(M-1)/2$ to $N-1$, and the remaining $(M+1)/2$ samples, the zero and "positive-time" portion, are stored starting at the beginning of the buffer, from sample 0 to $(M-1)/2$. Thus, all zero padding occurs in the *middle* of the FFT input buffer.

2.4.3 Choice of hop size

Once the spectrum has been computed at a particular point in the waveform, the STFT process hops along the waveform and computes the spectrum of another section in the sound. This hop size H, i.e., how much the analysis time origin is advanced from frame to frame, is an important parameter. Its choice depends very much on the purpose of the analysis. In general, more overlap will give more analysis points and therefore smoother results across time, but the computational expense is proportionately greater. A general and valid criterion is that the successive frames should overlap in time in such a way that all the data are weighted equally. For overlap-add synthesis, as will be seen in next section, this criterion is strictly followed. However for other synthesis techniques or different applications this criterion may be overly conservative when the signal is stable and too adventurous for fast-changing signals. Thus, the choice of hop size is determined by the application and the sound characteristics.

For certain window types there exist perfect overlap factors, that is, the windows can add perfectly to a constant. For example, a Rectangular window can hop by M/j, where j is any positive integer, and a Hanning or Hamming window can use any hop size of the form $(M/2)/j$. This overlap factor can be expressed by

$$A_w(m) \triangleq \sum_{n=-\infty}^{\infty} w(m - nH) = c \tag{2.18}$$

where $w(n)$ is the window, H the hop size, $A_w(m)$ the resulting envelope, and c a constant.[8] For most windows, however, there is no perfect hop size, and $A_w(m)$ is not a constant, other than for $H = 1$. A measure of the deviation of A_w from a constant is the difference between the maximum and minimum values for the envelope as a percentage of the maximum value,

$$d_w = 100 \times \frac{\max_m[A_w(m)] - \min_m[A_w(m)]}{\max_m[A_w(m)]} \tag{2.19}$$

This measure is referred to as the *amplitude deviation* of the overlap factor. Hop sizes of the form $(M/2)/j$ give good results (i.e., small amplitude deviations) in windows such as the Blackman or Blackman-Harris. However, with the Kaiser window the amplitude deviation depends on β. For the case of the Kaiser window, Figure 2.8 shows the envelope $A_w(m)$ for two different hop sizes and Figure 2.9 shows the amplitude deviation values for a hop size of $M/4$ and different β values.

The STFT process finishes when a spectrum has been computed at every frame in the waveform. Each spectrum is a complex valued function, that, as was shown, can be converted into polar coordinates to obtain the phase and magnitude values.

---

[8] Due to numerical errors there is always some small deviation from the ideal constant.

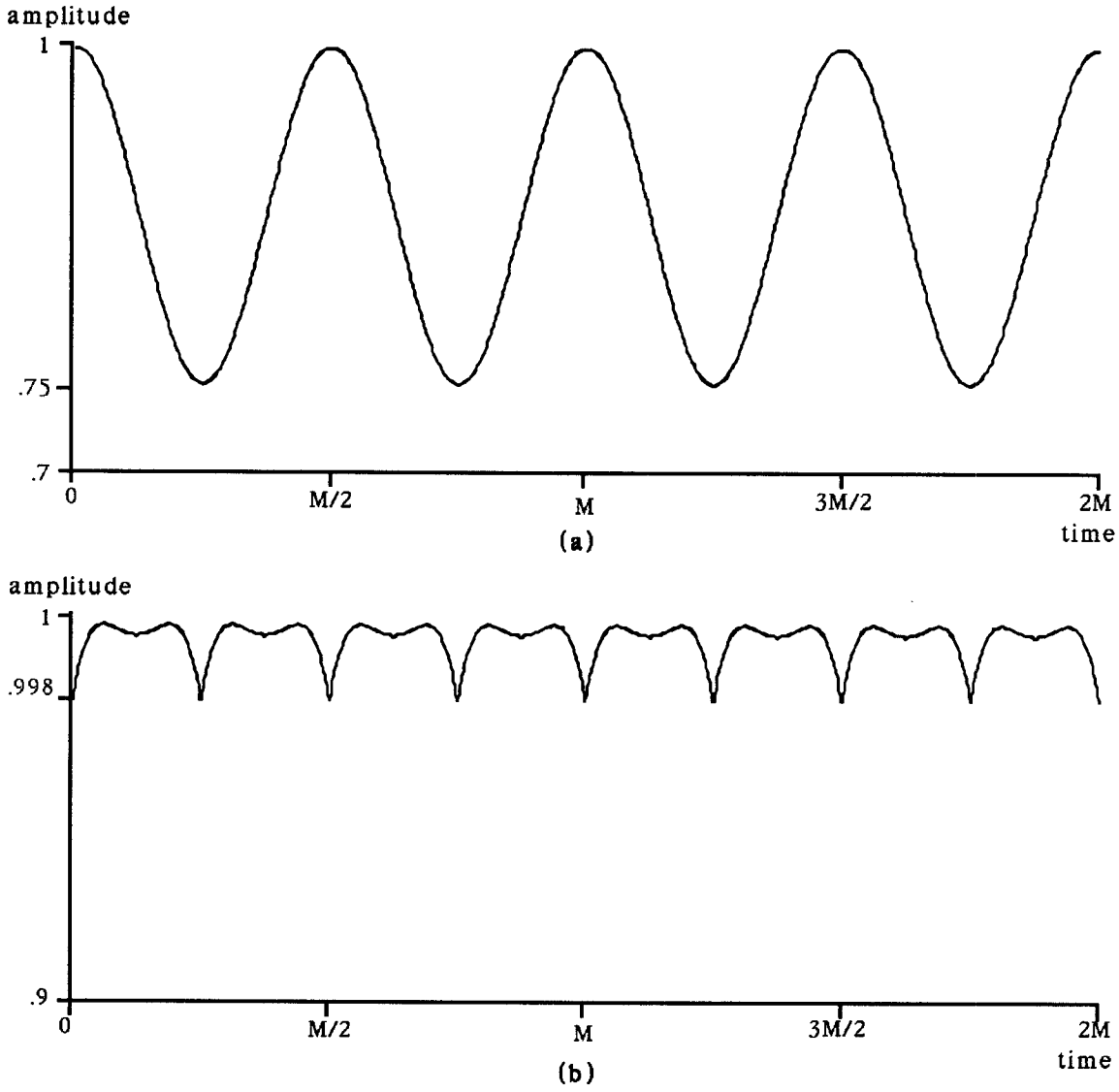

Figure 2.8: Overlap of a Kaiser window with $\beta = 2.5$ and length M: (a) hop size of $M/2$, (b) hop size of $M/4$.

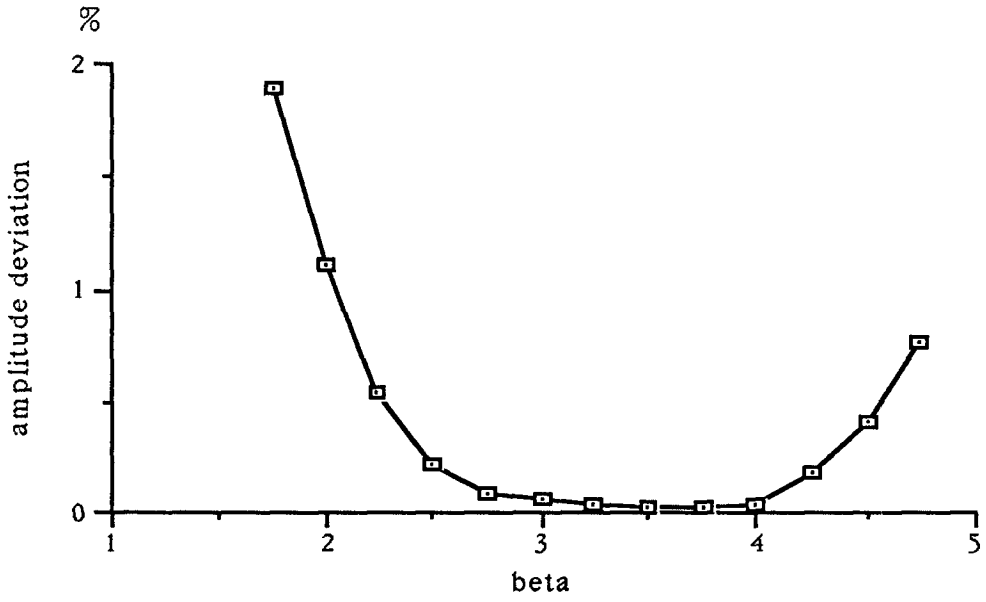

Figure 2.9: Amplitude-deviation for the Kaiser window for a constant hop size of 1/4 of the window length. The horizontal axis is the β parameter and the vertical axis is the deviation as a percentage of the maximum value.

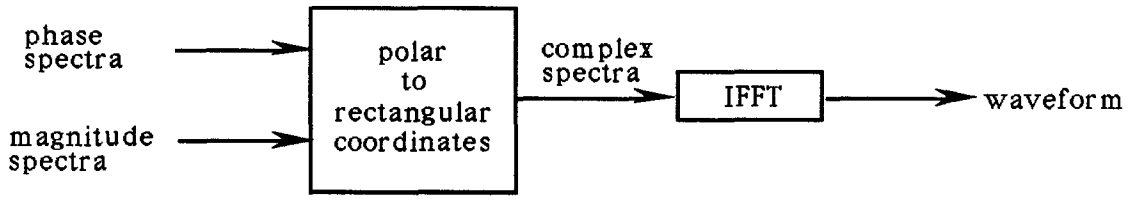

Figure 2.10: General diagram of the inverse STFT.

The control of the STFT is accomplished through the use of four parameters: *window-type, window-length, hop-size*, and *FFT-size*. These parameters will be often referred to in the following chapters.

2.5 Inverse Short-Time Fourier Transform

From the analysis data returned by the STFT we can recover the original waveform by computing the inverse STFT. Thus, the inverse process can be viewed as the synthesis part of the STFT. We can also apply some modifications to the spectra before resynthesis, such as multiplication by a smooth function (i.e., filtering in the frequency domain). A general diagram of the STFT synthesis process is shown in Figure 2.10.

The overlap-add interpretation of the STFT yields a particular method of synthesis (Allen and Rabiner, 1977). It corresponds to equation 2.6 and its implementation can be expressed by

$$s(n) = \sum_{l=0}^{L-1} \text{Shift}_{lH,n}\left[\frac{1}{K}\sum_{k=0}^{K-1} X_l(k)e^{j\omega_k m}\right] \tag{2.20}$$

where $X_l(k)$ is the spectrum computed at frame l, $m = 0, 1, \ldots, K-1$ is the time index inside the frame, and

$$\text{Shift}_{s,n}\left[x(m)\right] = \begin{cases} 0, & n < s \\ x(n-s), & s \le n < s+K \\ 0, & n \ge s+K \end{cases} \tag{2.20}$$

expresses the shift that takes place before the frames are added together.

This says that in order to reconstruct the signal, the spectrum $X_l(k)$ is inverse transformed for each l at which the analysis was performed, which, by definition of X, gives

$$s_l(n) = x(n + lH)\tilde{w}(n) \tag{2.21}$$

where $\tilde{w}(n)$ is the normalized window,

$$\tilde{w}(n) = \frac{w(n)}{\sum_{i=0}^{M/H-1} w(iH)} \tag{2.22}$$

responsible for normalizing the overlap-add sum.

Then $s_l(n)$ is summed over l to give

$$s(n) = \sum_{l=0}^{L-1} s_l(n - lH) = x(n) \sum_{l=0}^{L-1} w(n - lH) \tag{2.23}$$

that is, the final output is the sum of a series of windowed waveforms. A graphical example is shown in Figure 2.11.

Thus, it can be seen that analysis and resynthesis by overlap-add (in the absence of spectral modifications) is an *identity operation* (i.e., $s(n) = x(n)$) if the overlapped and added analysis windows sum to a constant (i.e., if $A_w(m) = \sum_{l=0}^{L-1} w(n - lH) =$ constant). When the overlap factor $A_w(m)$ is not constant, it is then an *amplitude modulation* envelope with period H (see Figure 2.8). That is, when the analysis window does not displace and add to a constant, the output is amplitude modulated by a periodic signal having its fundamental frequency at the frame rate f_s/H. However, this amplitude modulation is negligible when the amplitude deviation of the overlap factor is on the order of .03% or less.

A more general method of the inverse-STFT, called the weighted overlap-add method (Crochiere, 1980; Portnoff, 1980), uses a synthesis window prior to overlap-adding.[9] The identity property of the process is maintained if and only if the analysis window, $w(n)$, and the synthesis window, $w'(n)$, are related by

$$\sum_{n=-\infty}^{\infty} w(m - nH)w'(m - nH) = c \qquad \text{for all } m \tag{2.24}$$

that is, the function created by multiplying the two windows overlaps and adds to a constant.

The synthesis window is needed when the phase spectrum is modified. This is because the inverse Fourier transform of a spectrum with modified phases does not maintain the windowing performed in the analysis. Each resulting frame does not tapper smoothly to zero at the ends, which creates discontinuities at the frame boundaries.[10] In Chapter 5 this synthesis window is used for a similar purpose.

The filter-bank interpretation of the STFT (Allen and Rabiner, 1977) yields a synthesis method which synthesizes a sine wave for each frequency bin obtained in the analysis. These sine waves are added to recover the original waveform.

---

[9]The inverse-STFT discussed above assumes the synthesis window to be a rectangular window.

[10]In the phase-vocoder, where the original phase is redefined as the integral of the instantaneous frequency, the synthesis window makes sure that each inverse Fourier-transform tapers at the frame ends, thus allowing for a good overlapping.

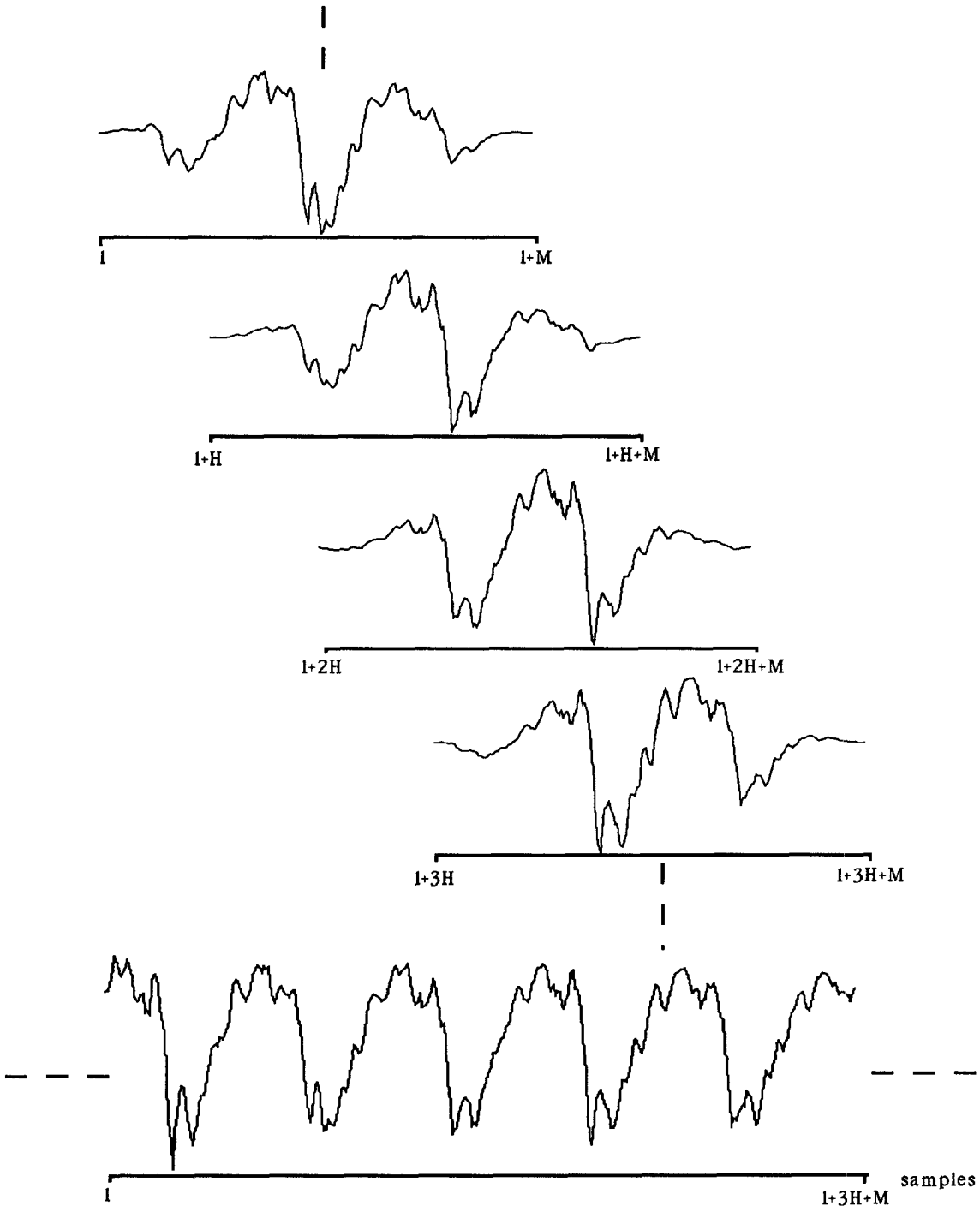

Figure 2.11: Example of the overlap-add synthesis process.

2.6 Summary of the STFT Analysis

A review of the main steps for a traditional computer implementation of an analysis system based on the STFT is given below.

1. Read M samples of the input signal x into a local buffer,

$$x_l(m) \triangleq x(m+lH), \quad m = \frac{-M-1}{2}, \frac{-M-1}{2}+1, \ldots, 0, \ldots, \frac{M-1}{2}-1, \frac{M-1}{2}, \quad M \text{ odd}, \quad l = 0,1,2,\ldots \tag{2.25}$$

where x_l is called the *frame* of the input signal at time lH, and M is the frame length.[11] The time advance H (in samples) from one frame to the next is called the *hop-size.*

2. Multiply the data frame pointwise by a length M *analysis window* $w(m)$, to obtain the windowed data frame at time l,

$$\tilde{x}_l(m) \triangleq x_l(m)w(m), \qquad m = \frac{-M-1}{2}, \ldots, \frac{M-1}{2} \tag{2.26}$$

3. Extend $\tilde{x}_l$ with zeros on both sides to obtain a *zero-padded* windowed data frame,

$$\tilde{x}'_l(m) \triangleq \begin{cases} \tilde{x}_l(m), & |m| \le \frac{M-1}{2} \\ 0, & \frac{M-1}{2} < m \le \frac{N}{2}-1 \\ 0, & -\frac{N}{2} \le m < -\frac{M-1}{2} \end{cases} \tag{2.26}$$

where N is chosen as a power of two larger than M. The number N/M is called the *zero-padding* factor.

4. Take a length N FFT of $\tilde{x}'_l$ to obtain the spectrum,

$$\tilde{X}'_l(k) = \sum_{n=-N/2}^{N/2-1} \tilde{x}'_l(n)e^{-j\omega_k n} \tag{2.27}$$

where $\omega_k = 2\pi k/N$, and k is the FFT *bin number.*

5. Convert each FFT bin of $\tilde{X}'_l(k)$ from rectangular to polar form to get the magnitude and phase in each FFT bin,

$$\begin{aligned} A_l(k) &\triangleq |\tilde{X}'_l(k)| \\ \Theta_l(k) &\triangleq \angle \tilde{X}'_l(k) \qquad \text{(radians)} \end{aligned} \tag{2.28}$$

[11] This summary considers the frame to be of odd length. The same formulation can be made for an even length frame.

2.7 Summary of the STFT Synthesis

Below are described the steps for a computer implementation of the STFT-synthesis by using what is called *overlap-add* reconstruction.

1. Apply any desired modification to the magnitude and phase spectra, such as multiplying it by a filter frequency response function, to obtain the modified frame spectra $\hat{A}_l(k)$ and $\hat{\Theta}_l(k)$.

2. Convert each FFT bin from polar to rectangular form to get back the complex number in each FFT bin,

$$\begin{aligned} \mathrm{Re}\{S_l(k)\} &\triangleq \hat{A}_l(k)\cos[\hat{\Theta}_l(k)] \\ \mathrm{Im}\{S_l(k)\} &\triangleq \hat{A}_l(k)\sin[\hat{\Theta}_l(k)] \end{aligned} \tag{2.29}$$

3. Inverse FFT S_l to obtain a time waveform,

$$s_l(m) = \frac{1}{N}\sum_{k=-N/2}^{N/2-1} S_l(k)e^{j\omega_k m} \tag{2.30}$$

   When modifications are performed on the phase spectrum apply a synthesis window to the resulting $s_l(m)$.

4. Reconstruct the final output by overlapping and adding the output frames,

$$s(n) = \sum_{l=0}^{L-1} s_l(n - lH) \tag{2.31}$$

2.8 Examples

The STFT is an identity system, independent of the parameter settings, as long as the analysis windows overlap and add to one. There is only relevance in the parameters when some modification is desired on the analysis data.

The following example shows the identity property of the STFT plus its capability to stretch a sound by integer factors. For examples of filtering using the STFT see appendix B.

2.8.1 Sound example 1

Excerpt from the Gloria of the Mass in C minor, K427, by Wolfgang Amadeus Mozart. (*sampling-rate* = 34000, length = 12 sec.)

Analysis parameters: *window-type* = Hamming, *window-length* = 2048 samples (.06 sec), *FFT-size* = 2048 samples, *hop-size* = 256 samples (.0075 sec).

1. original sound
2. synthesis from the STFT analysis
3. synthesis with a time expansion by a factor of 2

The time expanded synthesis is the result of doubling the hop size in the inverse-STFT from 256 samples to 512 samples (notice that the final overlap is still 1/4 of the window length).

The resynthesis without modifications is clearly identical to the original sound. The resynthesis with time expansion has quite a bit of distortion.

2.9 Conclusions

In this chapter the short-time Fourier transform (STFT) has been presented. This analysis/synthesis technique can be thought of as the time-varying version of the Fourier transform, in which a sound is represented by a set of complex spectra. For the purpose of this dissertation, the STFT is not appropriate for the manipulation of sounds; only limited time scaling and filtering are possible.[12] However it is an excellent intermediate technique for more suitable analysis/synthesis systems. The next chapter presents a simplification of the STFT that gives a more flexible sound representation.

[12]The Phase-vocoder is a more flexible STFT implementation which allows for better time-scaling transformations plus frequency transposition of sounds. However, there has been no need to discuss this technique in detail.

Chapter 3

A Sinusoidal Model

3.1 Introduction

The analysis/synthesis technique presented in the previous chapter, the STFT, is not a flexible sound representation, and thus, not very appropriate for sound modifications. However, it is useful as the basis of more suitable representations. In this chapter a sinusoidal representation based on the STFT is introduced that is characterized by the amplitudes, frequencies, and phases of the component sine waves. The representation results from following the amplitude, frequency, and phase of the most prominent peaks over time in the series of spectra returned by the STFT. From this representation, or a modification of it, a sound is generated by synthesizing a sine wave for each peak trajectory found. In the absence of modifications the process can produce a perceptual identity; that is, the synthesized sound can be made to be perceptually equal to the original one. The analysis results can be modified to obtain new sounds in the synthesis process.

This kind of system can be understood as an instantiation of a tracking phase-vocoder (Dolson, 1983) in which there are a set of band-pass filters and each filter follows and extracts a particular energy component of the input sound. The traditional phase-vocoder (Flanagan, 1966; Portnoff, 1976) is the particular case in which the filters are equally spaced and non-time-varying. We can also interpret the sinusoidal representation as a simplification of the output of the STFT, where only the relevant spectral peaks are taken from the set of spectra returned by the STFT. These peaks, each representing a sinusoid, are then grouped into frequency trajectories.

Sinusoidal representations have been used extensively in music applications (Risset and Mathews, 1969; Grey, 1975; Moorer, 1973, 1975, 1977, 1978; Strawn, 1980). However the particular sinusoidal representation discussed in this chapter has only recently been proposed and used (McAulay and Quatiery, 1984, 1986; Quatiery and McAulay, 1986; Smith and Serra 1987; Maher 1989). This representation has proved to be more general than the previous sinusoidal representations. For the purpose of this thesis its interest is as an analysis/transformation/synthesis system, where sounds can be analyzed and transformed in different ways before resynthesis. It will be shown that even though it is more flexible than

the STFT as a sound modification technique, sinusoidal representations are not appropriate for manipulating sounds that have noise components. In the next two chapters, alternative representations that extend this one are presented to include such sounds.

In this chapter, the model which serves as the basis for the sinusoidal representation is presented first. Then, there is a general description of the system, and in the following sections the different steps involved in the process are discussed. The chapter ends with a summary of the system, a presentation of some sound examples, and conclusions.

3.2 The Sinusoidal Model

The sinusoidal model is the basis for the analysis/synthesis system presented in this chapter. In this model the waveform $s(t)$ is assumed to be the sum of a series of sinusoids,

$$s(t) = \sum_{r=1}^{R} A_r(t) \cos[\theta_r(t)] \tag{3.1}$$

where R is the number of sine-wave components, $A_r(t)$ the instantaneous amplitude and $\theta_r(t)$ the instantaneous phase. This instantaneous phase is defined by:

$$\theta_r(t) = \int_0^t \omega_r(\tau) d\tau + \theta_r(0) + \phi_r \tag{3.2}$$

where $\omega_r(t)$ is the instantaneous radian frequency, $\theta_r(0)$ the initial phase value, and ϕ_r the fixed phase offset, which accommodates the fact that the sine waves are generally not in phase.

3.3 General Description of the System

Figure 3.1 shows a general block diagram of a system based on the sinusoidal model. It starts by computing the STFT, in the manner presented in Chapter 2. Then, from the magnitude and phase spectra returned by the STFT, a series of peak trajectories are extracted by a peak detection and a peak continuation algorithm. Each trajectory represents a sinusoid characterized by time-varying phase, frequency, and magnitude values. The synthesis part of the system uses the peak trajectories to generate sine waves that are added to create the final synthesized waveform.

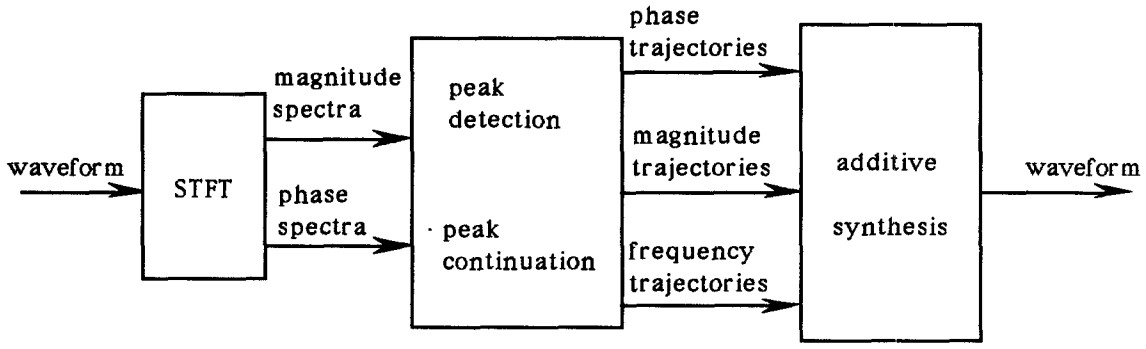

Figure 3.1: General block diagram of the sinusoidal system.

3.4 Computation of the Magnitude and Phase Spectra

The analysis/synthesis system starts by computing a set of spectra with the STFT. Since the details of this computation were discussed in the previous chapter, only the distinct aspects affecting the current system are mentioned here.

The sinusoidal system detects the prominent spectral peaks out of the magnitude and phase spectra of the sound. Thus, conversion of each spectrum from rectangular to polar coordinates is required. Then, since the system detects the prominent peaks in the magnitude spectra, it is important to have the peaks as well resolved as possible. It was shown in Chapter 2 that zero-padding results in a smoother spectrum, making the peak detection easier and more accurate. Here, the zero-padding factor should be as large as it is practical.

Another point concerning the STFT is related to the synthesis part of the system. The synthesis process is based on an additive synthesis model, not an overlap-add one. This implies that the restriction imposed for the overlap-add method, that the analysis windows add to a constant (or close to it), is unnecessary. Now the *hop-size* of the analysis window, a parameter that affects the overlap factor, is more flexible than in an overlap-add synthesis process.

3.5 Spectral Peak Detection

Once the set of complex spectra of a sound is computed and converted to polar coordinates, the system extracts the prominent peaks of each spectrum. In this section, the peak detection algorithm is described.

A peak is defined as a local maximum in the magnitude spectrum $|X_l(k)|$, where l is the frame number. If k_β is a bin number in the spectrum, then its value is a maximum when

$$|X(k_\beta - 1)| \leq |X(k_\beta)| \geq |X(k_\beta + 1)| \tag{3.3}$$

However not all the peaks are equally prominent in the spectrum and it is important to have control over their selection. This is done by measuring the height of the peaks in relation to the neighboring valleys. Where the neighboring valleys are the closest local minima on both sides of the peak. If the detected valleys for $X(k_\beta)$ are $X(k_\gamma-)$ and $X(k_\gamma+)$, left and right respectively, then a measure of the peak height, $h(k_\beta)$, is determined by

$$h(k_\beta) \triangleq \frac{|X(k_\beta)|}{[|X(k_\gamma-)| + |X(k_\gamma+)|]/2} \tag{3.4}$$

For perceptual purposes it is useful to convert the magnitude into decibels (dB) by

$$\hat{X}(k) = 20\log_{10}|X(k)| \tag{3.5}$$

where $X(k)$ is the linear magnitude spectra and $\hat{X}(k)$ is the magnitude spectra in dB. Then, the peak height is redefined as

$$h(k_\beta) \triangleq \hat{X}(k_\beta) - \frac{\left[\hat{X}(k_\gamma-) + \hat{X}(k_\gamma+)\right]}{2} \tag{3.6}$$

A parameter in the peak detection algorithm, called *minimum-peak-height*, uses this measure to control the minimum height (in dB) at which a peak is detected.

This is more complex because not all peaks of the same height are equally relevant perceptually, their amplitude and frequency is very important. There are many factors which intervene on this issue and it can become an extremely complicated problem. Here, a very simple method is devised that controls the frequency and magnitude ranges to be considered in each spectrum. A more elaborate strategy is proposed by Terhardt (Terhardt, Stoll and Seewann, 1982a, 1982b) for the purpose of perceptual analysis, which however, is not appropriate in an analysis/synthesis system.

The spectral peaks are searched within a *frequency-range* described by its lower and upper bounds. If f_l and f_h are these bounds in Hz, the corresponding frequency bins, k_l and k_h, are then obtained by

$$\begin{aligned} k_l &= f_l N/f_s \\ k_h &= f_h N/f_s \end{aligned} \tag{3.7}$$

where N is the *FFT-size* and f_s the sampling rate.

By choosing an appropriate range, regions outside the auditory frequency range are discarded. Practical values for f_l and f_h are 20Hz and 16KHz respectively.

The selection of a magnitude range is more complicated. First, since the perception of magnitude is approximately logarithmic, it is important to use a dB scale as calculated in equation 3.5. For convenience, the maximum value is set to 0dB. Then, the magnitude range is specified by a number that expresses the lowest dB magnitude that the peak detection algorithm will search for. In most situations it is important to have two different ranges, one relative to the overall sound (*general-dB-range*) and another one relative to the maximum magnitude of the current frame (*local-dB-range*). For each spectrum the two ranges are compared and the widest one is taken. Typical bottom values of the ranges are -70dB for the overall one and -60dB for the local one. Then, for example, if a peak is at -75dB in a spectrum whose local maximum is 30dB below the overall maximum (the peak is 45dB down from the local maximum), this peak is detected since it is inside the local range, even though is outside the overall range. Thus, in a quiet passage softer peaks are detected, mimicking the auditory system.

Another attribute of the auditory system is that it does not necessarily perceive two different frequency components of the same complex tone (e.g., two partials) with the same physical magnitude as being equally loud. The equal loudness curve across the frequency range is not flat. Thus, prior to the peak detection, we might want to equalize the magnitude spectrum according to an equal-loudness criterion. The problem is to find the appropriate equal-loudness curve to use. Unfortunately, the data of traditional loudness experiments are valid only for the comparison of separate tones, whether they are sinusoids (Fletcher and Munson, 1933) or complex tones (Zwicker and Scharf, 1965). Here we are dealing with components of a complex tone, not independent tones, and there is no conclusive literature on this subject. A practical compromise is to design a smooth function which approximates one of the equal loudness curves from Fletcher and Munson (Fletcher and Munson, 1933). We have chosen the 40dB curve, whose approximation is given by the function

$$Q(x) = x10^{-x} \tag{3.8}$$

where

$$x = .05 + \frac{4000}{f} \tag{3.9}$$

and f is the frequency in Hz. This function is then applied to every spectrum, independent of the specific magnitude of each component frequency. In Figure 3.2 this function and its effect on a spectrum are shown.

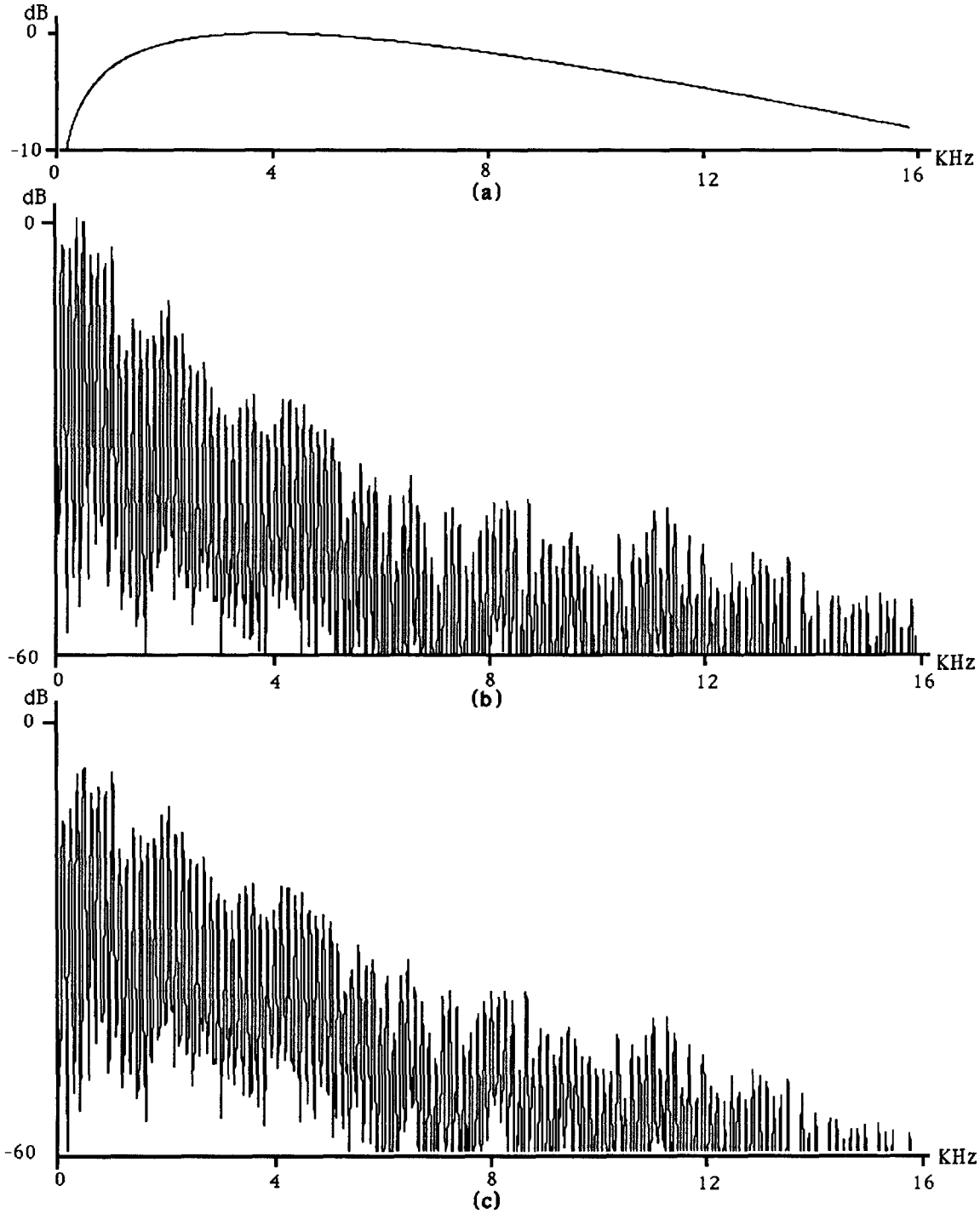

Figure 3.2: Applying an equal-loudness curve to a spectrum: (a) equal-loudness curve, (b) magnitude spectrum of a saxophone sound, (b) equalized spectrum.

3.5.1 Peak interpolation

Due to the sampled nature of the spectra returned by the STFT, each peak–a spectral bin that is a local maximum–is accurate only to within half a sample. A bin (sample in the frequency spectrum) represents a frequency interval of f_s/NHz, where N is the FFT size. As we saw in Chapter 2, zero-padding in the time domain increases the number of DFT bins per Hz and thus increases the accuracy of the simple peak detection. However, to obtain frequency accuracy on the level of 0.1% of the distance from the top of the sinc function to its first zero crossing (in the case of a rectangular window), the zero-padding factor required is 1000. Since we take at least two periods in the data frame (for a Rectangular window), a 100Hz sinusoid at a sampling rate of 50KHz has a period of $50,000/100 = 500$ samples, so that the FFT size must exceed one million. A more efficient spectral interpolation scheme is to zero-pad only enough so that quadratic (or other simple) spectral interpolation, using only bins immediately surrounding the maximum-magnitude bin, suffices to refine the estimate to 0.1% accuracy.

We have seen that a sinusoid appears as a shifted window transform, which is a sinc-like function. A robust method for estimating peak frequency of stable sinusoidal components with very high accuracy fits a window transform to the sampled spectral peaks by cross-correlating the whole window transform with the entire spectrum and taking an interpolated peak location in the cross-correlation function as the frequency estimate. This method offers much greater immunity to noise and to interference from other signal components. But such a method is computationally very expensive and not appropriate for peaks which do not correspond to stable sinusoidal components. For the current system a practical solution is to use a parabolic interpolator which fits a parabola through the highest three samples of a peak to estimate the true peak location and height (Smith and Serra, 1987), as shown in Fig. 3.3.

To describe the parabolic interpolation strategy, let us define a coordinate system centered at $(k_\beta, 0)$, where k_β is the bin number of a spectral magnitude maximum (Fig. 3.3). We desire a general parabola of the form

$$y(x) \triangleq a(x-p)^2 + b, \tag{3.10}$$

where p is the center of the parabola, a is a measure of the concavity, and b is the offset. In the current problem we set $y(-1) = \alpha$, $y(0) = \beta$, and $y(1) = \gamma$, where α, β, and γ are the values of the three highest samples,

$$\begin{aligned} \alpha &\triangleq 20\log_{10}|X(k_\beta - 1)| \\ \beta &\triangleq 20\log_{10}|X(k_\beta)| \\ \gamma &\triangleq 20\log_{10}|X(k_\beta + 1)| \end{aligned} \tag{3.11}$$

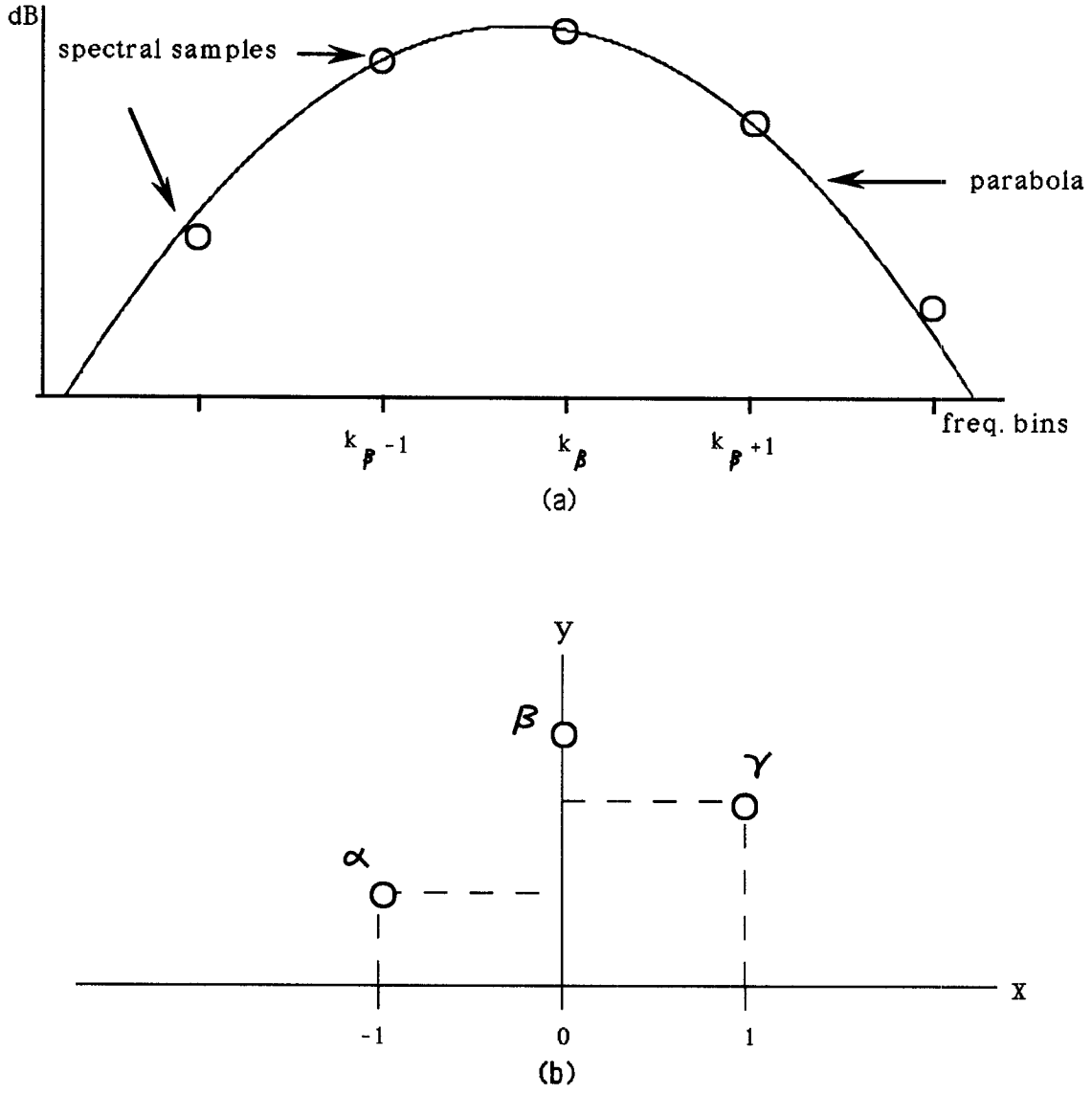

Figure 3.3: Parabolic interpolation: (a) illustration on a spectral peak, (b) coordinates to perform the interpolation.

It has been found empirically that the frequencies tend to be about twice as accurate when dB magnitude is used rather than linear magnitude.

Solving for the parabola peak location p, we get

$$p = \frac{1}{2}\frac{\alpha - \gamma}{\alpha - 2\beta + \gamma} \tag{3.12}$$

then the estimate of the true peak location (in bins) is

$$k^* \triangleq k_\beta + p \tag{3.13}$$

and the peak frequency in Hz is $f_s k^*/N$. Using p, the peak height (magnitude) estimate is then

$$y(p) = \beta - \frac{1}{4}(\alpha - \gamma)p \tag{3.14}$$

In the system, the magnitude spectrum is used only to find p, but $y(p)$ is computed separately for the real and imaginary parts of the complex spectrum to yield a complex-valued peak estimate (magnitude and phase). The result of the peak detection algorithm is a triad of the form $(\hat{A},\hat{\omega},\hat{\varphi})$ for every peak, where $\hat{A}$ is the estimated amplitude of the peak, $\hat{\omega}$ the radian frequency, and $\hat{\varphi}$ the phase.

The success of the parabolic interpolation depends on the analysis window used. Among all the windows the Gaussian is, in theory, particularly suited for parabolic interpolation. This window, which is of the form

$$w(x) = e^{-(1/2)x^2} \tag{3.15}$$

transforms to a Gaussian window (Harris, 1978), and its log is just a parabola,

$$\ln[w(x)] = -\frac{1}{2}x^2 \tag{3.16}$$

Thus, parabolic interpolation in the dB spectrum is perfect for the Gaussian window. However, this window does not terminate and in practice it is truncated, discarding the tails. Then, the perfect interpolation is lost in part. A possible compromise is to taper the ends of the window smoothly, with, for example, a Kaiser window, thus preserving some of the characteristics.

It is important to normalize the amplitude values returned by the peak detection in such a way that they correspond to the actual sinusoidal amplitudes. Then the synthesis generates sinusoids which reproduce the amplitudes of the original sound. The amplitude of the spectral peak is dependent on the analysis window used. In order to normalize it the measured amplitude is multiplied by a scale factor,

$$\alpha = \frac{2}{W(0)} \tag{3.17}$$

where $W(0)$ is the value of the window transform at time 0, which can be calculated in the time domain by

$$W(0) = \sum_{m=0}^{M-1} w(m) \tag{3.18}$$

where $w(n)$ is the time domain window and M is the *window-length.*

Figure 3.4 shows the result of the peak detection algorithm on a magnitude and a phase spectrum.

3.6 Spectral Peak Continuation

The peak detection process returns the estimated magnitude, frequency, and phase of the prominent peaks in a given frame sorted by frequency. The next step is to assign these peaks to frequency trajectories using the peak continuation algorithm. If the number of spectral peaks were constant with slowly changing amplitudes and frequencies, this task would be straightforward. However, this is not often the case.

There are many possibilities for such a process. Here, we present a simple and general method which is adequate for the analysis/synthesis system of this chapter. This algorithm is used by McAulay and Quatiery in their sinusoidal representation (McAulay and Quatiery, 1986). A more complex algorithm is developed in Chapter 4 for a different type of system.

To describe the peak continuation process let us assume that the frequency trajectories are initialized at frame 1 and that we are currently at frame n. Suppose that at frame $n-1$ the frequency values for the p track are $f_1, f_2, \ldots, f_p$, and that we want to match them to the r peaks of frame n, with frequencies $g_1, g_2, \ldots, g_r$.

Each trajectory looks for its peak in frame n by finding the one which is closest in frequency to its current value. The ith trajectory claims frequency g_j for which $|f_i - g_j|$ is minimum. The change in frequency must be less than a specified maximum Δf_i, which can be frequency-dependent (e.g., linear, corresponding to a relative frequency change limit). The parameter controlling this value is called *maximum-peak-deviation.* The possible situations are as follows:

1. If a match is found inside the *maximum-peak-deviation*, the trajectory is continued (unless there is a conflict to resolve, as described below).
2. If no match is made, it is assumed that the trajectory with frequency f_i must be "killed" entering frame n, and f_i is matched to itself with zero magnitude. Since the track amplitudes are linearly ramped from one frame to the next, the terminating trajectory ramps to zero over the duration of one hop size.

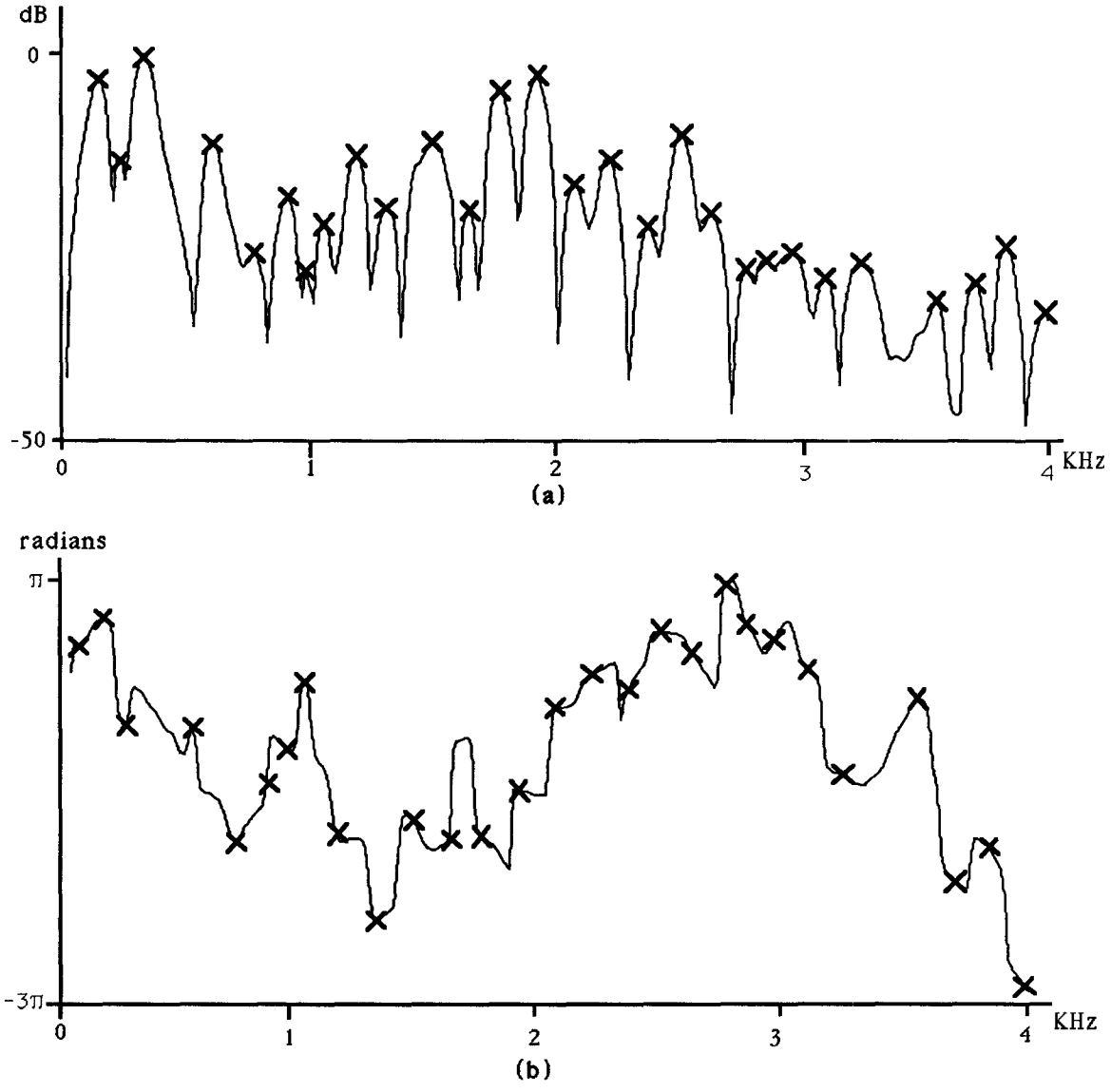

Figure 3.4: Peak detection on a spectrum of a piano attack sound: (a) magnitude spectrum, (b) phase spectrum.

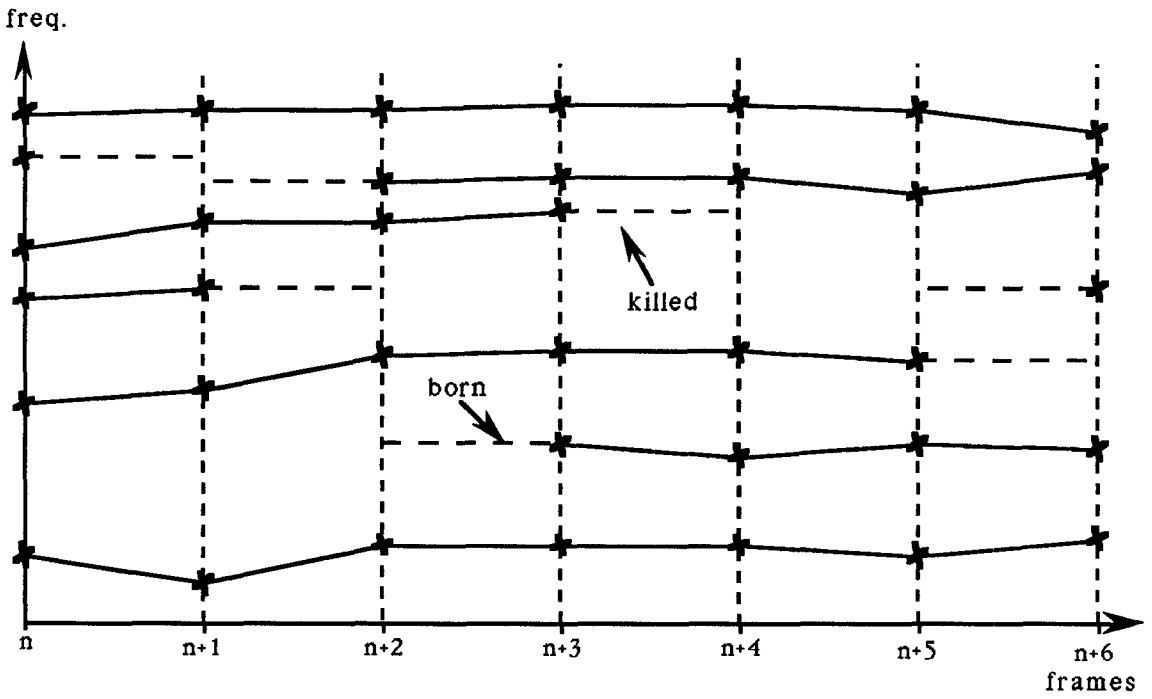

Figure 3.5: Illustration of the peak continuation process.

3. If a trajectory finds a match that has already been claimed by another one, we give the peak to the trajectory which is closest in frequency, and the "loser" looks for another match. If the current trajectory loses the conflict, it simply picks the best available non-conflicting peak which is inside the allowable deviation. If the current trajectory wins the conflict, it calls the assignment procedure recursively on behalf of the dislodged trajectory. When the dislodged trajectory finds the same peak and wants to claim it, it sees that there is a conflict which it loses and will move on. This process is repeated for each trajectory, solving conflicts recursively, until all existing tracks are matched or "killed."

After each trajectory has extended itself forward in time, or turned off, the peaks of frame n which have not been used are considered to be new trajectories and a new trajectory is "born" for each one of them up to the maximum number of tracks specified. The new trajectories are started at frame $n-1$ with zero magnitude, and ramped to the correct amplitude at the current frame n. A few frames of the peak-matching process are illustrated by Fig. 3.5.

3.7 Sinusoidal Synthesis

The peak continuation algorithm returns the values of the prominent peaks organized into frequency trajectories. Each peak is a triad $(\hat{A}_r^l, \hat{\omega}_r^l, \hat{\varphi}_r^l)$ where l is the frame number and r the track number to which it belongs.

The synthesis process takes these trajectories, or their modification, and computes one frame of the synthesized sound $s(n)$ by

$$s^l(m) = \sum_{r=1}^{R^l} \hat{A}_r^l \cos[m\hat{\omega}_r^l + \hat{\varphi}_r^l], \qquad m = 0, 1, 2, \ldots, S-1 \tag{3.19}$$

where R^l is the number of trajectories present at frame l and S is the length of the synthesis frame.[1] The final sound $s(n)$ results from the juxtaposition of all the synthesis frame (i.e., there is no overlap). To avoid "clicks" at the frame boundaries, the parameters $(\hat{A}_r^l, \hat{\omega}_r^l, \hat{\varphi}_r^l)$ are smoothly interpolated from frame to frame.

Let $(\hat{A}_r^{(l-1)}, \hat{\omega}_r^{(l-1)}, \hat{\varphi}_r^{(l-1)})$ and $(\hat{A}_r^l, \hat{\omega}_r^l, \hat{\varphi}_r^l)$ denote the sets of parameters at frames $l-1$ and l for the rth frequency trajectory (we will simplify the notation by omitting the subscript r). These parameters are taken to represent the state of the signal at time 0 (the left endpoint) of the frame.

The instantaneous amplitude $\hat{A}(m)$ is easily obtained by linear interpolation,

$$\hat{A}(m) = \hat{A}^{l-1} + \frac{(\hat{A}^l - \hat{A}^{l-1})}{S} m \tag{3.20}$$

where $m = 0, 1, \ldots, S-1$ is the time sample into the lth frame.

Frequency and phase values are tied together (frequency is the phase derivative), and both control the instantaneous phase $\hat{\theta}(m)$, defined as

$$\hat{\theta}(m) = m\hat{\omega} + \hat{\varphi} \tag{3.21}$$

Given that four variables affect the instantaneous phase: $\hat{\omega}^{(l-1)}$, $\hat{\varphi}^{(l-1)}$, $\hat{\omega}^l$, and $\hat{\varphi}^l$, we need three degrees of freedom for its control, but linear interpolation gives only one. Therefore, we need a cubic polynomial as an interpolation function,

$$\hat{\theta}(m) = \zeta + \kappa m + \eta m^2 + \iota m^3 \tag{3.22}$$

---

[1] A synthesis frame is S samples long and does not correspond to an analysis frame. Without time scaling the synthesis frame l goes from the middle of the analysis frame $l-1$ to the middle of the analysis frame l, i.e., corresponds to the analysis hop size.

It is unnecessary to go into the details of solving this equation since they are described by McAulay and Quatieri (McAulay and Quatieri, 1986). The result is

$$\hat{\theta}(m) = \hat{\varphi}^{(l-1)} + \hat{\omega}^{(l-1)}m + \eta m^2 + \iota m^3 \tag{3.23}$$

where η and ι are calculated using the end conditions at the frame boundaries,

$$\begin{aligned} \eta &= \frac{3}{S^2}(\hat{\varphi}^l - \hat{\varphi}^{l-1} - \hat{\omega}^{l-1}S + 2\pi M) - \frac{1}{S}(\hat{\omega}^l - \hat{\omega}^{l-1}) \\ \iota &= -\frac{2}{S^3}(\hat{\varphi}^l - \hat{\varphi}^{l-1} - \hat{\omega}^{l-1}S + 2\pi M) + \frac{1}{S^2}(\hat{\omega}^l - \hat{\omega}^{l-1}) \end{aligned} \tag{3.24}$$

This gives a set of interpolating functions depending on the value of M, among which we select the maximally smooth function. This is done by choosing M to be the integer closest to x, where x is

$$x = \frac{1}{2\pi}\left[(\hat{\varphi}^{l-1} + \hat{\omega}^{l-1}S - \hat{\varphi}^l) + \frac{S}{2}(\hat{\omega}^l - \hat{\omega}^{l-1})\right] \tag{3.25}$$

Finally, the synthesis equation for frame l becomes

$$s^l(m) = \sum_{r=1}^{R^l} \hat{A}_r^l(m)\cos[\hat{\theta}_r^l(m)] \tag{3.26}$$

which goes smoothly from frame to frame with each sinusoid accounting for both the rapid phase changes (frequency) and the slowly varying phase changes (Fig. 3.6).

3.8 Representation Modifications

The possibilities that this analysis/synthesis system offers for sound transformations have a number of musical applications. Quatieri and McAulay (Quatieri and McAulay, 1986) indicate some useful modifications for speech applications and Smith and Serra (Smith and Serra, 1987) discuss more musical applications. All the modifications are obtained by scaling and/or resampling the amplitude and the frequency trajectories.

Time-scale modifications are accomplished by resampling the amplitude, frequency, and phase trajectories. This is done by changing the synthesis frame-size, slowing down or speeding up the sound while maintaining pitch and formant structure. A time-varying frame-size gives a time-varying modification. However, due to the sinusoidal nature of the representation, a considerable time stretch in a "noisy" part of a sound, causes the individual sine waves to be heard and the noise-like quality is lost.

Frequency transformations, with or without time scaling, are also possible. A simple one is to scale the frequencies to alter pitch and formant structure together. A more powerful class of spectral modifications comes about by decoupling the sinusoidal frequencies (which convey pitch and inharmonicity information) from the spectral envelope (which conveys

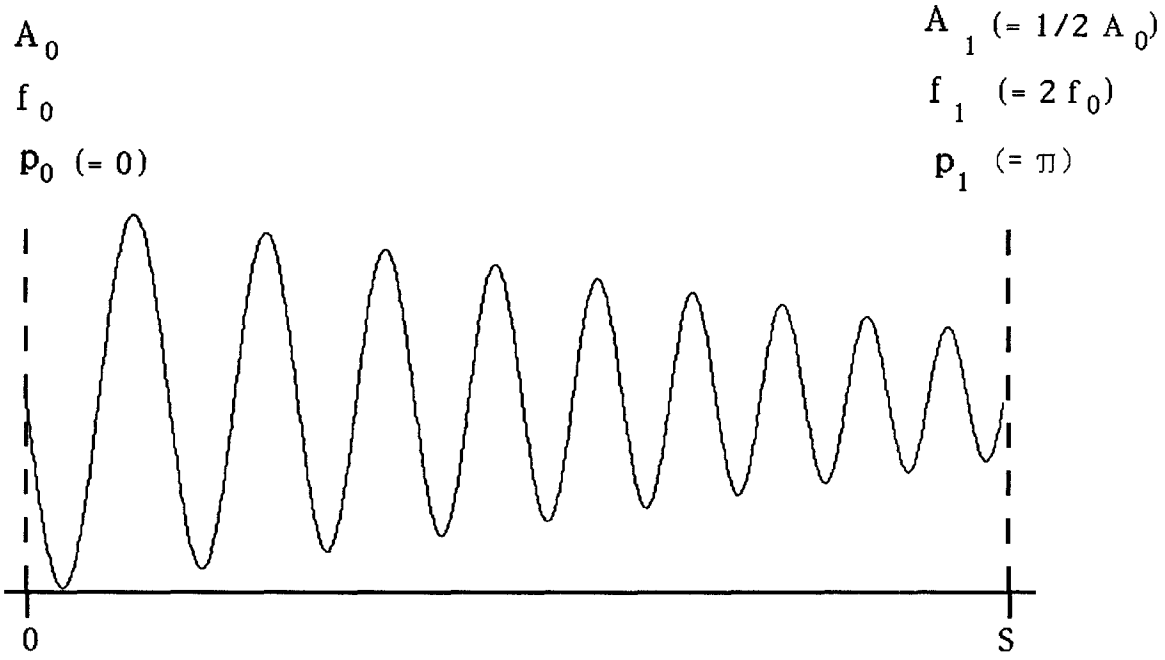

Figure 3.6: Example of the frame to frame interpolation used in the synthesis process. A, f, and p are the amplitude frequency and phase values respectively.

formant structure so important to speech perception and timbre). By measuring the formant envelope of a harmonic spectrum (e.g., by drawing straight lines or splines across the tops of the sinusoidal peaks in the spectrum and then smoothing), modifications are introduced which alter only the pitch or only the formants.

3.9 Magnitude-Only Analysis/Synthesis

A traditional principle of sound perception is that the ear is mainly sensitive to the short-time spectral magnitude and not to the phase, provided phase continuity is maintained. Our experience has been that this depends on the sound and application. If the phase information is discarded, the analysis, modification, and synthesis processes are simplified enormously. Thus, it is better to use the magnitude-only option of the system whenever auditory considerations permit.

In the peak-detection process, we calculate the magnitude and phase of each peak by using the complex spectrum. Once we decide to discard the phase information, there is no need for complex spectra, and the magnitude of the peak is calculated by doing the parabolic interpolation directly on the log magnitude spectrum.

The synthesis also becomes easier; there is no need for a cubic function to interpolate the instantaneous phase. The phase becomes a function of the instantaneous frequency, and we only require phase continuity at the frame boundaries. Therefore, the frequency, like the amplitude, can be linearly interpolated from frame to frame. Without phase matching the synthesized waveform looks very different from the original (Fig. 3.7), but for many applications the perceived sound quality is the same.

3.10 Summary of the Technique

To summarize the technique presented in this chapter let us enumerate the main steps that are carried out. Figure 3.8 shows a block diagram.

1. Perform a STFT with specific values for *window-type, window-length, FFT-size*, and *hop-size*,

$$X_l(k) \triangleq \sum_{n=0}^{N-1} w(n)x(n+lH)e^{-j\omega_k n}, \qquad l = 0, 1, 2, \ldots \tag{3.27}$$

where $w(n)$ is the analysis window, l the frame number, and H the *hop-size*. The result is a series of complex spectra.

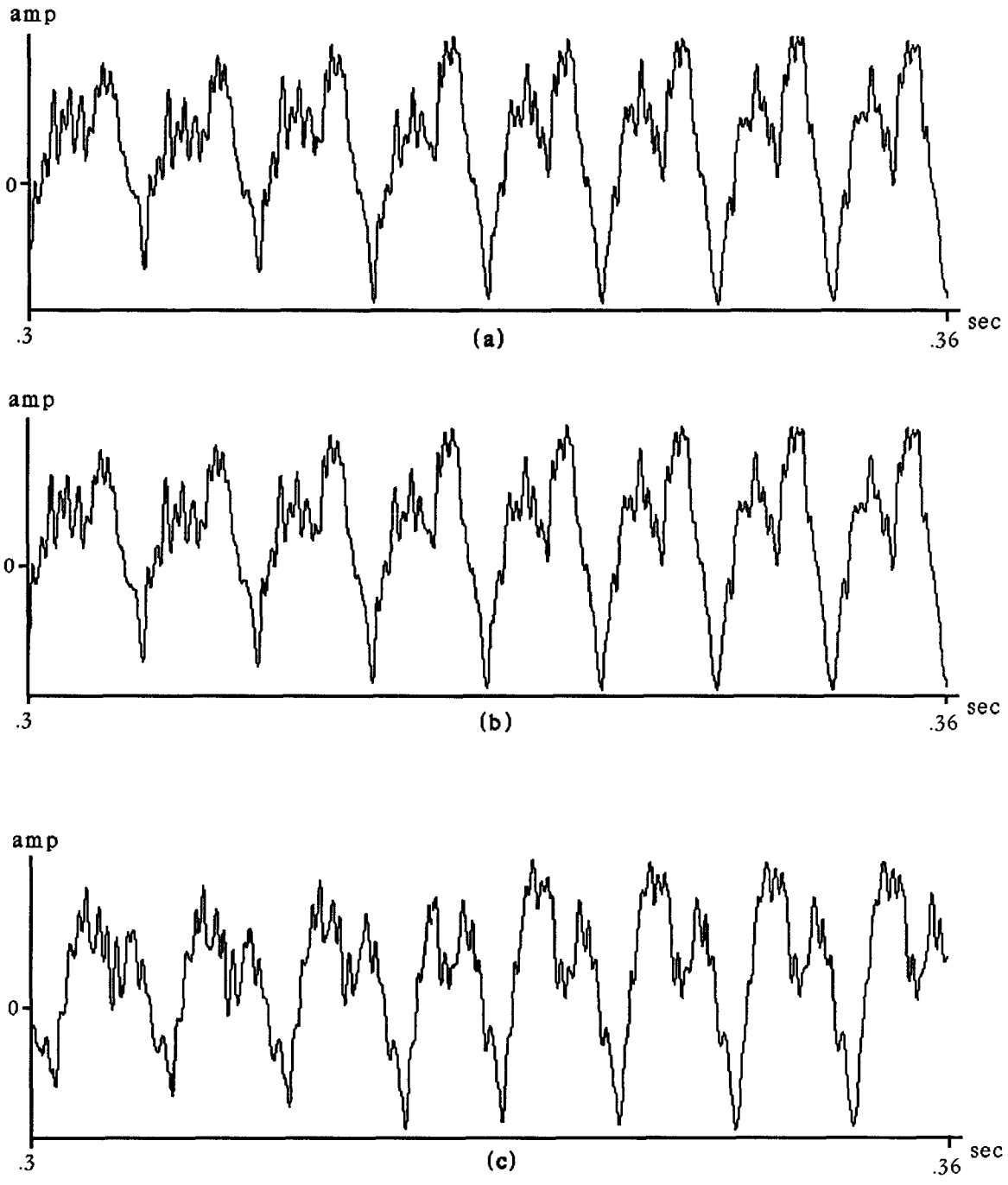

Figure 3.7: Sinusoidal synthesis example: (a) original cello sound, (b) synthesis using phase information, (c) synthesis without phase information.

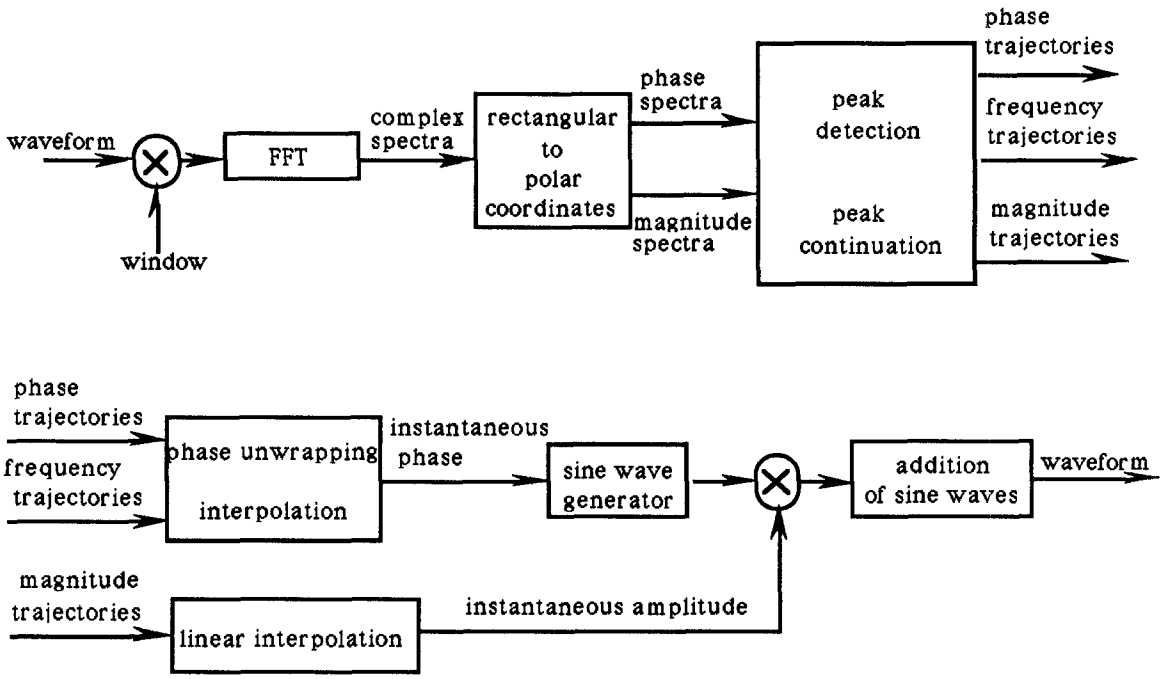

Figure 3.8: Block diagram of the sinusoidal system.

2. Convert to polar coordinates,

$$\begin{aligned} A_l(k) &\triangleq |X_l(k)| \\ \Theta_l(k) &\triangleq \angle X_l(k) \qquad \text{(radians)} \end{aligned} \tag{3.28}$$

3. Convert each magnitude spectrum to dB magnitude,

$$\hat{X}_l(k) = 20\log_{10} A_l(k) \tag{3.29}$$

4. Find prominent spectral peaks by using the peak detection algorithm, given the *minimum-peak-height* in dB, and the frequency and amplitude ranges.

5. Perform a parabolic interpolation to refine the peak location (frequency), the peak height in the magnitude spectra (amplitude), and the phase value. This returns amplitude, frequency, and phase estimates of the form $(\hat{A},\hat{\omega},\hat{\varphi})$.

6. Assign each peak to a frequency trajectory by matching the peaks of the previous frame with the current one. These trajectories are "born," or "killed" at any frame by ramping the amplitude from or toward 0.

7. Apply any desired modification to the analysis parameters before resynthesis.

8. Generate a sine wave for each frequency trajectory, and sum them all,

$$s^l(m) = \sum_{r=1}^{R^l} \hat{A}_r^l(m) \cos[\hat{\theta}_r^l(m)] \tag{3.30}$$

The instantaneous amplitude, and phase for each sine wave are calculated by interpolating the values from frame to frame. The length of the synthesis frame is equal to the hop size H (unless time expansion or compression is desired), which is typically some fraction of the window length M.

3.11 Examples

The sinusoidal analysis/synthesis system is more flexible than the STFT. The following two examples show some of the possibilities of the sinusoidal representation, first on a complex musical excerpt and then on a more simple one.

3.11.1 Sound example 2

Excerpt from "El Amor Brujo" by Manuel de Falla. (*sampling-rate* = 34000, length = 6.8 sec.)

Analysis parameters: *window-type* = Kaiser (β = 2), *window-length* = 1601 samples (.047 sec.), *FFT-size* = 2048 samples, *hop-size* = 400 samples (.012 sec.), *local-dB-range* = 75dB, *general-dB-range* = 85dB, *minimum-peak-height* = .5dB, *frequency-range* = 30Hz–16KHz, *maximum-peak-deviation* = 80Hz, *number-trajectories* = 250.

1. original sound
2. synthesis with phase
3. synthesis without phase
4. synthesis with time expansion by factor of 1.68
5. synthesis with frequency transposition by factor of 1.4
6. synthesis with frequency transposition by factor of .8

The synthesis has some modulation which is the result of not tracking enough peaks (only 250). For a higher quality version many more trajectories are required.

The difference between the synthesis with phase and the one without phase is minimal. It is more noticeable with sounds with a prominent noise component.

The sound transformations presented are quite successful, however bigger stretches or more pronounced frequency transpositions result in noticeable problems. The most common one is that the component sine waves do not fuse together.

3.11.2 Sound example 3

Guitar passage. (*sampling-rate* = 34000, length = 7.14 sec.)

Analysis parameters: *window-type* = Kaiser (β = 2.5), *window-length* = 801 samples (.024 sec.), *FFT-size* = 2048 samples, *hop-size* = 200 samples (.0059 sec.), *local-dB-range* = 70dB, *general-dB-range* = 75dB, *minimum-peak-height* = .5dB, *frequency-range* = 30Hz–16KHz, *maximum-peak-deviation* = 80Hz, *number-trajectories* = 150.

1. original sound
2. synthesis with phase tracking
3. synthesis without phase tracking
4. synthesis with time expansion by a factor of 1.45

Due to the simplicity of the sound, compared with the previous example, the synthesis is successful with only 150 sinusoids. However the attacks of the guitar sound are very sensitive to transformation and very easily the noise component present in it acquires a tonal quality.

3.12 Conclusions

In this chapter, an analysis/synthesis system based on a sinusoidal model has been presented. The resulting representation is characterized by the amplitudes, frequencies, and phases of the component sine waves. This system is more flexible than the one presented in Chapter 2 and a wider variety of sound transformations can be performed. However it is still not ideal, especially for sounds with noise components. In the next chapter, a modification to the sinusoidal system is made in order to accommodate noise.

Chapter 4

A Deterministic plus Residual Model

4.1 Introduction

The sinusoidal model presented in the previous chapter is not appropriate for the manipulation of sounds that contain noise components. Noise does not lend itself easily to a sinusoidal representation. For example, the breathy part of a flute sound, the attack of a drum stroke, or the sound of rain, are not well represented by sinusoids. In this chapter an alternative model is introduced which considers a sound to be composed of a deterministic part plus a residual.

A deterministic signal is traditionally defined as anything that is not noise (i.e., a perfectly predictable part, predictable from measurements over any continuous interval). However in the present discussion the class of deterministic signals considered is restricted to sums of quasi-sinusoidal components (sines with piecewise linear amplitude and frequency variation). Each sinusoid models a quasi sinusoidal component of the original sound and it is an independent element which can be synthesized by itself. By contrast, in the sinusoidal model, each sinusoid modeled a peak in the spectrum (not always a sinusoidal component) and it was only the sum of all sinusoids that made any sense. In more musical terms the deterministic component models the partials[1] of the sound, not just any energy. The residual is then defined as the difference between the original and the estimated deterministic part. In musical instruments this residual generally comprises the energy produced by the excitation mechanism (e.g., the bow in a string instrument) that is not transformed by the resonating body into stationary vibrations, plus any other energy component that is not sinusoidal in nature. The sum of the two components results in the original sound.

This decomposition technique does not allow extensive modifications of the representation. While the deterministic component can be modified easily, the residual (a waveform) is difficult to transform. In this respect it is a step backwards in attaining the goal of this dissertation. However, this technique is an intermediate step en route to the next

[1] A partial is a sinusoidal component of a sound that usually corresponds to a mode of vibration of the producing sound system.

chapter, where, by making some assumptions about both the deterministic and residual components, a more flexible representation is obtained. Nevertheless, it will be shown how this decomposition method is useful in itself for a variety of applications.

In this chapter the decomposition system is elaborated. First, the deterministic plus residual model, the basis for the technique, is presented, followed by a general description of the system. The next sections include a detailed discussion of the steps carried out by the decomposition process. The chapter ends with a summary of these different steps, some examples, and a conclusion.

4.2 The Deterministic plus Residual Model

A sound model assumes certain characteristics of the waveform or of the sound generation mechanism. In general, every analysis/synthesis technique has an underlying model. For example, many speech analysis/synthesis techniques are based on what is known as the speech production model (Rabiner and Schafer, 1978). Such a model assumes that a speech waveform $s(t)$ is the result of passing a glottal excitation waveform $e(t)$ through a linear time-varying filter $h(t,\sigma)$ that models the characteristics of the vocal tract. If the time-varying impulse response of the vocal tract filter is $h(t,\sigma)$, then

$$s(t) = \int_0^t h(t, t-\tau)e(\tau)d\tau \tag{4.1}$$

Thus, the speech waveform is the convolution of the filter $h(t,\sigma)$ with the excitation $e(t)$.

In the current system, the deterministic-plus-residual model considers a waveform $s(t)$ as the sum of a series of sinusoids plus a residual $e(t)$,

$$s(t) = \sum_{r=1}^{R} A_r(t)\cos[\theta_r(t)] + e(t) \tag{4.2}$$

where R is the number of sinusoids, $A_r(t)$ is the instantaneous amplitude and $\theta_r(t)$ the instantaneous phase. The residual is the difference between the signal and the deterministic part. This deterministic component is defined in the same way as in the sinusoidal model, thus the instantaneous phase $\theta(t)$ is defined (as in equation 3.2) by

$$\theta_r(t) = \int_0^t \omega_r(\tau)d\tau + \theta_r(0) + \phi_r \tag{4.3}$$

where $\omega_r(t)$ is the instantaneous radian frequency, $\theta_r(0)$ is the initial phase value, and ϕ_r is the fixed phase offset. The only differences between the deterministic and sinusoidal model are that: (1) we now keep a residual signal, and (2) now the sinusoids are restricted to be stable (i.e., follow stable quasi-sinusoidal components), thus they model only the partials

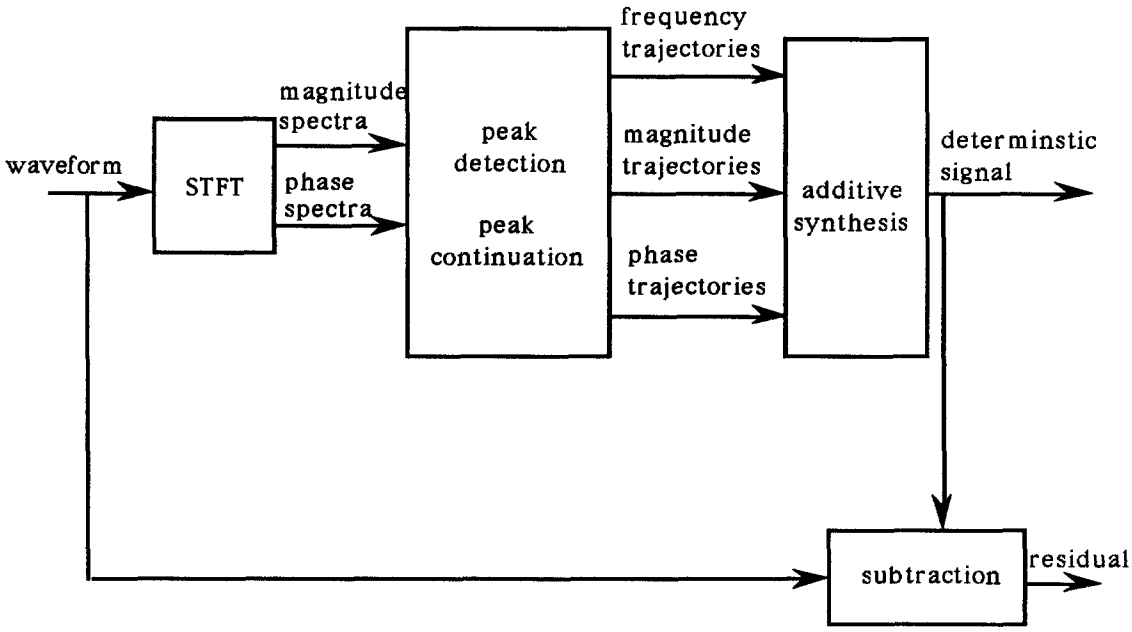

Figure 4.1: General diagram of the deterministic plus residual system.

of the sound. In the sinusoidal model, spectral peaks need not form meaningful long-term trajectories.

The deterministic part is now more constrained, but the new residual signal actually increases the generality of the model with respect to the sinusoidal one.

4.3 General Description of the System

We use the deterministic-plus-residual model to develop a system that decomposes a sound into a sum of sinusoids plus a residual. A general diagram of the method is shown in Fig. 4.1. The process starts by computing a set of magnitude and phase spectra from a sound with the STFT. From these spectra, sinusoidal trajectories are found with peak detection and peak continuation algorithms. The behavior of the peak trajectories is more restrained than in the system of Chapter 3 in order to accommodate the "deterministic signal" concept, and a new peak continuation algorithm is presented in this chapter for this purpose. The sinusoids are then generated with additive synthesis, and the residual is calculated simply by subtracting the deterministic signal from the original waveform.

In the next sections, a detailed discussion of each step is presented.

4.4 Computation of the Magnitude and Phase Spectra

The computation of the magnitude and phase spectra is the first step in the decomposition technique. It is in the spectrum that the sinusoids are tracked and the decision takes place as to whether a part of the signal is considered deterministic or residual. The computation of the spectra is carried out by the STFT using the steps and criteria presented in Chapter 2. It is important to set the parameters (*window-length, window-type, FFT-size,* and *hop-size*) in accordance with the sound to be processed.

There are two points to be considered when setting the STFT parameters. The first is that a good resolution of the sinusoidal peaks is desired, more so than in the sinusoidal system of Chapter 3, because the process that extracts the sinusoids must isolate the peaks which correspond to the deterministic component. The second point is that the phase information is particularly important in this decomposition technique. Therefore (as discussed in Chapter 2) the STFT should use an odd-length analysis window and the windowed data should be centered in the FFT-buffer at the origin in order to obtain a constant-phase spectrum.

The result of this computation is a complex spectrum of the form $X_l(k)$, where l is the frame number and k is the frequency-bin index in the spectrum, for every frame. Then, by changing the coordinates from rectangular to polar, the magnitude and phase spectra are obtained.

4.5 Spectral Peak Detection

Once the set of magnitude and phase spectra are computed, the next step is to find the prominent peaks in every spectrum. The process is the same as for the sinusoidal system.

The restriction added to the sinusoids of this model, i.e., that they be partials of the sound, does not change the peak detection algorithm. Theoretically, a partial (a sinusoid) that is stable both in amplitude and in frequency has a well defined frequency representation, the transform of the analysis window used to compute the Fourier transform. It should be possible to take advantage of this characteristic to distinguish partials from other frequency components. However, in practice this is rarely the case since most natural sounds are not perfectly periodic and do not have nicely spaced and clearly defined peaks in the frequency domain. There are interactions between the different components and the shape of the spectral peaks cannot be used in the peak detection process. Only some instrumental sounds (e.g., the steady-state part of a violin or oboe sound) are periodic enough and sufficiently free from prominent noise components that the frequency representation of a stable sinusoid could be used in the peak detection.

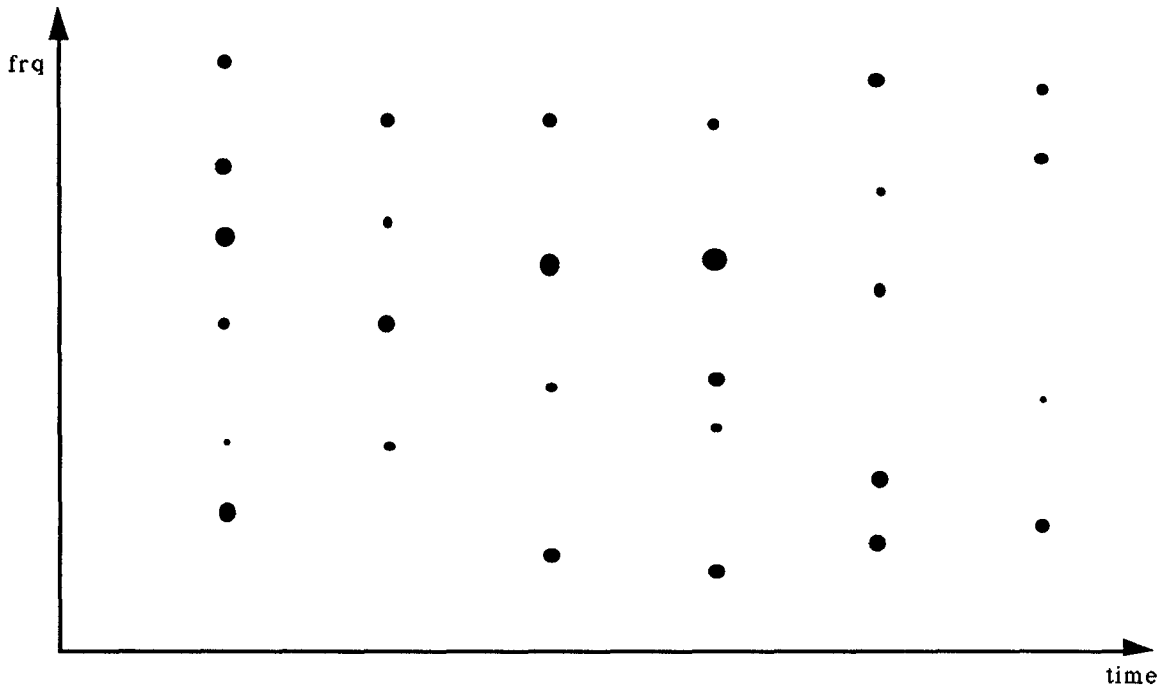

Figure 4.2: Example of a surface of spectral peaks. The size of each dot represents the amplitude of the peak.

A practical solution is to detect as many peaks as possible and delay the decision of deterministic, or "well behaved partial," to the next step in the system, the peak continuation algorithm. Nonetheless, some simple control of the peak detection process is achieved through the *minimum-peak-height* parameter and the specification of the magnitude and frequency ranges where the search for peaks takes place, thus rejecting areas of the magnitude spectrum which are known to have irrelevant partials. This results in fewer peaks, thus making the peak continuation process easier and faster.

4.6 Spectral Peak Continuation

Once the spectral peaks have been detected, a subset of them is organized by the peak continuation algorithm into peak trajectories. The design of such an algorithm can be approached as a line detection problem, where out of a surface of discrete points, each one being a peak (as shown in Fig. 4.2), the algorithm finds lines according to the characteristics imposed by the model.

Approaches to this type of problem are found in the image processing literature under line and edge detection (Rosenfeld, 1969; Tou and Gonzales, 1974; Wahl, 1987), and in the

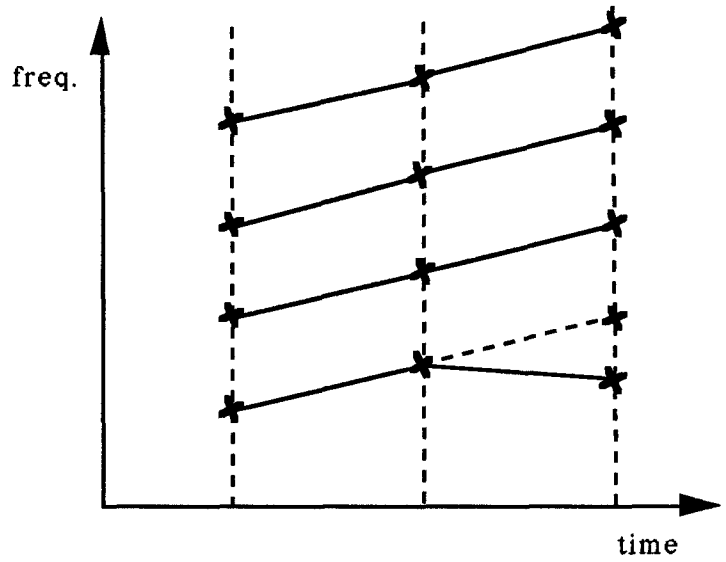

Figure 4.3: Tracking error of the peak continuation algorithm presented in Chapter 3. The dotted line is the actual continuation of the partial which the algorithm is not able to follow.

estimation and detection theory literature under extraction of sine waves from Gaussian noise (Van Trees, 1968; Wolcin, 1980). However, the algorithms found on both types of literature are difficult to apply to this particular situation. The alternative has been to adapt the sequential scheme of the peak continuation algorithm of Chapter 3 to the current model.

The algorithm presented in Chapter 3 finds peak trajectories both in the noise and deterministic parts of a waveform, thus obtaining a sinusoidal representation for the whole sound. That algorithm is unsuitable when it is desired that each partial be extracted by a single trajectory; it is unlikely for a trajectory to follow a particular partial. The trajectories are easily turned on and off whenever the peaks pertaining to a partial are not perfectly linearly aligned. For example, when the partials change in frequency substantially from frame to frame, the algorithm easily switches from the partial that it was tracking to another one which at that point is closer. Figure 4.3 shows a typical example of this problem. These tracking errors are responsible in part for the distortions created in the synthesis when some transformations are done to the sinusoidal representation.

In this chapter the objective is to design an algorithm such that: (1) only the clear and stable partials of a sound are tracked, and (2) each partial belongs to a single trajectory.

4.6.1 Description of the peak continuation algorithm

The input to the algorithm is the set of peaks returned by the peak detection process. The output is a subset of the same peaks ordered into trajectories, where each trajectory represents a stable sinusoid.

The algorithm is intended for a variety of sounds. The behavior of a partial, and therefore the way to track it, varies depending on the sound. Whether it is speech, an instrumental sound with a harmonic spectrum, a sound of a gong, a sound of an animal, or any other, the time evolution of the component partials will vary. Thus, the algorithm, apart from being general, requires some knowledge about the characteristics of the sound that is being analyzed. In the current algorithm there is no attempt to make the process completely automatic. The user is expected to know some of the characteristics of the sound beforehand, specifying them through a set of parameters.

The basic idea of the algorithm is that a set of *frequency guides* advances in time through the spectral peaks, looking for the appropriate ones (according to the specified constraints) and forming trajectories out of them.[2] The instantaneous state of the guides, their frequency, is kept in $\tilde{f}_1, \tilde{f}_2, \tilde{f}_3, \ldots, \tilde{f}_p$, where p is the number of existing guides. These values are continuously updated as the guides are turned on, advanced, and finally turned off.

Before explaining in detail the algorithm let us describe the control parameters available to the user.

- *initial-guides.* With this parameter the user specifies the approximate frequency of partials which are known to be present in the sound, thus reserving guides for them. The algorithm adds new guides to this initial set as it finds them. When no initial guides are specified, the algorithm creates all of them.
- *maximum-peak-deviation.* Guides advance through the sound selecting peaks. With this parameter there is a control on the maximum allowable frequency distance from a peak to the guide that it is selected by. It is useful to make this parameter a function of frequency in such a way that the allowable distance is bigger for higher frequencies than for lower ones. Thus, the deviation can follow a log-scale, which is perceptually more meaningful than a linear frequency scale.

[2]It is important to distinguish between guides and trajectories. A guide is an abstract entity which is used by the algorithm to create the peak trajectories. The trajectories are the actual result of the peak continuation algorithm.

- *peak-contribution-to-guide.* The frequency of each guide does not have to correspond to the frequency of the actual trajectory. It is updated every time it incorporates a new peak. This parameter is a number from 0 to 1 that controls how much the guide frequency changes when a new peak is incorporated. That is, given that the current guide has a frequency $\tilde{f}$, what will be its value when it incorporates a peak with frequency h. For example, if the value of the parameter is 1 it means that the value of the guide, $\tilde{f}$, is updated to h, thus the peak makes the maximum contribution. If the value of the parameter is smaller, the contribution of the peak is correspondingly smaller; the new value falls between the current value of $\tilde{f}$ and h. This parameter is useful, for example, to circumscribe a guide to a narrow frequency band (done by specifying a number close to 0).

- *maximum-number-of-guides.* This is the maximum number of guides used by the peak continuation process at each particular moment in time. The total number of guides may be bigger because when a guide is turned off a new one can use its place.

- *minimum-starting-guide-separation.* A new guide can be created at any frame from a peak which has not yet been incorporated into any existing guide. This parameter specifies the minimum required frequency separation from a peak to the existing guides in order to create a new guide at that peak. Consequently, through this parameter peaks which are very close to existing guides can be rejected as candidates for starting guides.

- *maximum-sleeping-time.* When a guide has not found a continuation peak for a certain number of frames the guide is killed. This parameter specifies the maximum "non active" time, that is, the maximum number of frames that the guide can be alive while not finding continuation peaks. If its value is 0 the guide is killed as soon as it does not find a continuation peak.

- *maximum-length-of-filled-gaps.* Given that a certain sleeping time is allowed, we may wish to fill the resulting gaps. This parameter specifies the length of the biggest gap to be filled (a number smaller or equal than *maximum-sleeping-time*). The gaps are filled by interpolating between the end points in the trajectory.

- *minimum-trajectory-length.* Once all the trajectories are created, this parameter controls the minimum trajectory length. All trajectories shorter than this length are deleted.

To describe the peak continuation algorithm let us assume that the frequency guides were initialized with *initial-guides* and that they are currently at frame n. Suppose that the guide frequencies at the current frame are $\tilde{f}_1, \tilde{f}_2, \tilde{f}_3, \ldots, \tilde{f}_p$,[3] where p is the number

[3]These frequencies may not be the frequencies of the actual trajectories; this depends on the value of *peak-contribution-to-guide.*

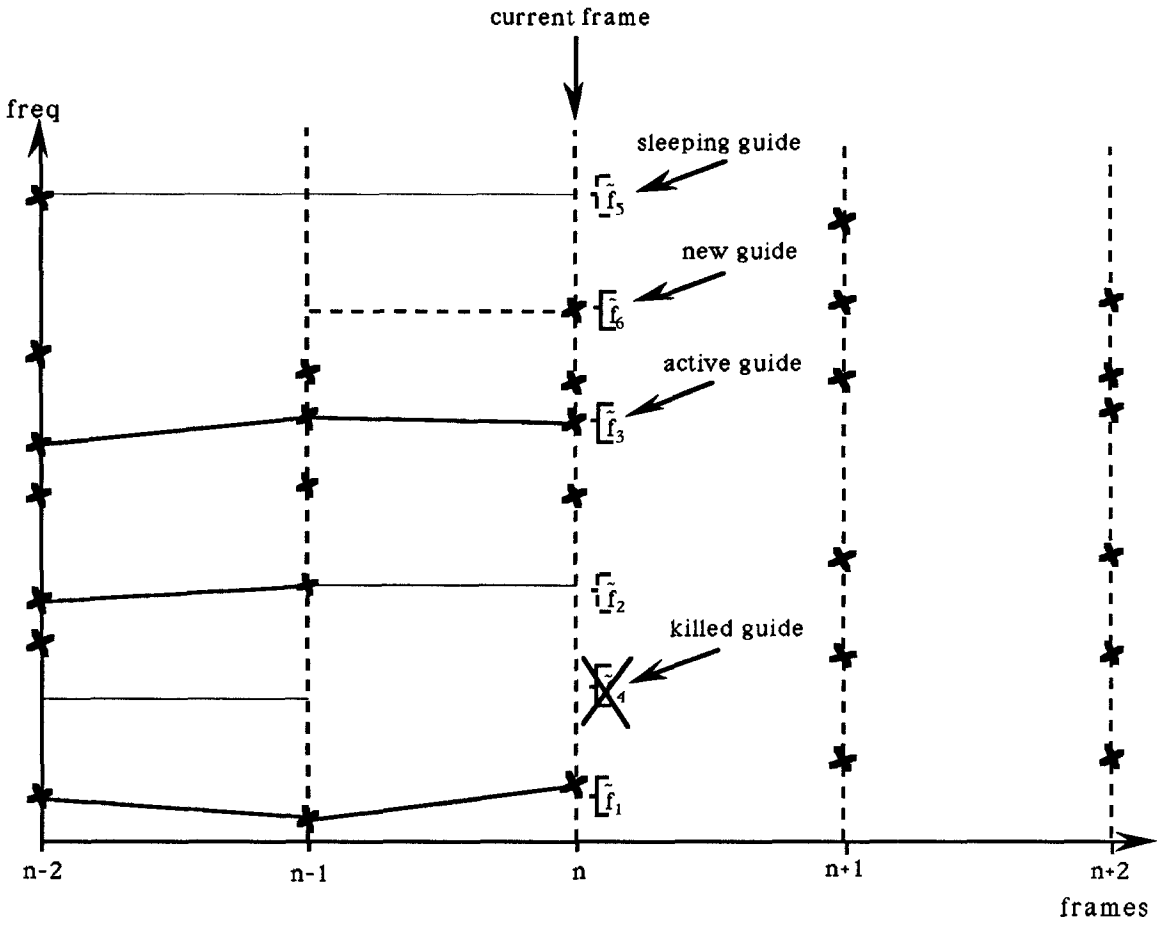

Figure 4.4: Illustration of the peak continuation algorithm. The *frequency-guides* have just been continued through frame n.

of existing guides. We want to continue the p guides through the peaks of frame n with frequencies $g_1, g_2, g_3, \ldots, g_m$, thus continuing the corresponding trajectories. Figure 4.4 illustrates the algorithm.

There are three steps in the algorithm: (1) guide advancement, (2) update of the guide values, and (3) start of new guides. Next, these steps are described.

4.6.1.1 Guide advancement

Each guide is advanced through frame n by finding the peak closest in frequency to its current value. The rth guide claims frequency g_i for which $|\tilde{f}_r - g_i|$ is a minimum. The

change in frequency must be less than *maximum-peak-deviation.* The possible situations are as follows:

1. If a match is found within the maximum deviation, the guide is continued (unless there is a conflict to resolve, as described below). The selected peak is incorporated into the corresponding trajectory.

2. If no match is found, it is assumed that the corresponding trajectory must "turn off" entering frame n, and its current frequency is matched to itself with zero magnitude. Since the trajectory amplitudes are linearly ramped from one frame to the next, the terminating trajectory ramps to zero over the duration of one hop size. Whether the actual guide is "killed" or not, depends on the *maximum-sleeping-time.*

3. If a guide finds a match which has already been claimed by another guide, we give the peak to the guide that is closest in frequency, and the "loser" looks for another match. If the current guide loses the conflict, it simply picks the best available non-conflicting peak which is within the *maximum-peak-deviation.* If the current guide wins the conflict, it calls the assignment procedure recursively on behalf of the dislodged guide. When the dislodged guide finds the same peak and wants to claim it, it sees there is a conflict which it loses and moves on. This process is repeated for each guide, solving conflicts recursively, until all possible matches are made.

4.6.1.2 Update of the guide values

Once all the existing guides and their trajectories have been continued through frame n, the guide frequencies are updated. There are two possible situations:

1. If a guide finds a continuation peak, its frequency is updated from $\tilde{f}_r$ to $\tilde{h}_r$ according to:

$$\tilde{h}_r = \alpha(g_i - \tilde{f}_r) + \tilde{f}_r, \qquad \alpha\epsilon[0,1] \tag{4.4}$$

   where g_i is the frequency of the peak that the guide has found at frame n, and α is the *peak-contribution-to-guide.* When α is 1 the frequency of the peak trajectory is the same than the frequency of the guide, therefore the difference between guide and trajectory is lost.

2. If a guide does not find a continuation peak for *maximum-sleeping-time* frames, the guide is killed at frame n. If it is still under the *sleeping-time* it keeps the same value (its value can be negated in order to remember that it has not found a peak). When *maximum-sleeping-time* is 0 any guide that does not find a continuation peak at frame n is killed. In order to distinguish between guides that find a continuation peak from the ones that do not but still are alive, we refer to the first ones as *active guides* and the second ones as *sleeping guides.*

4.6.1.3 Start of new guides

New guides, and therefore new trajectories, are created from the peaks of frame n that are not incorporated into trajectories by the existing guides. If the number of current guides is smaller than *maximum-number-of-guides* a new guide can be started.

A guide is created at frame n by searching through the "unclaimed" peaks of the frame for the one with the highest magnitude which is separated from every existing guide by at least *minimum-starting-guide-separation.* The frequency value of the selected peak is the frequency of the new guide. The actual trajectory is started in the previous frame, $n-1$, where its amplitude value is set to 0 and its frequency value to the same as the current frequency, thus ramping in amplitude to the current frame. This process is recursively done until there are no more unclaimed peaks in the current frame, or the number of guides has reached *maximum-number-of-guides.*

In order to minimize the creation of guides with little chance of surviving (thus interfering with more consolidated guides in the peak continuation process), a temporary buffer is used for the starting guides. The peaks selected to start a trajectory are stored into this buffer and continued by only using peaks that have not been taken by the "consolidated" guides. Once these temporary guides have reached a certain length they become "normal guides."

The attack of most sounds is quite "noisy," and the search for partials is harder in such a rich spectrum. A useful modification to the algorithm is to start the process from the end of the sound, that is, to start tracking the peaks from the last frame and work towards the front. The tracking process encounters the end of the sound first, and since this is a very stable part in most instrumental sounds, the algorithm finds a very clear definition of the partials. When the guides arrive at the attack, they are already tracking the main partials and can reject non-relevant peaks appropriately, or at least evaluate them with some acquired knowledge.

4.6.1.4 Sound composed of several events

When the sound to be analyzed includes several events (or notes), with different characteristics and non-overlapped in time, it is useful to specify the boundaries of these events so that each one can be treated as a separate unit by the peak continuation algorithm. The guides are then reset at the beginning of each event and new values for the control parameters are specified.

Unless the variation between events is very pronounced in terms of frequency components, it is unnecessary to perform a time varying STFT analysis and peak detection algorithm. It is sufficient to reinitialize the guides of the peak continuation process.

4.6.2 Variations on the algorithm for harmonic sounds

When the sound to be analyzed is known to be harmonic, the peak continuation algorithm can be restricted even more. Now the *frequency guides* are circumscribed to track harmonic frequencies. There are three major changes with respect to the general algorithm, (1) there is a specific fundamental frequency at every frame, (2) the number of guides remain constant throughout the sound, and (3) each guide tracks a specific harmonic number. For these changes the following parameters are added to the algorithm,

- *initial-fundamental.* This is the approximate fundamental frequency at the beginning of the sound.
- *fundamental-range.* With this parameter the user specifies the approximate maximum and minimum values reached by the fundamental frequency throughout the sound.
- *maximum-fundamental-deviation.* This parameter controls the maximum allowable deviation in the fundamental frequency from frame to frame.

In this version of the peak continuation algorithm the *frequency guides* are initialized with the harmonic series of the *initial-fundamental.*[4] Unless *initial-fundamental* is given the algorithm does not create trajectories until it finds a fundamental frequency and it is able to initialize the guides. The number of guides (i.e., number of harmonics) is constant throughout the sound, set by *maximum-number-of-guides.* Therefore, the guides are not started or killed as in the general algorithm.

There are two parameters used in the general algorithm which are unnecessary in the harmonic case. These are the *initial-guides* and the *minimum-starting-guide-separation.*

The steps involved in this new algorithm are slightly different. At the current frame the succession of steps is now as follow: (1) detection of fundamental, (2) guide-advancement, and (3) update of the guide values.

The only new step is the detection of the fundamental, the other two have already been discussed for the general case.

---

[4] A harmonic series is formed by taking the fundamental frequency and the integer multiples of it.

4.6.2.1 Detection of fundamental

Before advancing the guides through frame n a pitch detection algorithm searches for the fundamental frequency of frame n. If this is found the guide values are reset to the harmonic series of the new fundamental. If the fundamental is not found the guides keep the frequency values that resulted from the previous frame.

Given the set of peaks of frame n, with magnitude and frequency values for each one, there are many possible fundamental detection strategies (Schroeder, 1968; Piszczalski and Galler, 1979; Terhardt, 1982b; Hess, 1983; Amuedo, 1985). In the current application we are dealing with single-source sounds and assuming that a fundamental peak exists.[5] With these restrictions, a simple algorithm is designed that suffices for this situation. It is based on finding the 3 highest peaks at frame n and then searching for the peak which is a fundamental for the three of them.[6] By choosing the highest peaks it is assured that they are "good" harmonic partials, therefore they are multiples of a fundamental. The search for the fundamental is done through the peaks which are within the *fundamental-range* and not further than *maximum-fundamental-deviation* from the previous fundamental.

If a clear fundamental is found, the guides are set to its harmonic series, otherwise their values are not changed. Then the guides are continued through frame n by the process already described. Finally the guide values are updated in case a clear fundamental is not found at frame $n + 1$ (in which case these new values are used).

This algorithm is succesful with many instrumental sounds and with speech.[7] However there are some cases for which the general algorithm is more adequate. This is the case with sounds that include superposed notes (i.e., one note rings through another one), or not very well defined partials (e.g., noisy recording), or with sounds that are not quite harmonic (e.g., the piano with its stretch partials).

---

[5]Most pitch detection algorithms do not have this restriction.

[6]The choice of three peaks has been a practical one and any other number is also possible.

[7]In the case of speech a further improvement can be made to assure that the algorithm does turn off the trajectories through unvoiced portions of the sound. This is done by simply not continuing trajectories when a fundamental is not found in the current frame.

4.6.3 Conclusions on the peak continuation algorithm

The algorithm presented is only one approach to the peak continuation problem. The creation of trajectories out of the spectral peaks is compatible with very different strategies and algorithms. For example, we might design algorithms to track partials in piano sounds, or in polyphonic music, or to separate multiple source sounds. In fact, the peak continuation algorithm is the part of this dissertation that is the most open for further work.

A computer implementation of this analysis/synthesis system should allow for extensions of the peak continuation algorithm, and even more, for the inclusion of other peak tracking techniques as part of the system.

4.7 Deterministic Synthesis

The synthesis of the sinusoidal part of the sound proceeds in the same way as in the previous chapter. The analysis returns a set of amplitudes $\hat{A}^l$, frequencies $\hat{\omega}^l$, and phases $\hat{\varphi}^l$ for each frame l, with a "triad" $(\hat{A}^l_r, \hat{\omega}^l_r, \hat{\varphi}^l_r)$ for each trajectory r (i.e., identical to the previous chapter). From these trajectories the synthesis process computes one frame of the deterministic component $d(n)$ by

$$d^l(m) = \sum_{r=1}^{R^l} \hat{A}^l_r \cos[m\hat{\omega}^l_r + \hat{\varphi}^l_r], \qquad m = 0, 1, 2, \ldots, S-1 \tag{4.5}$$

where R^l is the number of trajectories present at frame l and S is the length of the synthesis buffer. The final sound is the result of juxtaposing all the frames. In Chapter 3 it was shown how to interpolate the analysis values from frame to frame in order to obtain the instantaneous amplitudes and phases.

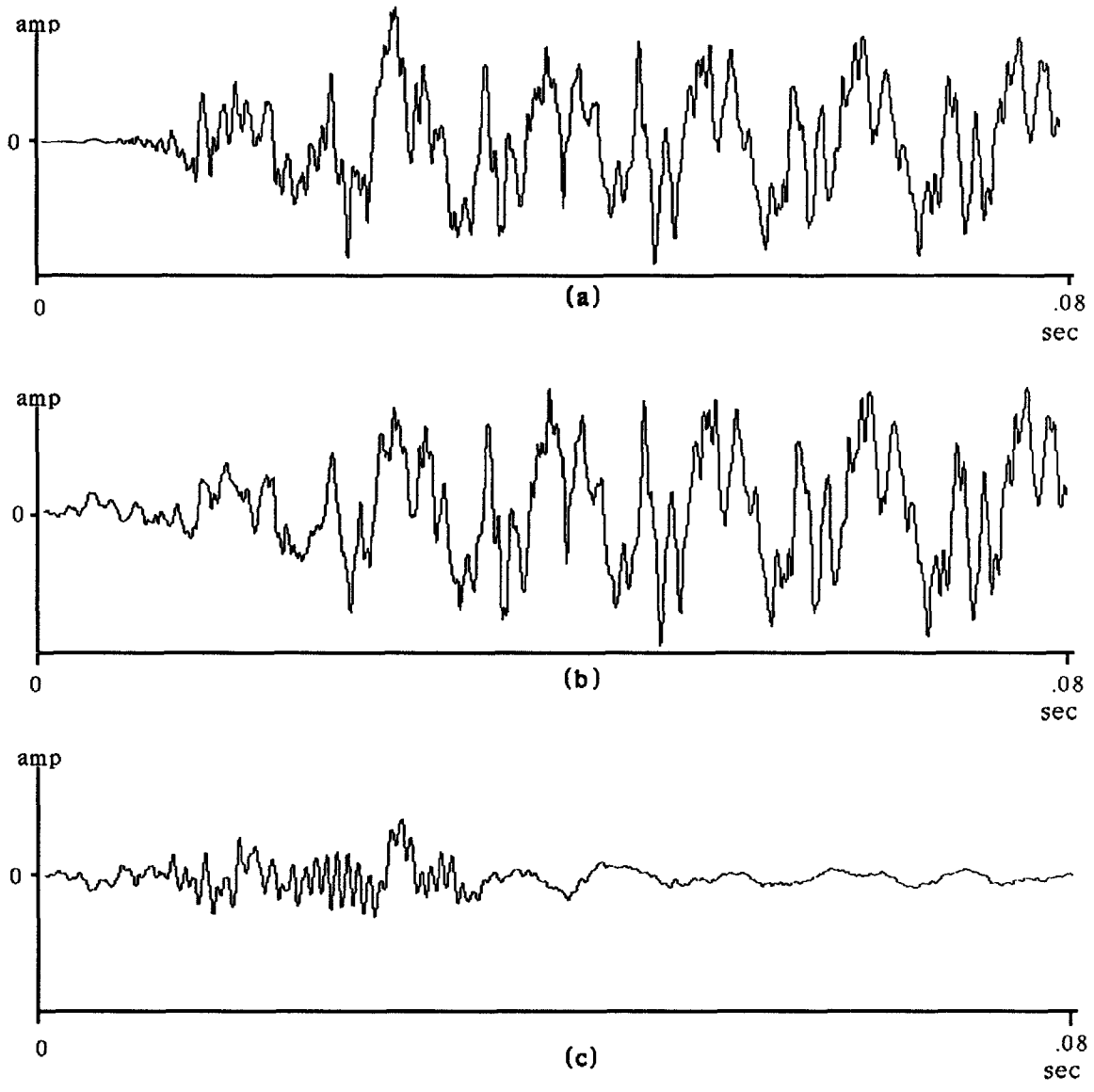

Figure 4.5: Decomposition example: (a) attack of a piano tone, (b) deterministic component, (c) residual.

4.8 Computation of the Residual

The waveform returned by the deterministic synthesis reproduces the instantaneous phase and amplitude of the partials of the original sound. Thus, it is possible to subtract the synthesized sinusoids from the original sound and obtain a meaningful residual. Figure 4.5 shows an example with a piano tone.

As can be seen in Figure 4.5 the attack of both the deterministic component and the residual are smeared in comparison with the original waveform. The reason for this is that the sharpest attack possible on the deterministic component is determined by the *hop-size* of the analysis window (since the amplitudes are ramped from frame to frame), and clearly the attack of the piano note is sharper than any reasonable *hop-size*. In most cases this is perceptually irrelevant. However if a more accurate attack is desired in the separate components, an even smaller *hop-size* may be specified. But if a more real attack is desired only in the residual part, the solution is to obtain the residual by the following subtraction

$$e(n) = \begin{cases} \max[x(n), d(n)] - d(n), & x(n) \geq 0 \\ \min[x(n), d(n)] - d(n), & x(n) < 0 \end{cases} \tag{4.5}$$

where $e(n)$ is the residual, $x(n)$ the original waveform, and $d(n)$ the deterministic component. Such a subtraction prevents the residual from having a bigger (positive or negative) amplitude than the original, and thus the attacks of the original sound are preserved.

4.8.1 Residual from magnitude-only analysis/synthesis

It is also possible to obtain a residual when the phase of the original sound is not preserved in the deterministic synthesis by performing the subtraction in the frequency domain. The identity property of the system is then lost, but perceptually the result is identical to the decomposition already presented.

A diagram of this alternative decomposition technique is shown in Figure 4.6. In this case the deterministic component does not maintain the phase of the original waveform, therefore the subtraction is done on the magnitude spectrum, since the frequency and magnitude of each partial has been preserved. Given that the magnitude spectrum of the original sound at frame l is $|X_l(k)|$ and that of the deterministic signal $|D_l(k)|$, the residual is

$$|E_l(k)| = |X_l(k)| - |D_l(k)| \tag{4.6}$$

The residual waveform results from an inverse-STFT. This is performed by using the magnitude-spectrum residuals and the set of phase spectra of the original sound (instead, the phase spectra of the deterministic component can also be used).

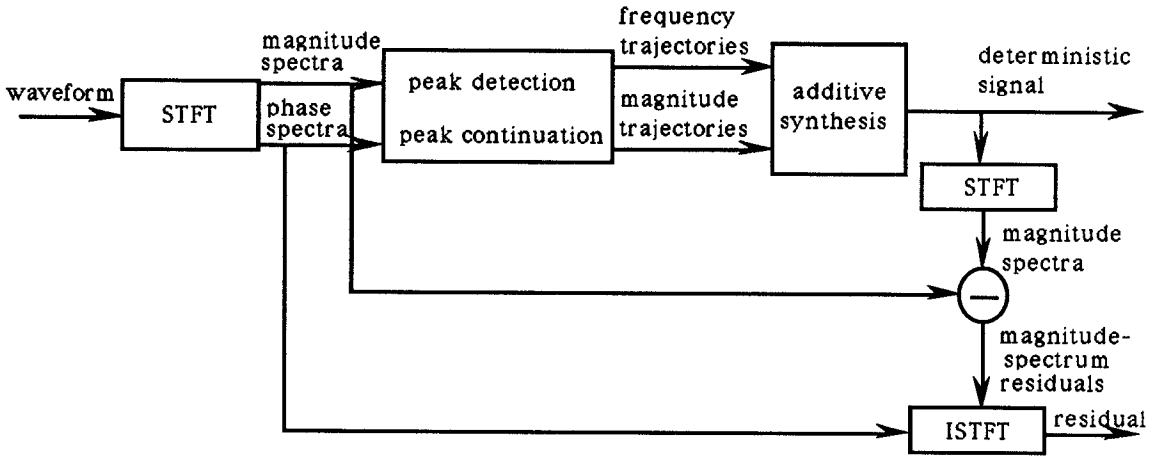

Figure 4.6: General diagram of an alternative deterministic plus residual decomposition for a magnitude-only analysis/synthesis system.

Since the phase tracking, and specially the additive synthesis using phase, are computationally quite expensive, this alternative is a valuable simplification of the decomposition system.

4.9 Summary of the Technique

A review of the main steps for a computer implementation of the decomposition technique is given below.

1. Perform a STFT with specific values for *window-type, window-length, FFT-size*, and *hop-size*,

$$X_l(k) \triangleq \sum_{n=0}^{N-1} w(n)x(n+lH)e^{-j\omega_k n}, \qquad l = 0, 1, 2, \ldots \tag{4.7}$$

   where $w(m)$ is the analysis window, H the *hop-size*, and l the frame number. The result is a series of complex spectra that are then converted to polar coordinates.

2. Detect the prominent spectral peaks in specific frequency and magnitude ranges using the peak-detection algorithm. The result is a set of amplitudes $\hat{A}^l$, frequencies $\hat{\omega}^l$, and phases $\hat{\varphi}^l$ for each frame l.

3. Create peak trajectories with the peak-continuation algorithm. Now, the values for each peak $(\hat{A}_r^l, \hat{\omega}_r^l, \hat{\varphi}_r^l)$ belong to a specific trajectory r.

4. Synthesize the deterministic component from the peak trajectories,

$$d^l(m) = \sum_{r=1}^{R^l} \hat{A}_r^l(m) \cos[\hat{\theta}_r^l(m)], \qquad m = 0, 1, 2, \ldots, S-1 \tag{4.8}$$

where R^l is the number of trajectories present at frame and S is the length of the synthesis frame. $\hat{A}_r^l(m)$ is the instantaneous amplitude, obtained by interpolating the amplitude trajectories, and $\hat{\theta}_r^l(m)$ is the instantaneous phase, obtained by a quadratic interpolation using the frequency and phase trajectories.

5. Obtain the residual $e(n)$ by subtracting the deterministic component $d(n)$ from the original signal $x(n)$,

$$e(n) = x(n) - d(n) \tag{4.9}$$

Therefore the synthesized signal is

$$s(n) = d(n) + e(n) \tag{4.10}$$

which by definition is equal to $x(n)$.

4.10 Examples

This decomposition system can be applied to a variety of sounds and it is very flexible in terms of defining the deterministic component. For the next examples the analysis parameters are set so that the deterministic component extracts as many partials as possible.

4.10.1 Sound example 4

Guitar passage. (*sampling-rate* = 34000, length = 7.14 sec., lowest fundamental $\approx$ 215Hz, highest fundamental $\approx$ 291Hz)

STFT parameters: *window-type* = Kaiser ($\beta = 2.8$), *window-length* = 801 samples (.024 sec.), *FFT-size* = 2048 samples, *hop-size* = 200 samples (.0059 sec.).

Peak detection parameters: *local-dB-range* = 75dB, *general-dB-range* = 85dB, *minimum-peak-height* = 1dB, *frequency-range* = 150Hz–12KHz.

Peak continuation parameters: *maximum-peak-deviation* = 80Hz, *peak-contribution-to-guide* = .4, *maximum-number-of-guides* = 30, *minimum-starting-guide-separation* = 90Hz, *maximum-sleeping-time* = 2 frames (.025 sec.), *length-of-filled-gaps* = 2 frames (.025 sec.), *minimum-trajectory-length* = 20 frames (.25 sec.).

1. original sound

2. deterministic synthesis

3. residual

4. deterministic synthesis plus residual

The peak continuation algorithm does not use the harmonic variation simply because it was unnecessary, thus resulting in one less complication.

The deterministic portion of this guitar passage includes all the stable modes of vibration of the string. The residual includes the left-hand finger-noise, non-linear components of the string vibration, plus other unstable components of the sound such as reverberation and tape-hiss.

4.10.2 Sound example 5

Flute sound. (*sampling-rate* = 18000, length = 2.3 sec., fundamental frequencies $\approx$ 1182Hz and 1118Hz)

STFT parameters: *window-type* = Kaiser ($\beta = 2.8$), *window-length* = 401 samples (.022 sec.), *FFT-size* = 1024 samples, *hop-size* = 100 samples (.007 sec.).

Peak detection parameters: *local-dB-range* = 65dB, *general-dB-range* = 80dB, *minimum-peak-height* = 2dB, *frequency-range* = 400Hz–9KHz.

Peak continuation parameters: *maximum-peak-deviation* = 100Hz, *peak-contribution-to-guide* = .5, *maximum-number-of-guides* = 30, *minimum-starting-guide-separation* = 200Hz, *maximum-sleeping-time* = 0 frames, *length-of-filled-gaps* = 0 frames, *minimum-trajectory-length* = 50 frames (.28 sec.).

1. original sound

2. deterministic synthesis

3. residual

4. deterministic synthesis plus residual

The residual of this sound is very prominent. Its main component is the air produced by the performer that is not transformed into periodic vibrations by the flute.

4.10.3 Sound example 6

Vocal sound. (*sampling-rate* = 16000, length = 4.5 sec., fundamental frequency $\approx$ 94Hz)

STFT parameters: *window-type* = Kaiser ($\beta = 3$), *window-length* = 1001 samples (.063 sec.), *FFT-size* = 2048 samples, *hop-size* = 250 samples (.016 sec.).

Peak detection parameters: *local-dB-range* = 60dB, *general-dB-range* = 70dB, *minimum-peak-height* = 1dB, *frequency-range* = 50Hz–2KHz.

Peak continuation parameters: *harmonic-sound* = true, *maximum-peak-deviation* = 30Hz, *peak-contribution-to-guide* = .5, *maximum-number-of-guides* = 20, *maximum-sleeping-time* = 2 frames (.025 sec.), *length-of-filled-gaps* = 2 frames (.025 sec.), *minimum-trajectory-length* = 40 frames (.5 sec.), *initial-fundamental* = 90Hz, *fundamental-range* = 85Hz–120Hz.

1. original sound
2. deterministic synthesis
3. residual
4. deterministic synthesis plus residual

This is an example of a very "raspy" vocal sound. There is a lot of breath noise and only a few stable harmonics. Most of the high harmonics are completely masked by the breath noise and the deterministic analysis is unable to find them. They are kept in the residual.

4.10.4 Sound example 7

Piano passage. (*sampling-rate* = 34000, length = 4 sec., lowest fundamental $\approx$ 140Hz, highest fundamental $\approx$ 270Hz)

STFT parameters: *window-type* = Kaiser (β = 2.8), *window-length* = 1201 samples (.035 sec.), *FFT-size* = 2048 samples, *hop-size* = 150 samples (.0044 sec.).

Peak detection parameters: *local-dB-range* = 75dB, *general-dB-range* = 85dB, *minimum-peak-height* = 2dB, *frequency-range* = 100Hz–14KHz.

Peak continuation parameters: *maximum-peak-deviation* = 60Hz, *peak-contribution-to-guide* = .4, *maximum-number-of-guides* = 45, *minimum-starting-guide-separation* = 100Hz, *maximum-sleeping-time* = 1 frames, *length-of-filled-gaps* = 1 frames, *minimum-trajectory-length* = 20 frames (.09 sec.).

1. original sound
2. deterministic synthesis
3. residual
4. deterministic synthesis plus residual

The stretching of the piano partials does not allow the use of the harmonic variation of the peak continuation algorithm. A special variation should be designed for piano tones.

This example shows how much noise is present in a normal piano sound. The residual is a very important component of the sound and includes the noise that the fingers make when playing and the noise produced by the piano action.

4.11 Conclusions

In this chapter an analysis/synthesis technique has been introduced that is based on a deterministic plus residual decomposition. The deterministic component comprises sinusoids which are described by a set of magnitude, frequency, and phase values. The residual is defined as the subtraction of the deterministic signal from the original sound. Such a system is an identity process (i.e., the input and the output are mathematically identical). However, it is not very flexible for performing sound transformations. In the next chapter a simplification is presented that is appropriate for sound modifications.

Nonetheless, this sound decomposition is useful in itself for a number of applications. The deterministic component is a set of partials, and the residual includes noise and very unstable components of the sound. This technique has been used by Robert Schumacher and Chris Chafe (Chafe, 1989; Schumacher and Chafe, 1989) to study bow noise in string instruments and breath noise in wind instruments. Other possible decompositions only extract a certain number of partials. In general this decomposition can give a lot of insight into the makeup of sounds.

The residual component is the part of the instrumental sounds that the existing synthesis techniques have a harder time reproducing. This residual is most important in the attack. A practical application would be to add these residuals to synthesized sounds in order to make them more realistic. Since these residuals remain invariant throughout most of the instrumental range, only a few residuals would be necessary to cover all the sounds of a single instrument.

In appendix D a variation of this technique is used to splice an attack of an original sound into an additive synthesis reproduction of it.

Chapter 5

A Deterministic plus Stochastic Model

5.1 Introduction

By going a step further in the deterministic plus residual model presented in the previous chapter, a more flexible and useful representation for sound manipulation is obtained. The change is based on the observation that for the dissertation objective it is irrelevant to preserve the instantaneous phase of the original waveform, both in the deterministic component and in the residual. Even more, when the residual approaches a stochastic, or noise, signal it is also irrelevant to maintain the exact frequency characteristics (i.e., the exact magnitude spectrum), and the spectrum's general shape suffices. These observations lead to the deterministic plus stochastic model.

To disregard phase in the deterministic signal means that the phase tracking performed in the previous chapter is unnecessary. Now, only phase continuity is preserved.

To restrict the residual to be a stochastic signal simplifies enormously the residual signal, but it implies that the deterministic component has to account for whatever is not stochastic. In contrast, in the previous chapter, if the deterministic component omitted some partials, these were preserved in the residual. Therefore in the current system, the extraction of the deterministic part is more critical than before.

With this new model, a musically useful representation is obtained that enables transformations of a variety of sounds, extending the capabilities of the previous models. The

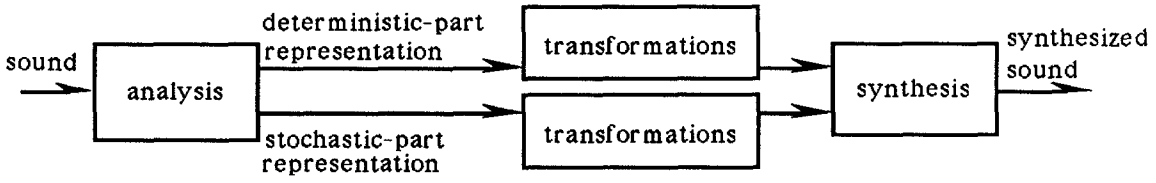

Figure 5.1: General diagram of the deterministic plus stochastic system.

resulting representation (Figure 5.1) has two parts: (1) a series of frequency and amplitude functions for the deterministic component, and (2) a series of magnitude-spectrum envelopes for the stochastic part of the sound.

The deterministic signal is generated from the amplitude and frequency functions with additive synthesis. The stochastic component is created by performing the inverse-STFT of the spectral envelopes. The sum of the two resulting waveforms is, for many sounds, perceptually very close to the original.

Compared with the system of the previous chapter there is a gain in the flexibility of the representation in exchange for the identity property of the process. With the deterministic plus residual model, any sound was represented; on the other hand, with the deterministic plus stochastic model not every sound can be fit by the model. For example, multiple source sounds are not appropriate because the deterministic plus stochastic separation is more confused.[1] But, for the objective of this dissertation, the flexibility is the most important attribute of a representation, and some compromises in generality are made in order to achieve it.

The sections of this chapter are organized in the following way. In the next section the deterministic plus stochastic model is presented, followed by a general description of the analysis/synthesis technique based on it. Then, in the next sections there is a detailed presentation of the steps carried out by the technique. The chapter ends with a set of examples, and the conclusions.

5.2 The Deterministic plus Stochastic Model

This model is based on a modification of the deterministic plus residual model presented in Chapter 4. That model considered a waveform $s(t)$ as the sum of a series of sinusoids plus a residual,

$$s(t) = \sum_{r=1}^{R} A_r(t)\cos[\theta_r(t)] + e(t) \tag{5.1}$$

where $A_r(t)$ and $\theta_r(t)$ are the instantaneous amplitude and phase of each sinusoid and $e(t)$ is the residual signal.

Now, in the deterministic plus stochastic model the general equation 5.1 still applies, but there is a simplification of the components. In this new model each sinusoid is described only by its amplitude and frequency (i.e., the phase term is ignored). The instantaneous phase is then taken to be the integral of the instantaneous frequency,

$$\theta_r(t) = \int_0^t \omega_r(\tau)d\tau \tag{5.2}$$

[1] There are some possible extensions to the technique which could handle multiple source sounds.

where $\omega(t)$ is the radian frequency measured from the power spectrum, and r is the sinusoid number. (This equation is the counterpart of equation 4.3.)

The simplification of the residual $e(t)$ in (5.1) is based on assuming that it is a stochastic signal. Such an assumption permits modeling the residual as filtered white noise,

$$\hat{e}(t) = \int_0^t h(t, t-\tau)u(\tau)d\tau \tag{5.3}$$

where $u(t)$ is white noise and $h(t,\sigma)$ is the impulse response of a slowly time varying filter (at time t the impulse response is $h(t,\cdot)$). That is, the residual is modeled by the convolution of white noise with a frequency shaping filter.

The filtering of a noise signal can be implemented by taking the inverse Fourier transform of the filter frequency response times a random phase term. This last approach is the one taken in this chapter to synthesize the stochastic signal.

5.3 General Description of the System

Figure 5.2 shows a general diagram of a system based on the deterministic plus stochastic model. First we derive a series of magnitude spectra from the waveform by computing the STFT. The phase spectrum is not needed and therefore it is not calculated. Next we detect and follow the prominent peaks on each spectrum, resulting in a set of peak trajectories with a magnitude and a frequency value for every frame. From these trajectories we synthesize the deterministic part of the sound by generating a sine wave for each trajectory.

To calculate the stochastic part of the waveform we first obtain the magnitude-spectrum residual. This is done by computing the magnitude spectrum of the deterministic component and then subtracting it from the corresponding magnitude spectrum of the original waveform. Each magnitude-spectrum residual is simplified by fitting an envelope to it. The resulting set of envelopes constitutes the stochastic representation. From every spectral envelope the corresponding complex spectrum is generated. Then, the stochastic waveform is synthesized by performing an inverse-STFT using the overlap-add method.

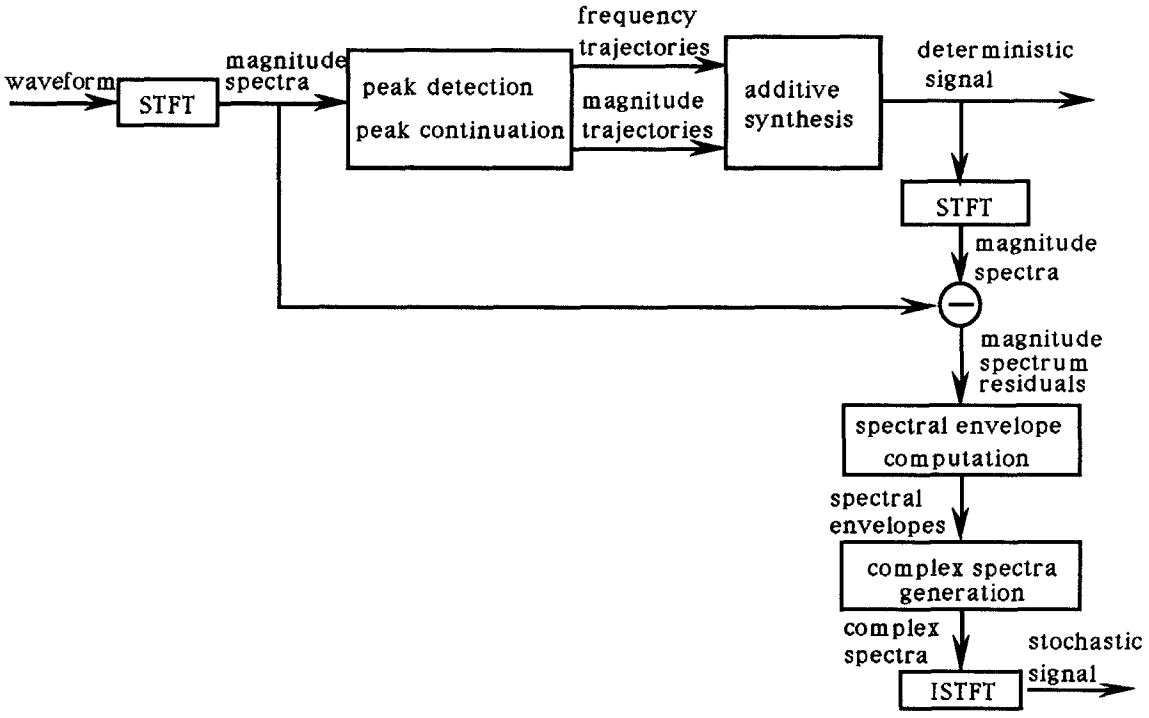

Figure 5.2: Block diagram of the deterministic plus stochastic system.

5.4 Computation of the Magnitude Spectra

With the assumption that the sinusoids are deterministic and the residual stochastic, the phase spectrum is unnecessary in the analysis process. Therefore, the first step is the computation of the set of magnitude spectra of the sound. This is accomplished by using the STFT as described in Chapter 2, and simply discarding each phase spectrum. Thus, there are a few steps in the STFT process that are simplified.

When the phase spectrum is relevant it is important to place the windowed data in the FFT-buffer in such a way that the phase of each spectral peak corresponds to the actual phase of the sinusoid in the time domain (i.e., constant-phase spectrum). This is realized by centering the windowed data around the origin of the data buffer when the FFT is computed (as described in Chapter 2). On the other hand, when the phase spectrum is irrelevant this step is unnecessary and the windowed data can be placed anywhere in the FFT-buffer. For the same reason the analysis window is not restricted to an odd-length size in order to be perfectly centered around the origin.

The settings for the rest of the parameters used in the STFT, that is, *window-type*, *FFT-size*, and *hop-size*, can be the same as in the deterministic plus residual system of the previous chapter.

5.5 Spectral Peak Detection and Continuation

Once the set of magnitude spectra are computed, the next step is to find the prominent peaks in every spectrum, and their continuation in time. The process is the same as for the deterministic plus residual model. The only difference is that the analysis returns only magnitude and frequency values for each peak, not phase. Therefore the deterministic analysis comprises a simplified version of the peak detection and peak continuation algorithms discussed in Chapter 4.

It has already been mentioned that for the current model to be successful the deterministic analysis should be able to extract as many partials of the sound as possible. Thus, a careful setting of all the parameters is necessary. This does not mean that all the analysis data has to be used for the final synthesis. All the data is used to perform the subtraction, but once the residual is obtained the deterministic representation can be simplified, thus excluding irrelevant sinusoids.

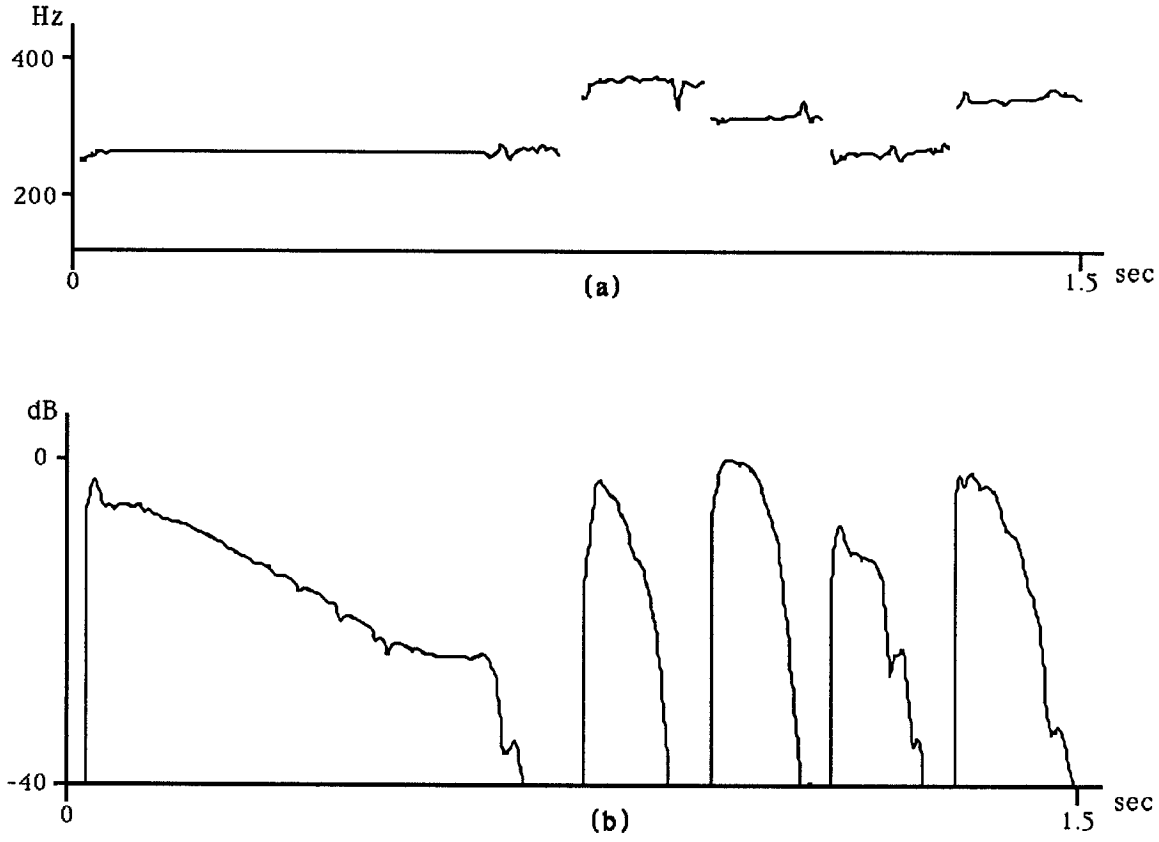

Figure 5.3: Deterministic representation example: (a) frequency function of the first partial of a piano phrase, (b) amplitude function for the same partial.

5.6 Representation of the Deterministic Part

The output of the peak continuation algorithm is a set of peak trajectories. These represent the deterministic component, i.e., the partials of the analyzed sound. Each peak is a pair of numbers of the form $(\hat{A}_r(l), \hat{\omega}_r(l))$ where $\hat{A}$ and $\hat{\omega}$ are the amplitude and frequency, respectively, for each frame l and each trajectory r. The pairs corresponding to a trajectory r are interpreted as breakpoints for amplitude and frequency functions, one breakpoint for each frame l (Figure 5.3). From these functions a series of sinusoids can be synthesized which reproduce the deterministic part of the sound.

These amplitude and frequency functions can be further processed to achieve a data reduction of the representation or a smoothing of the functions. A data reduction strategy is to perform a line-segment approximation on each function, thus reducing the number of breakpoints (Grey, 1975; Strawn, 1980). However, for the purpose of easy manipulation of the representation it is useful to have equally spaced points along each function, and

thus it may be better to keep one breakpoint per frame as returned by the analysis, unless data reduction is a priority. Another alternative for data reduction is to combine groups of similar functions into a single one, thus reducing the number of functions.

5.7 Deterministic Synthesis

Given the representation of the deterministic part of the sound, the generation of the time domain waveform is done with an additive synthesis technique. From the amplitude and frequency functions, $\hat{A}_r(l)$ and $\hat{\omega}_r(l)$, a frame of the deterministic sound is obtained by

$$d^l(m) = \sum_{r=1}^{R^l} \hat{A}_r^l \cos[m\hat{\omega}_r^l], \qquad m = 0, 1, 2, \ldots, S-1 \tag{5.4}$$

where R^l is the number of trajectories present at frame l and S is the length of the synthesis frame (without any time scaling $S = H$, the analysis hop size). The final sound $d(n)$ results from the juxtaposition of all the synthesis frames. To avoid "clicks" at the frame boundaries, the parameters $(\hat{A}_r^l, \hat{\omega}_r^l)$ are smoothly interpolated from frame to frame.

This synthesis process corresponds to the one used for the sinusoidal model of Chapter 3, but now the phase trajectory is discarded. The instantaneous amplitude $\hat{A}(m)$ is obtained by linear interpolation,

$$\hat{A}(m) = \hat{A}^{l-1} + \frac{(\hat{A}^l - \hat{A}^{l-1})}{S} m \tag{5.5}$$

where $m = 0, 1, \ldots, S-1$ is the time sample in the lth frame. The main difference with Chapter 3 is that the instantaneous phase is now taken to be the integral of the instantaneous frequency, where the instantaneous radian frequency $\hat{\omega}(m)$ is obtained by linear interpolation,

$$\hat{\omega}(m) = \hat{\omega}^{l-1} + \frac{(\hat{\omega}^l - \hat{\omega}^{l-1})}{S} m \tag{5.6}$$

and the instantaneous phase for the rth trajectory is

$$\hat{\theta}_r(m) = \hat{\theta}_r(l-1) + \hat{\omega}_r(m) \tag{5.7}$$

Finally, the synthesis equation becomes

$$d^l(m) = \sum_{r=1}^{R^l} \hat{A}_r^l(m) \cos[\hat{\theta}_r^l(m)] \tag{5.8}$$

where $\hat{A}(m)$ and $\hat{\theta}(m)$ are the calculated instantaneous amplitude and phase.

5.8 Computation of the Stochastic Part

Once the deterministic component of the sound has been detected, the next step is to obtain the residual, which in a simplified form becomes the stochastic component.

In the previous chapter the deterministic component preserved the instantaneous phase of the original sound, thus allowing a time domain subtraction. In the current system the phase is discarded, making the time domain subtraction useless. However since the magnitude and frequency of each sinusoid are preserved, the magnitude spectrum of both signals is comparable, as shown in Figure 5.4. Accordingly it is possible to perform a frequency domain subtraction from the magnitude spectra of both signals. The result is a set of magnitude-spectrum residuals.

One of the underlying assumptions of the current model is that the residual is a stochastic signal. Such an assumption implies that the residual is fully described by its amplitude and its general frequency characteristics. It is unnecessary to keep either the instantaneous phase or the exact frequency information. Based on this the stochastic residual can be completely characterized by the envelopes of the magnitude-spectrum residuals, i.e., these envelopes keep the amplitude and the general frequency characteristic of the residual. The set of envelopes form the stochastic representation.

The computation of the stochastic representation involves: (1) subtraction of each magnitude spectrum of the deterministic component from the corresponding magnitude spectrum of the original sound, and (2) approximation of each residual spectrum with an envelope. Each step is described next.

5.8.1 Computation of the magnitude-spectrum residuals

The first step in obtaining the stochastic component is to subtract the set of magnitude spectra of the deterministic signal from that of the original sound. This results in the magnitude-spectrum residuals (Figure 5.5). For this to be feasible the spectra to be subtracted have to be comparable, and therefore have to be computed in the same manner. The STFTs from which they are obtained use the same *window-type*, *window-length*, *FFT-size*, and *hop-size*.

Given that the magnitude spectrum of the original sound at frame l is $|X_l(k)|$ and that of the deterministic signal $|D_l(k)|$, then the residual is

$$|E_l(k)| = |X_l(k)| - |D_l(k)| \tag{5.9}$$

To avoid negative numbers, in case the spectrum of the deterministic signal results in slightly higher spectral peaks than the original spectrum, the subtraction is done as

$$|E_l(k)| = \max\left([|X_l(k)| - |D_l(k)|], 0\right) \tag{5.10}$$

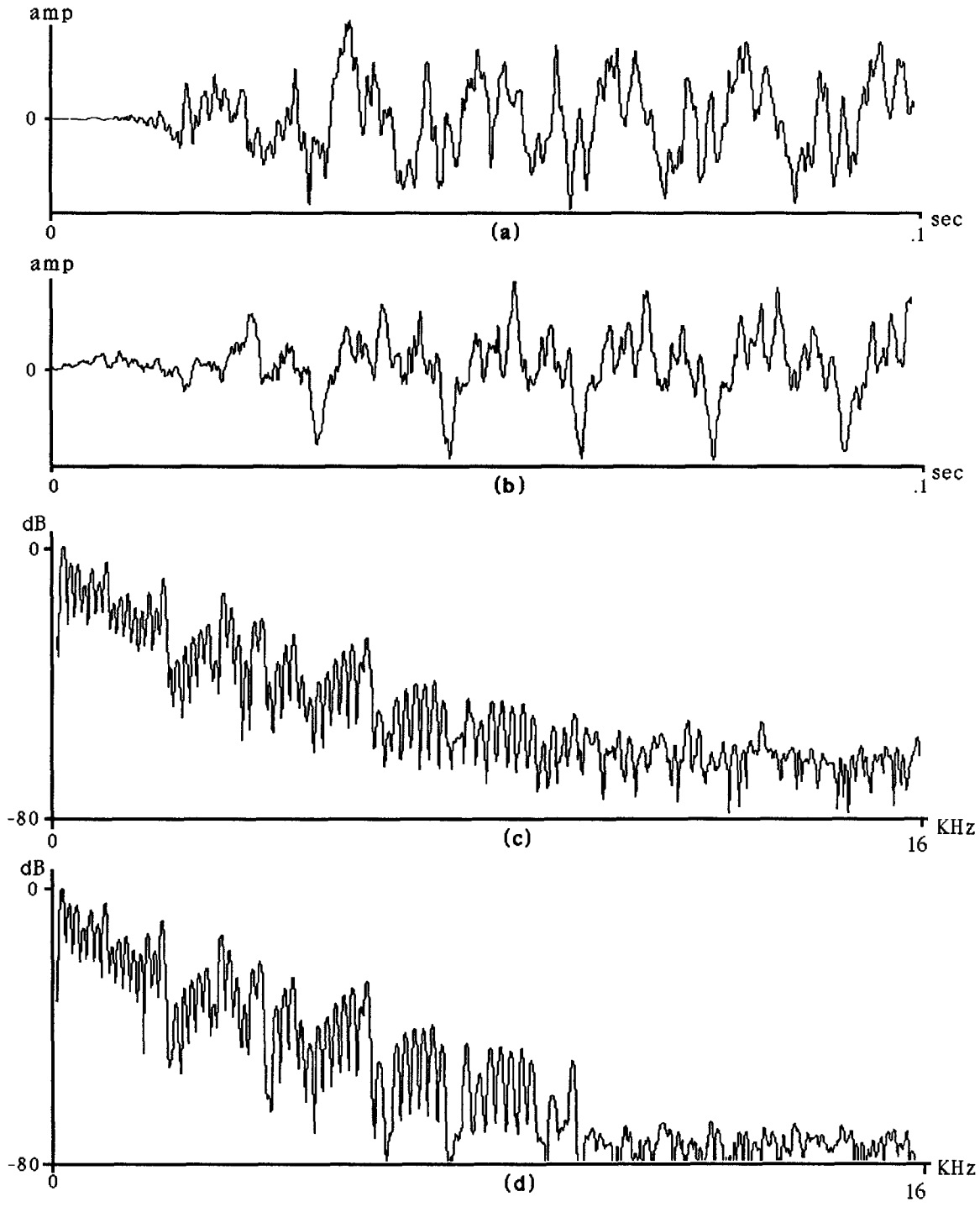

Figure 5.4: Example of the deterministic synthesis without phase tracking: (a) waveform from an original piano tone, (b) deterministic component without phase tracking, (c) magnitude spectrum of original sound, (d) magnitude spectrum of deterministic component.

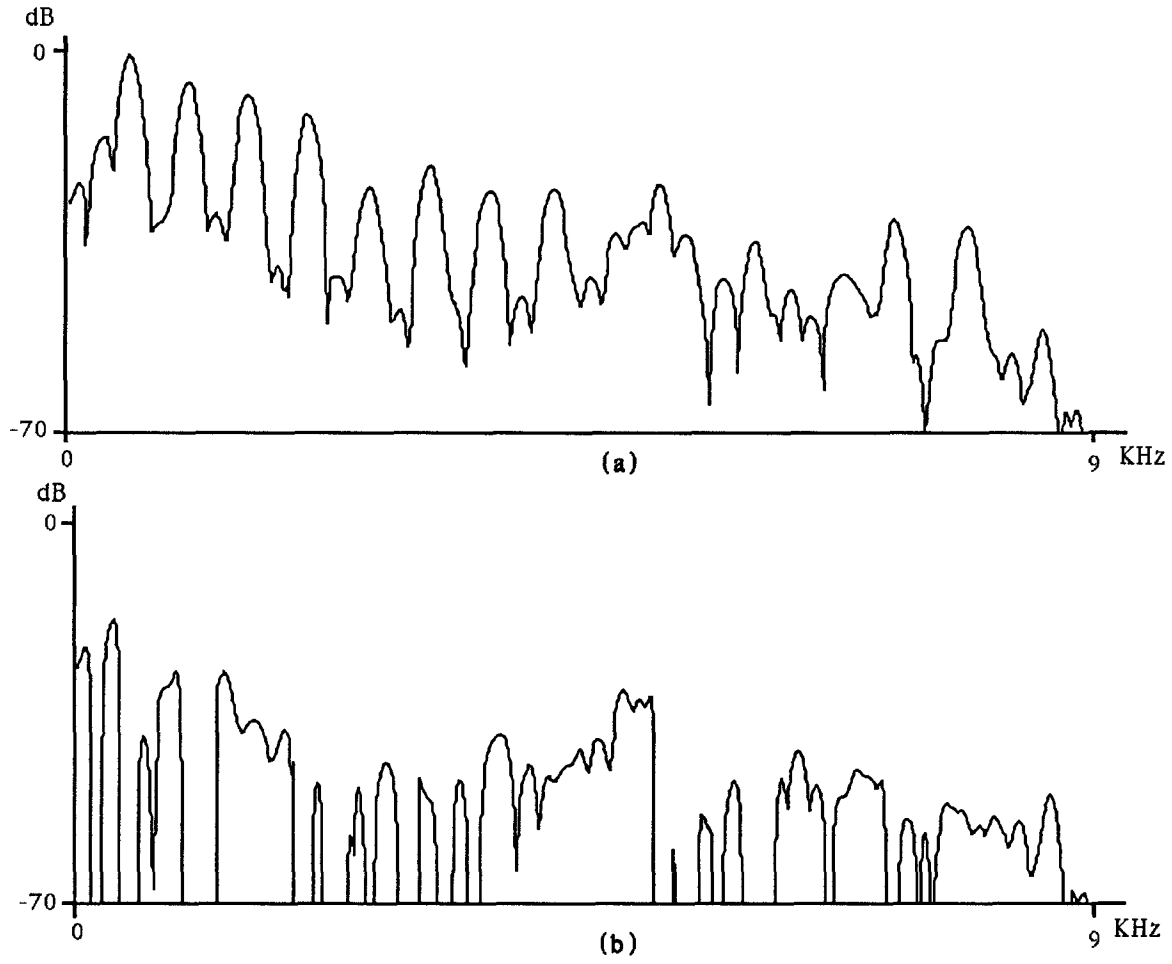

Figure 5.5: Example of the magnitude spectrum subtraction: (a) magnitude spectrum from a piano tone, (b) residual spectrum.

5.8.2 Approximation of the spectral residual

Assuming that the residual signal is quasi-stochastic, each magnitude-spectrum residual can be approximated by its envelope, since only its shape contributes to the sound characteristics.

This type of problem is generally solved by performing some sort of curve fitting (Strawn, 1980; Sedgewick, 1988), i.e., finding a function which matches the general contour of a given curve, which in our case is a magnitude spectrum. Standard techniques are: spline interpolation (Cox, 1971), the method of least squares (Sedgewick, 1988), or straight line approximations (Phillips, 1968). For the purpose of this thesis a simple line-segment approximation is accurate enough and gives the desired flexibility.

Another practical alternative is to use a type of least squares approximation called linear predictive coding, LPC (Makhoul, 1975; Markel and Gray, 1976). LPC is a popular technique used in speech research for fitting an nth-order polynomial to a magnitude spectrum. Its use in this particular situation is discussed in Appendix C. Here it is sufficient to say that the line-segment approach is more flexible than LPC, and even though LPC results in less analysis points, the flexibility is considered more important.

The particular line-segment approximation performed here is done by stepping through the magnitude spectrum and finding local maxima in every section,

$$\tilde{E}_l(q) = \max_k(|E_l(qM+k)|), \qquad \begin{aligned} k &= -M/2, -M/2+1, \ldots, 0, \ldots, M/2-2, M/2-1, \\ q &= 0, 1, \ldots, N/M-1 \end{aligned} \tag{5.11}$$

where M is the step size and the window size (or size of the section) and $\tilde{E}_l(q)$ is the maximum of section q at frame l. This gives $Q(= N/M)$ equally spaced points in the spectrum that are connected by straight lines to create the spectral envelope (Figure 5.6). The accuracy of the fitting is given by the number of points, Q, which is set depending on the sound complexity.

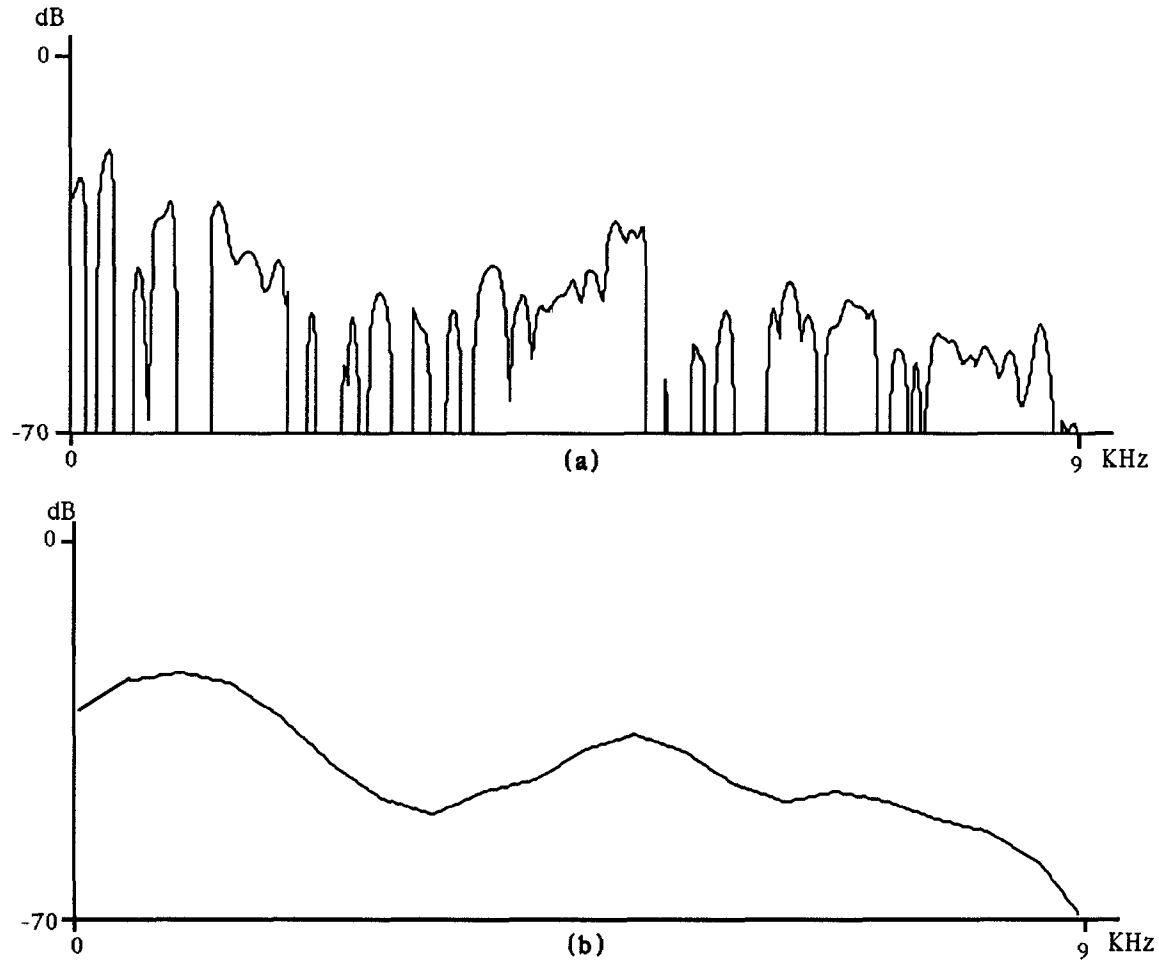

Figure 5.6: Example of the line-segment approximation on a spectral residual: (a) residual spectrum from a piano tone, (b) envelope approximation of the residual.

5.9 Representation of the Stochastic Part

The stochastic analysis returns an envelope $\tilde{E}_l(q)$ for every frame l, where q is the breakpoint number in the envelope, $q = 0, 1, \ldots, Q-1$. These envelopes can be interpreted differently depending on which variable, l or q, is considered fixed.[2] When l is fixed the interpretation is as a series of envelopes or frequency-shaping filters, one per frame. But when q is fixed the interpretation is as equally spaced band-pass filters, each one centered at $f_s q/2Q$ Hz and with a bandwidth of $f_s/2Q$Hz.

These frequency envelopes or time functions, depending on the interpretation, can be simplified and smoothed as in the deterministic representation. As with that representation it is useful to keep the same number of breakpoints on both the frequency and the time axes.

5.10 Stochastic Synthesis

The synthesis of the stochastic component can be understood as the generation of a noise signal that has the frequency and amplitude characteristics described by the spectral envelopes of the stochastic representation. The intuitive operation is to filter white noise with these frequency envelopes, that is, performing a time-varying filtering of white noise. But in practice we generate the stochastic signal by an overlap-add synthesis technique (discussed in Chapter 2) from the spectral envelopes. The inverse Fourier transform of each envelope is computed and the resulting waveforms are overlaped and added.

Before the inverse-STFT is performed, a complex spectrum (i.e., magnitude and phase spectra), is obtained from each frequency envelope. The magnitude spectrum is generated by linear interpolating the approximation $\tilde{E}_l(q)$ of length Q to a curve of length $N/2$, where N is the *FFT-size*. There is no phase information in the stochastic representation, but since the phase spectrum of noise is a random signal, the phase spectrum can be created with a random number generator. To avoid a periodicity at the frame rate different values are generated at every frame. Therefore the magnitude and phase spectra at frame l are

$$\begin{aligned} A_l(k) &= \tilde{E}'_l(k) \\ \Theta_l(k) &= \pi - \text{ran}(2\pi) \end{aligned} \tag{5.12}$$

where $\tilde{E}'_l(k)$ is the interpolated spectral envelope and $\text{ran}(2\pi)$ is a function that produces random numbers in the range from 0 to 2π (i.e., according to its argument).

[2]This is similar to the difference between overlap-add versus filter-bank interpretation in the STFT.

From the interpolated magnitude envelope and the random phase spectrum, the complex spectrum $\hat{E}_l(k)$ results from a change of the coordinates,

$$\begin{aligned} \mathrm{Re}\{\hat{E}_l(k)\} &\triangleq A_l(k)\cos[\Theta_l(k)] \\ \mathrm{Im}\{\hat{E}_l(k)\} &\triangleq A_l(k)\sin[\Theta_l(k)] \end{aligned} \tag{5.13}$$

Its inverse Fourier transform gives one frame of the noise waveform,

$$\hat{e}'_l(m) = \frac{1}{N}\sum_{k=-N/2}^{N/2-1} \hat{E}_l(k)e^{j\omega_k m}, \qquad m = 0,1,\ldots,N-1 \tag{5.14}$$

The waveform $\hat{e}'_l(m)$ is a constant-amplitude waveform of size N, where N is the *FFT-size*. Since the phase spectrum used is not the result of an analysis process (with windowing of a waveform, zero-padding, and FFT computation) the resulting waveform does not maintain the windowing characteristics of the analyzed waveform. This is because a phase spectrum with random values corresponds to a phase spectrum of a rectangular-windowed noise waveform of size N (i.e., no windowing or zero-padding). But in order to succeed in the overlap-add we need a windowed waveform of size M, where M is the analysis-window length. Therefore the resulting waveform $\hat{e}'_l(m)$ is multiplied by a length M window,

$$\hat{e}_l(m) = \hat{e}'_l(m)w(m), \qquad m = 0,1,\ldots,M-1 \tag{5.15}$$

This windowing process corresponds to the use of a synthesis window in the inverse-STFT, as discussed in Chapter 2. There is no reason to use the same window as in the STFT-analysis, nor to use a very "sophisticated" one. A simple Hanning window suffices. Then the stochastic signal results from the overlap and add of these windowed waveforms,

$$\hat{e}(n) = \sum_{l=0}^{L-1} \hat{e}_l(n - lH) \tag{5.16}$$

where H is the analysis *hop-size* and l is the frame number. Figure 5.7 shows an example of the stochastic synthesis.

5.11 Representation Modifications

The deterministic analysis results in a set of amplitude and frequency functions, $\hat{A}_r(l)$ and $\hat{\omega}_r(l)$, where r is the function number, and l the breakpoint number in each function. The stochastic analysis results in a set of spectral envelopes, $\tilde{E}_l(q)$, where q is the breakpoint number on the envelope. Together these representations are ideal for modification purposes, they allow a great number of sound transformations. The modifications are applied separately to the deterministic and stochastic representations.

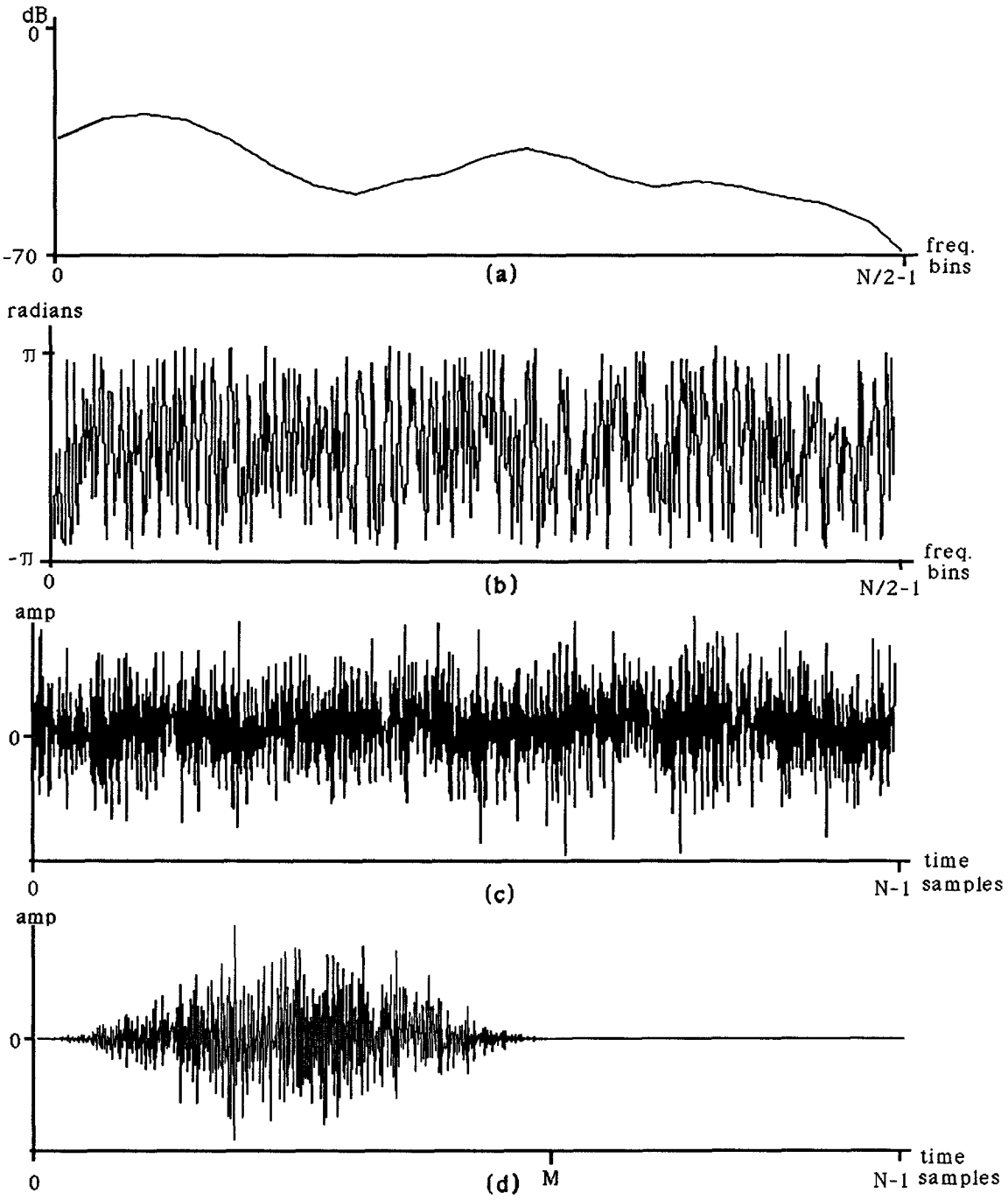

Figure 5.7: Stochastic synthesis example: (a) envelope of a spectral residual, (b) generated phase-spectrum, (c) inverse Fourier-transform of the complex spectrum, (d) windowed inverse Fourier-transform.

Time-scale modifications are accomplished in both representations by resampling the analysis points in time. This is done by changing the synthesis frame-size (and the hop-size in the case of the stochastic synthesis) and results in slowing down or speeding up the sound while maintaining pitch and formant structure. A time-varying frame-size gives a time-varying modification. Due to the stochastic and deterministic separation, this representation is more successful in time-scale modifications than the sinusoidal representation (presented in Chapter 3). Now the noise part of the sound remains "noise" no matter how much the sound is stretched.

In the deterministic representation each function pair, amplitude and frequency, represents a partial of the original sound. The manipulation of these functions is easy and musically intuitive. All kinds of frequency and magnitude transformations are possible. For example, the partials can be transposed in frequency, with different values for every partial and varying along the sound. It is also possible to decouple the sinusoidal frequencies from their amplitude, obtaining effects such as changing pitch while maintaining formant structure. Graphical interfaces are easily devised to manipulate this representation.

The stochastic representation is modified by changing the shape of each of the envelopes. This can be done by applying functions to these envelopes, or simply editing them by hand. Changing the envelope shape corresponds to filtering the stochastic signal further. Their manipulation is much simpler and more intuitive than the manipulation of a set of all-pole filters, such as those resulting from an LPC analysis.

Interesting effects are accomplished by changing the relative amplitude of the two components, thus emphasizing one or the other at different moments in time.

The characterization of a single sound by two different representation may cause problems. When different transformations are applied to each representation it is easy to create a sound in which the two components, deterministic and stochastic, do not fuse into a single entity. This may be desirable for some musical applications, but in general it is avoided, and requires some practical experimentation with the actual representations.

The best synthesis is generally considered the one that results in the best perceptual identity with respect to the original sound. Then, transformations are performed on the corresponding representation. But for musical applications this may not be always desirable. Very interesting effects result from purposely setting "wrong" the analysis parameters. For example we may set the parameters such that the deterministic analysis only captures partials in a specific frequency range, leaving the rest to be considered stochastic. The result is a sound with a much stronger noise component (Wolman, 1989).

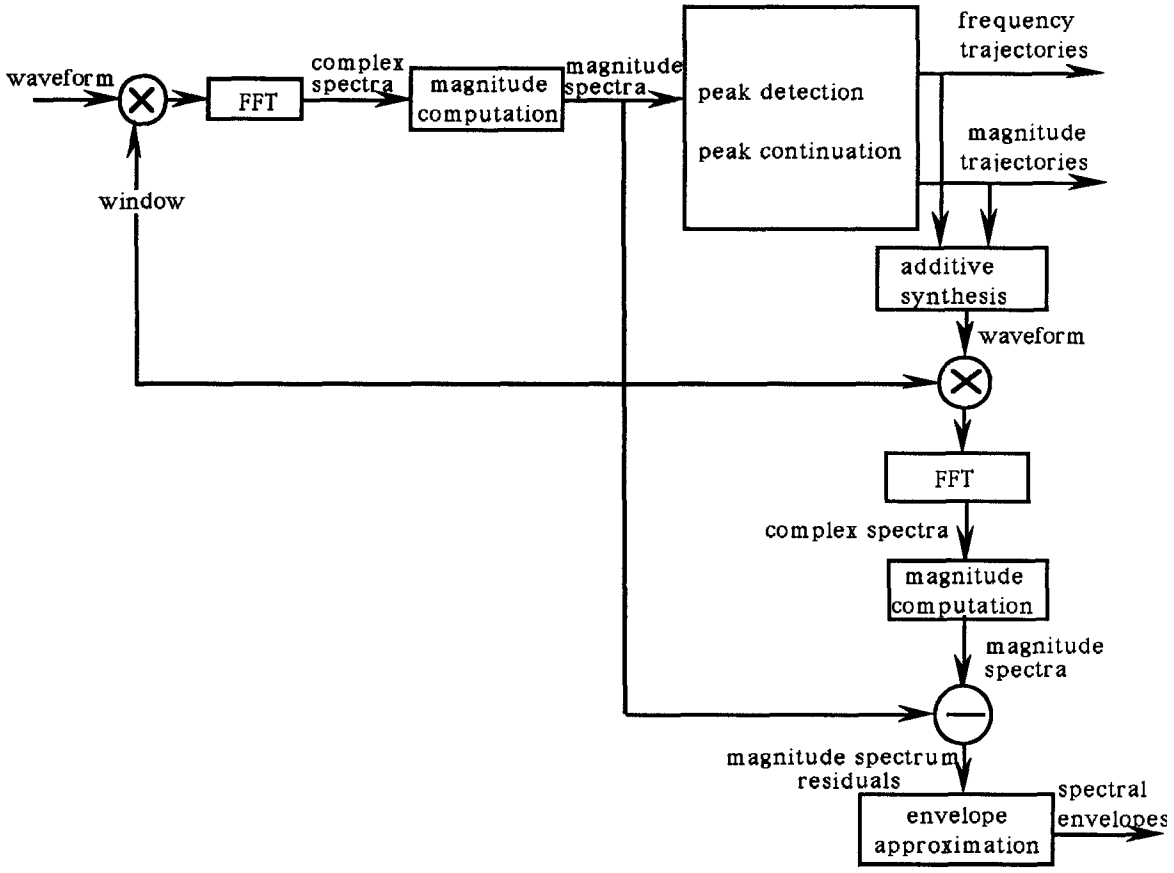

Figure 5.8: Block diagram of the analysis part of the deterministic plus stochastic system.

5.12 Summary of the Analysis Technique

Figure 5.8 shows a block diagram of the analysis part of the system. A review of the steps for a computer implementation is given below.

1. Perform a STFT with specific values for *window-type, window-length, FFT-size*, and *hop-size*,

$$X_l(k) \triangleq \sum_{n=0}^{N-1} w(n)x(n+lH)e^{-j\omega_k n} \qquad l = 1, 2, \ldots \tag{5.17}$$

where $w(m)$ is the analysis window, H the *hop-size*, and l the frame number. The result is a series of complex spectra from which only the magnitude spectra $|X_l(k)|$ are computed.

2. Detect the prominent magnitude-peaks in specific frequency and magnitude ranges using the peak-detection algorithm. The result is a set of amplitudes $\hat{A}_i^l$, and frequencies $\hat{\omega}_i^l$, for each frame l.

3. Organize peak trajectories from a subset of the peaks using the peak-continuation algorithm. Now, the values for each peak $(\hat{A}_r^l, \hat{\omega}_r^l)$ belong to a specific frequency trajectory r.

4. Synthesize the deterministic component (as described below), then compute its STFT with the same parameters as for the original sound, and subtract each magnitude spectrum of the synthesized sound from the corresponding spectrum of the original sound. This results in a set of magnitude-spectrum residuals,

$$|E_l(k)| = |X_l(k)| - |D_l(k)| \tag{5.18}$$

where $|X_l(k)|$ is the original spectrum, $|D_l(k)|$ the deterministic spectrum, and $|E_l(k)|$ the residual spectrum.

5. Compute the envelope for each spectral residual by performing a line-segment approximation on each one,

$$\begin{aligned} \tilde{E}_l(q) = \max_k(|E_l(qM+k)|), \qquad & k = -M/2, -M/2+1, \ldots, 0, \ldots, M/2-2, \\ & M/2-1, \\ & q = 0, 1, \ldots, N/M-1 \end{aligned} \tag{5.19}$$

where M is the step size and the window size (or size of the section) and $\tilde{E}_l(q)$ is the maximum of section q and frame l.

5.13 Summary of the Synthesis Technique

Figure 5.9 shows the diagram of the synthesis part of the deterministic plus stochastic system. The steps involved are the following.

1. Modify the amplitude and frequency functions of the deterministic representation,

$$\begin{aligned} \tilde{A}_r(l) &= T\left[\hat{A}_r(l)\right] \\ \tilde{\omega}_r(l) &= T\left[\hat{\omega}_r(l)\right] \end{aligned} \tag{5.20}$$

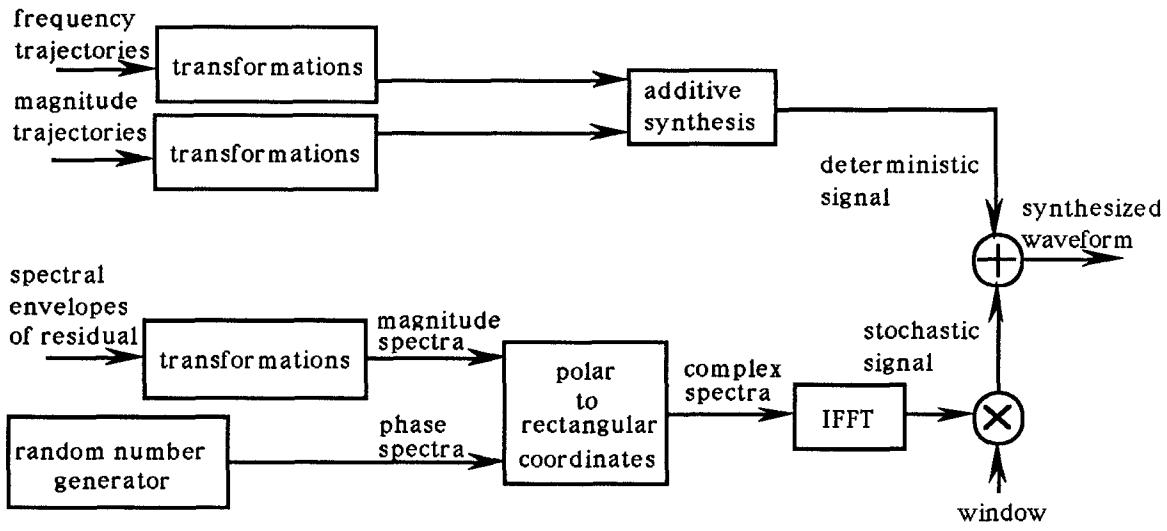

Figure 5.9: Block diagram of the synthesis part of the deterministic plus stochastic system.

2. Generate the deterministic signal from the modified amplitude and frequency functions,

$$d^l(m) = \sum_{r=1}^{R^l} \tilde{A}_r^l(m) \cos[\tilde{\theta}_r^l(m)], \qquad m = 0, 1, 2, \ldots, S-1 \tag{5.21}$$

where R^l is the number of trajectories present at frame l and S is the length of the synthesis frame (the analysis *hop-size* times the desired time-scale factor). The instantaneous amplitude $\tilde{A}_r^l(m)$ and phase $\tilde{\theta}_r^l(m)$ are obtained by linearly interpolating the amplitude and frequency trajectories respectively.

3. Apply any modifications to the stochastic representation,

$$\tilde{E}_l'(q) = T\left[\tilde{E}_l(q)\right] \tag{5.22}$$

4. Generate a magnitude and phase spectrum from every modified envelope. The magnitude spectrum is the result of interpolating the residual approximation, and the phase spectrum is generated with a random number generator,

$$\begin{aligned} A_l(k) &= \tilde{E}_l''(k) \\ \Theta_l(k) &= \pi - \mathrm{ran}(2\pi) \end{aligned} \tag{5.23}$$

where $\tilde{E}_l''(k)$ is the interpolated spectral envelope and $\mathrm{ran}(2\pi)$ is a function that generates random numbers in the range from 0 to 2π.

5. Convert each spectrum from polar to rectangular coordinates,

$$\begin{aligned} \mathrm{Re}\{\hat{E}_l(k)\} &\triangleq A_l(k)\cos[\Theta_l(k)] \\ \mathrm{Im}\{\hat{E}_l(k)\} &\triangleq A_l(k)\sin[\Theta_l(k)] \end{aligned} \tag{5.24}$$

6. Compute the inverse-FFT of $\hat{E}_l$,

$$\hat{e}'_l(m) = \frac{1}{N}\sum_{k=-N/2}^{N/2-1} \hat{E}_l(k)e^{j\omega_k m}, \qquad m = 0, 1, \ldots, N-1 \tag{5.25}$$

7. Apply a synthesis window,

$$\hat{e}_l(m) = \hat{e}'_l(m)w(m), \qquad m = 0, \ldots, M-1 \tag{5.26}$$

where $w(m)$ is a Hanning window. Its length M is the length of the analysis window times the desired stretch factor.

8. Generate the final stochastic signal by overlapping and adding the windowed waveforms,

$$\hat{e}(n) = \sum_{l=0}^{L-1} \hat{e}_l(n - lH) \tag{5.27}$$

where H is the analysis *hop-size* times the stretch factor.

9. Sum the deterministic and the stochastic components,

$$s(n) = d(n) + \hat{e}(n) \tag{5.28}$$

5.14 Examples

The current system is appropriate for the analysis/transformation/synthesis of a wide variety of sounds. Next we present a few sound examples that show the capabilities of the system.

5.14.1 Sound example 8

Guitar passage. (*sampling-rate* = 34000, length = 7.14 sec., lowest fundamental ≈ 215Hz, highest fundamental ≈ 291Hz)

STFT parameters: *window-type* = Kaiser ($\beta = 2.8$), *window-length* = 801 samples (.024 sec.), *FFT-size* = 2048 samples, *hop-size* = 200 samples (.0059 sec.).

Peak detection parameters: *local-dB-range* = 75dB, *general-dB-range* = 85dB, *minimum-peak-height* = 1dB, *frequency-range* = 150Hz–12KHz.

Peak continuation parameters: *maximum-peak-deviation* = 80Hz, *peak-contribution-to-guide* = .4, *maximum-number-of-guides* = 30, *minimum-starting-guide-separation* = 90Hz, *maximum-sleeping-time* = 2 frames (.025 sec.), *length-of-filled-gaps* = 2 frames (.025 sec.), *minimum-trajectory-length* = 20 frames (.25 sec.).

1. original sound
2. deterministic synthesis
3. stochastic synthesis
4. deterministic plus stochastic synthesis
5. frequency transposition by a factor of .3
6. frequency transposition by .7 and stretching of partials
7. compression of the frequency evolution
8. inversion of the frequency evolution
9. time-varying glissando and stretching of partials
10. time-varying time-scale
11. time expansion by 2.3
12. time expansion by 2.3 with time-varying time-scale and stretching of partials
13. time compression by .5 with time-varying time-scale and stretching of partials
14. time compression by .5 and frequency transposition by a factor of .4
15. time compression by .5 and glissando down

The setting of the analysis parameters is the same than for sound example 4.

It is important to remark that the final number of sinusoids is not the *maximum-number-of-guides*, the actual number is 25.

All the transformations performed in the examples of this chapter are obtained by applying functions to the three data structures that form the representation, that is, frequency and amplitude functions of the deterministic component and envelopes of the stochastic component.

5.14.2 Sound example 9

Speech phrase. (*sampling-rate* = 16000, length = 4.5 sec., fundamental frequency $\approx$ 94Hz)

STFT parameters: *window-type* = Kaiser ($\beta = 3$), *window-length* = 1001 samples (.063 sec.), *FFT-size* = 2048 samples, *hop-size* = 250 samples (.016 sec.).

Peak detection parameters: *local-dB-range* = 60dB, *general-dB-range* = 70dB, *minimum-peak-height* = 1dB, *frequency-range* = 50Hz–20KHz.

Peak continuation parameters: *harmonic-sound* = true, *maximum-peak-deviation* = 30Hz, *peak-contribution-to-guide* = .5, *maximum-number-of-guides* = 20, *maximum-sleeping-time* = 2 frames (.025 sec.), *length-of-filled-gaps* = 2 frames (.025 sec.), *minimum-trajectory-length* = 40 frames (.5 sec.), *initial-fundamental* = 165Hz, *fundamental-range* = 85Hz–120Hz.

1. original sound
2. deterministic synthesis
3. stochastic synthesis
4. deterministic plus stochastic synthesis
5. frequency transposition by a factor of .6
6. compression of the frequency evolution and frequency transposition by a factor of .4
7. frequency transposition by .4 and stretching of partials
8. time-varying glissando and stretching of partials
9. time-varying time-scale and time-varying compression of the frequency evolution
10. from deterministic to stochastic signal

11. time compression by .3, compression of the frequency, and frequency transposition by a factor of .4
12. time compression by .3 and compression of the frequency
13. time expansion by 3
14. time expansion by 3 of only stochastic component and time-varying time-scale

The sampling rate of this example is fairly low, 16000, and 20 guides suffice to detect the deterministic component.

5.14.3 Sound example 10

Conga passage. (*sampling-rate* = 32000, length = 3 sec.)

STFT parameters: *window-type* = Kaiser ($\beta = 2.8$), *window-length* = 800 samples (.025 sec.), *FFT-size* = 1024 samples, *hop-size* = 100 samples (.006 sec.).

Peak detection parameters: *local-dB-range* = 75dB, *general-dB-range* = 85dB, *minimum-peak-height* = 2dB, *frequency-range* = 150Hz–8000Hz.

Peak continuation parameters: *maximum-peak-deviation* = 80Hz, *peak-contribution-to-guide* = .4, *maximum-number-of-guides* = 30, *minimum-starting-guide-separation* = 90Hz, *maximum-sleeping-time* = 2 frames (.025 sec.), *length-of-filled-gaps* = 2 frames (.025 sec.), *minimum-trajectory-length* = 20 frames (.25 sec.).

1. original sound
2. deterministic synthesis
3. stochastic synthesis
4. deterministic plus stochastic synthesis
5. compression of the frequency evolution
6. compression of the frequency evolution and frequency transposition by .3
7. compression of the frequency evolution and frequency transposition by 2
8. stretch partials
9. glissando down
10. glissando up

11. time-varying change of noise component
12. time-varying time-scale
13. time-varying time-scale (inverse of previous example)
14. time-varying time-scale and time-varying stretch partials
15. change of the frequency evolution
16. inverse of the previous example
17. time expansion by 3

5.14.4 Sound example 11

Flute passage, from "Exchange" by Richard Karpen. (*sampling-rate* = 34000, length = 7 sec., lowest fundamental $\approx$ 311Hz, highest fundamental $\approx$ 1000Hz)

STFT parameters: *window-type* = Kaiser (β = 3), *window-length* = 900 samples (.026 sec.), *FFT-size* = 2048 samples, *hop-size* = 225 samples (.0066 sec.).

Peak detection parameters: *local-dB-range* = 65dB, *general-dB-range* = 75dB, *minimum-peak-height* = 4dB, *frequency-range* = 250Hz–10KHz.

Peak continuation parameters: *harmonic-sound* = true, *maximum-peak-deviation* = 200Hz, *peak-contribution-to-guide* = .4, *maximum-number-of-guides* = 15, *maximum-sleeping-time* = 2 frames, *length-of-filled-gaps* = 2 frames, *minimum-trajectory-length* = 40 frames (.26 sec.), *initial-fundamental* = 689Hz, *fundamental-range* = 300Hz–1100Hz.

1. original sound
2. deterministic synthesis
3. stochastic synthesis
4. deterministic plus stochastic synthesis
5. compression of the frequency evolution
6. frequency transposition by .5 and stretch partials
7. compression of the frequency evolution, frequency transposition by .8, time-varying time-scale, and stretch partials
8. inversion of frequency evolution

9. time compression by .4
10. time compression by .4, frequency transposition by .8, and compression of the frequency evolution

This example is particularly problematic. Many of the notes have a very strong subharmonic at half the fundamental frequency. For practical purposes we have decided not to track those components, that is, to track only the harmonics of the fundamental. Such a choice results into a lower quality synthesis, but also a simpler and a more flexible representation.

5.14.5 Sound example 12

Piano passage. (*sampling-rate* = 34000, length = 4 sec., lowest fundamental $\approx$ 140Hz, highest fundamental $\approx$ 270Hz)

STFT parameters: *window-type* = Kaiser (β = 2.8), *window-length* = 1201 samples (.035 sec.), *FFT-size* = 2048 samples, *hop-size* = 150 samples (.0044 sec.).

Peak detection parameters: *local-dB-range* = 75dB, *general-dB-range* = 85dB, *minimum-peak-height* = 2dB, *frequency-range* = 100Hz–14KHz.

Peak continuation parameters: *maximum-peak-deviation* = 60Hz, *peak-contribution-to-guide* = .4, *maximum-number-of-guides* = 45, *minimum-starting-guide-separation* = 100Hz, *maximum-sleeping-time* = 1 frames, *length-of-filled-gaps* = 1 frames, *minimum-trajectory-length* = 20 frames (.09 sec.).

1. original sound
2. deterministic synthesis
3. stochastic synthesis
4. deterministic plus stochastic synthesis
5. frequency transposition by .5
6. compression of the frequency evolution
7. time-varying stretch partials

The final number of sinusoids in the deterministic component is 35. This does not mean that we could have been able to set *maximum-number-of-guides* to 35 and obtain the same result. The 45 guides are used, but due to all the other restrictions, at the end of the peak continuation process the number of trajectories is reduced to 35.

5.14.6 Sound example 13

Singing-voice passage. (*sampling-rate* = 18000, length = 5 sec., lowest fundamental $\approx$ 115Hz, highest fundamental $\approx$ 270Hz)

STFT parameters: *window-type* = Kaiser ($\beta = 2.6$), *window-length* = 500 samples (.028 sec.), *FFT-size* = 1024 samples, *hop-size* = 125 samples (.007 sec.).

Peak detection parameters: *local-dB-range* = 70dB, *general-dB-range* = 80dB, *minimum-peak-height* = 2dB, *frequency-range* = 80Hz–8KHz.

Peak continuation parameters: *harmonic-sound* = true, *maximum-peak-deviation* = 100Hz, *peak-contribution-to-guide* = .7, *maximum-number-of-guides* = 55, *maximum-sleeping-time* = 2 frames, *length-of-filled-gaps* = 2 frames, *minimum-trajectory-length* = 5 frames (.035 sec.), *initial-fundamental* = 118Hz, *fundamental-range* = 115Hz–270Hz.

1. original sound
2. deterministic synthesis
3. stochastic synthesis
4. deterministic plus stochastic synthesis
5. frequency transposition by .8 and stretch partials
6. frequency transposition by .8 and compress partials
7. glissando and expansion of the frequency evolution
8. time-varying time-scaling
9. more percentage of stochastic component and increasing with time

More interesting transformations may be obtained if the vibrato and formant structure are extracted as independent parameters in the representation, thus being able to control them as independent variables.

5.15 Conclusions

In this chapter the final analysis/synthesis system of this dissertation has been presented. It is based on a model that considers a sound as composed of a deterministic and a stochastic component. This model results in a specific representation for each of the components. The deterministic representation includes a set of amplitude and frequency functions, one for every partial of the sound. The stochastic representation comprises a series of spectral envelopes. These envelopes describe the general frequency characteristics and the amplitude of the residual waveform, where the residual is defined as the difference between the original sound and the deterministic component. Additive synthesis is used to generate the deterministic signal. The stochastic signal is generated by performing the inverse Fourier-transform of the spectral envelopes, a process that can be thought as the filtering of white noise by these frequency envelopes.

The objective of this dissertation was to achieve a musically useful sound representation that would allow broad transformations of a variety of sounds. This representation achieves such a goal.

The analysis is the central part of the system. It is a complex set of algorithms that require the manual setting of a few control parameters. Further work may automate the analysis process, particularly if it is specialized to a group of sounds. Also, some aspects of the analysis are open to further research, in particular the peak-continuation algorithm.

The synthesis from the deterministic plus stochastic representation is simple and can be performed in real-time with current technology. A real-time implementation of this system would allow the use of this technique in performance. The representation would be precomputed and stored, and the sound transformations would be done interactively.

In this chapter some examples of the possible sound transformations have been presented, but it is beyond the scope of this dissertation to explore all the possible applications of this system. Many more sound transformations are possible, limited only by the musical intuition of the user, and applications other than sound modification have also been suggested.

Appendix A

Software and Hardware Environment

A.1 Introduction

This appendix describes the environment used for the development of the analysis/synthesis systems described in this dissertation. This environment integrates all the tools required to experiment with the techniques presented. It is very flexible and open ended, making it an ideal development environment where it is very easy to extend or change the available techniques and incorporate new ones.

The next section describes the software and hardware environment on which the program has been developed. Then follows a description of the actual program.

A.2 Description of the Environment

This research has been developed on a Lisp Machine workstation (Symbolics LM-2) and making use of an array processor (FPS AP-120B) for the signal processing calculations. The sound conversion is done with the DSC-200, 16-bit A/D and D/A converters. The software is written in Zetalisp and uses tools borrowed from SPIRE (Speech and Phonetics Interactive Research Environment).

A.2.1 The Lisp Machine

The Lisp Machine is conceptually unlike any other computer (Weinreb and Moon, 1981). It was originally developed at the M.I.T. Artificial Intelligence Laboratory as a means of effectively writing, using, and maintaining large interactive Lisp programs. The LM-2 was the first commercially available Lisp Machine, introduced by Symbolics in 1981.

The system software of the LM-2 constitutes a large-scale programming environment, with over half a million lines of system code accessible to the user. Object-oriented programming techniques are used throughout the system to provide a reliable and extensible integrated environment without the usual division between an operating system and programming languages. Zetalisp is the Lisp dialect used on the LM-2, which is closely related to the Maclisp developed in the 1970s, and to the Common Lisp specification.

The main characteristics of the LM-2 hardware are:

1. 36-bit processor
2. virtual memory
3. high resolution black and white display
4. color display
5. mouse
6. dedicated 300 Mbyte disc drive
7. Chaos network
8. Unibus

At CCRMA there are four LM-2s on the Chaos network. They share a tape drive for permanent storage and are connected to the main-frame computer of the center (Foonly F4) via Ethernet. Through the F4 the LM-2s have access to several printing devices and other peripherals. The DSC-200 converters and the array processor are connected to the Unibus of one of the machines. Figure A.1 shows the hardware configuration of the system.

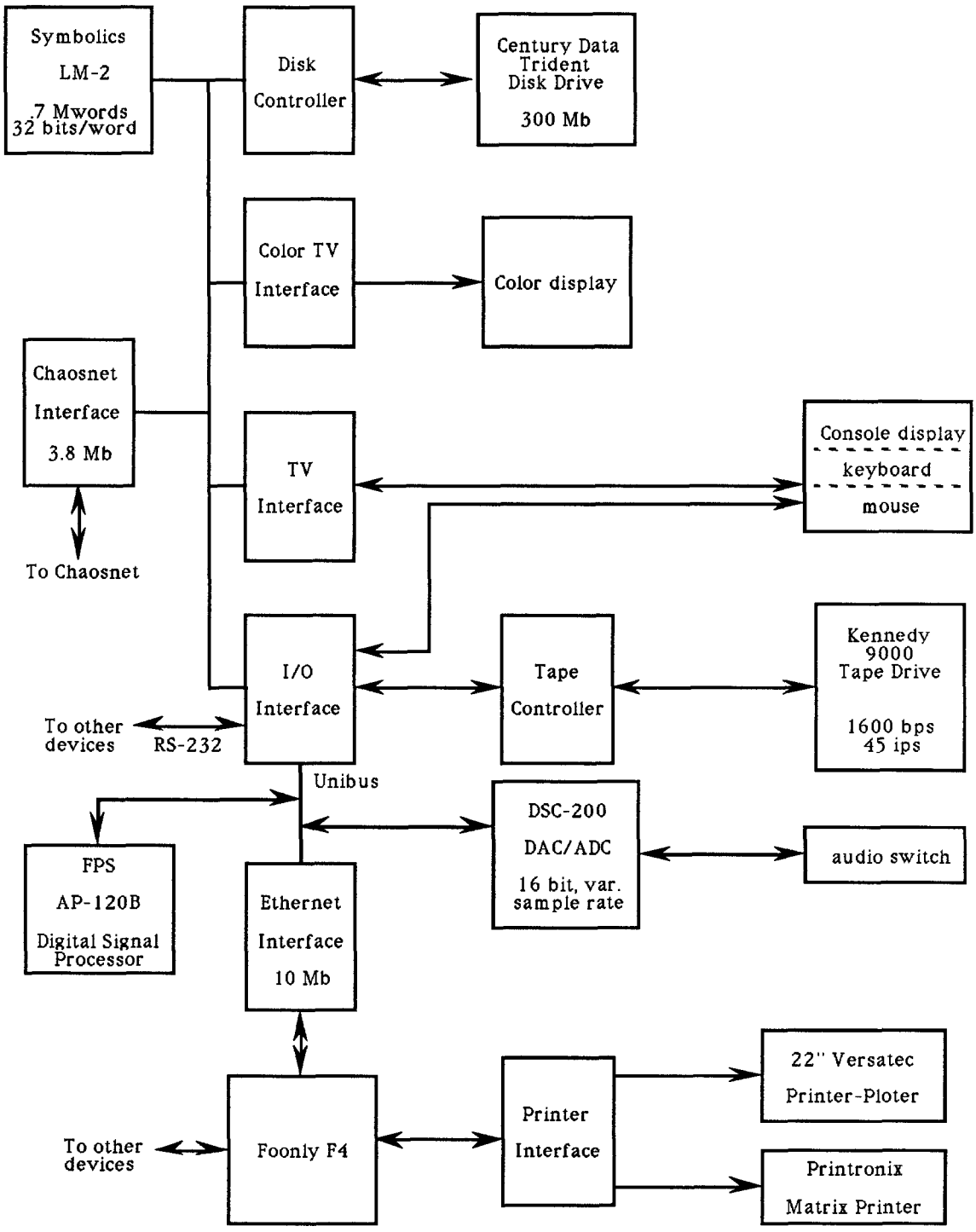

Figure A.1: Hardware configuration of the overall system.

A.2.2 Flavors

This environment, like all the LM-2 software, makes extensive use of objects, a programming style which was first used in the Smalltalk and Actor families of languages.

Object-oriented programming deals with objects, which are instances of types, and generic operations defined on those types. The definition of a type is done by defining the data known to the type and the operations that are valid for those data. Then an instance of that type can be created. Each instance maintains a local state and has an interface to the world through the defined operations. Thus, in object-oriented programming, data and procedures are encapsulated within an instance of the type.

The support of object-oriented programming on the LM-2 is done through a collection of language features known as the Flavor system. *Flavors* are the abstract types; *methods* are the generic operators. The objects are *flavor instances* that are manipulated by sending *messages,* which are requests for specific operations.

The flavor dependencies form a graph structure; they are not constrained to be hierarchical as in some languages that support an object-oriented style. Figure A.2 shows an example of flavor dependencies.

A.2.3 The Array Processor

Connected to the Unibus of the Lisp Machine is the array processor (AP-120B). The AP-120B (from Floating Point Systems, Inc.) is a high-speed (167-ns cycle time) peripheral floating-point Array Processor, which works in parallel with the host computer. Its internal organization is particularly well suited to performing the large numbers of reiterative multiplications and additions required in digital signal processing. The highly parallel structure of the AP-120B allows the "overhead" of array indexing, loop counting, and data fetching from memory to be performed simultaneously with arithmetic operations on the data. This allows much faster execution than on a typical general-purpose computer, where each of the above operations must occur sequentially.

The AP-120B comes with a Math Library which includes over 350 routines covering a wide range of array processing needs. These routines, written in AP Assembly Language, can be called by functions on the Lisp Machine or other programs written in AP Assembly Language. The AP performs arithmetic operations using a 38-bit floating-point format: one exponent sign bit, nine exponent bits, one mantissa sign bit, and 27 mantissa bits. The binary point is always located between the mantissa sign bit and the most significant bit of the mantissa.

The combination of the Lisp Machine and the Array Processor allows one to maintain a high level of both numeric and symbolic processing power, which is very appropriate for our application.

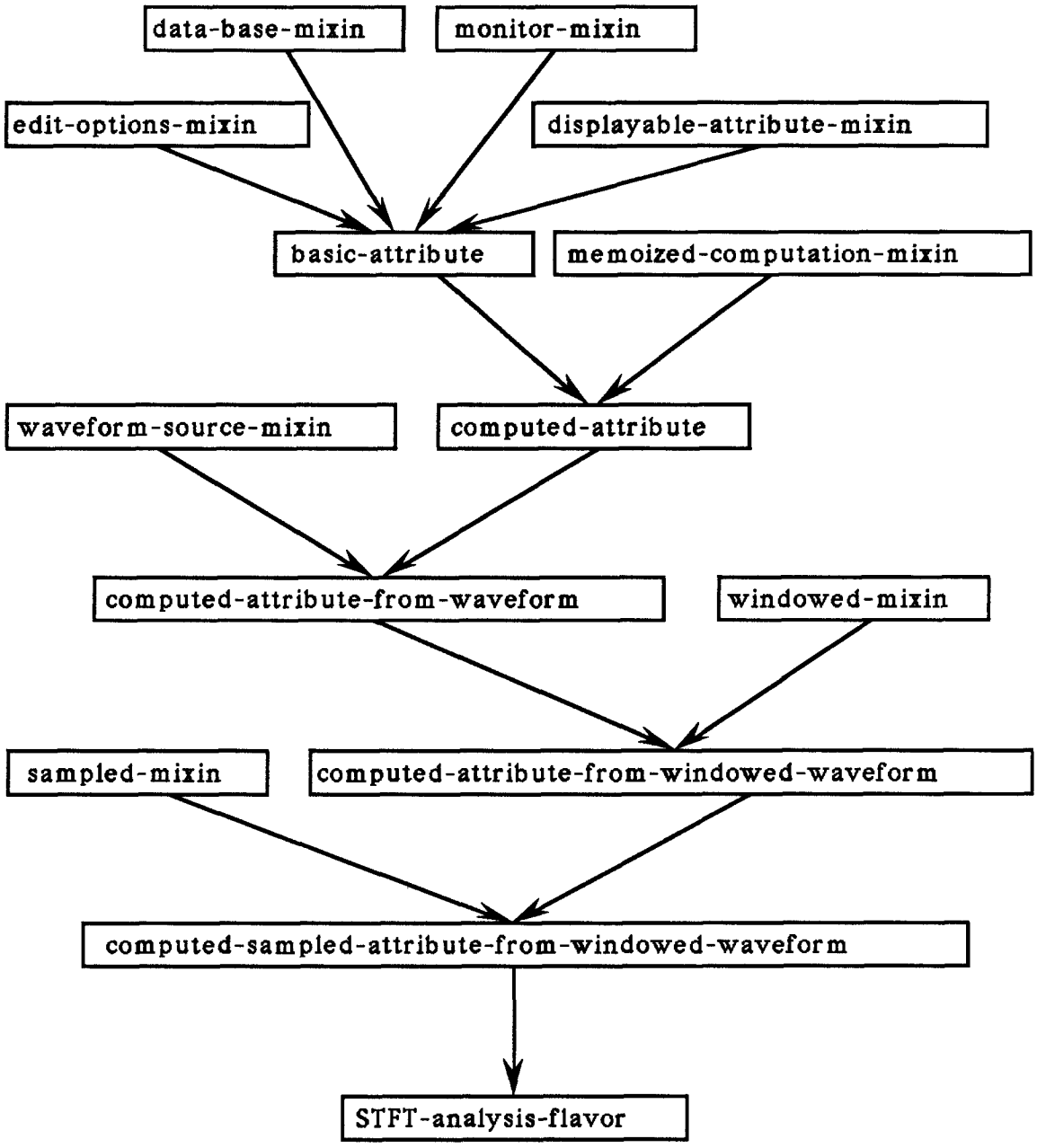

Figure A.2: Example of flavor dependencies.

A.2.4 The D/A and A/D converters

The DSC-200 includes a 16-bit digital-to-analog and analog-to-digital converters. It plugs into the Unibus of the Lisp machine and provides a link between standard audio equipment and the computer. It has variable sampling rate up to 60KHz and the machine in use at CCRMA includes two anti-aliasing low-pass filters, with cut-off frequencies of 8KHz and 16KHz.

A.2.5 SPIRE

SPIRE is the Speech and Phonetics Interactive Research Environment, which runs on Symbolics Lisp Machines (Shipman, 1982; Roads, 1983; Cyphers, 1985; Kassel, 1986). It is a program for manipulating speech signals and computations on those signals interactively. In addition, it can be used as a basis for developing other speech processing systems and can be extended and customized by the user to perform specific tasks.

SPIRE was implemented by David Shipman at MIT in 1982. Since then the program has been modified by many members of the Speech Communication Group of MIT and runs on the recent models of Lisp Machines built by Symbolics. The original SPIRE was designed for collecting speech data and looking at transformations of it, but the recent versions have become more general and allow other applications. On the LM-2 the last version of the SPIRE that can be run is the 1984 one. There are a few changes in Zetalisp from the time of the LM-2s, and SPIRE is supported only for the last releases of the Lisp Machine software.

SPIRE's basic tools can be used on a system for the analysis, transformation and resynthesis of musical sounds. However some tools are very specific to Speech and have been changed or completely rewritten, and some others that would be useful are not available in SPIRE. Therefore our program is a combination of unmodified parts of SPIRE, sections that have been rewritten, plus code written from scratch.

A.3 Description of the Program

The program is entirely written in Zetalisp. It makes extensive use of the Flavor system available on the LM-2. The graphic interface is based on the display system from SPIRE. The signal processing computations use the AP-120B and the collection of array processing utilities that come with it. The control of the AP-120B has also been borrowed from SPIRE.

The program is divided into two parts, the computation system and the display system. The basic data structure is the *utterance.*

A.3.1 Utterance

An *utterance* (term borrowed from speech and not very appropriate for music) is the basic data structure of the program. The utterance is implemented as an object and groups together information related to a single sound, including:

1. a digital representation of the sound
2. a set of computations based on the digitized waveform, called attributes
3. a pathname where the utterance is stored

The acoustic signal is digitized with the A/D converters and then stored as the *original waveform* of an utterance. All the analysis, transformations, documentation, or any kind of data that we may think of are stored with the utterance in the form of attributes.

A.3.2 Computation system

The computation system is responsible for analyzing, transforming, resynthesizing, and generally, manipulating utterances. This includes the control of the array processor, and all the signal-processing tools required for the computations.

The control of the array processor is borrowed from SPIRE and includes an assembler, a debugger, a library of AP routines supplied by the manufacturer, and software tools for loading the programs into the AP and transferring data to and from the AP.

Routines for the Array Processor can be written in AP-120B assembly language or in FPS-Lisp. FPS-Lisp is a highly-constrained Lisp subset which compiles into AP assembly language. The FPS-Lisp facility is primarily used to chain together or iterate through sequences of precoded routines which are available in the Math Library. For most of our purposes the FPS-Lisp facility is sufficient and there is no need to write in assembly language.

Signal processing tools are built on top of the low level AP routines and include all the algorithms described throughout this dissertation.

The computation results are called *attributes*, which are objects, or flavor-instances. They are computed by having messages sent to them. For example, an FFT is an instance of the flavor called "FFT-flavor." The instance is first created and then the FFT computed by sending messages to the instance with the input waveform and the values of its control variables. An attribute may receive a computing message from the display system, from another attribute which requires its data, or as a specific function call from outside the environment.

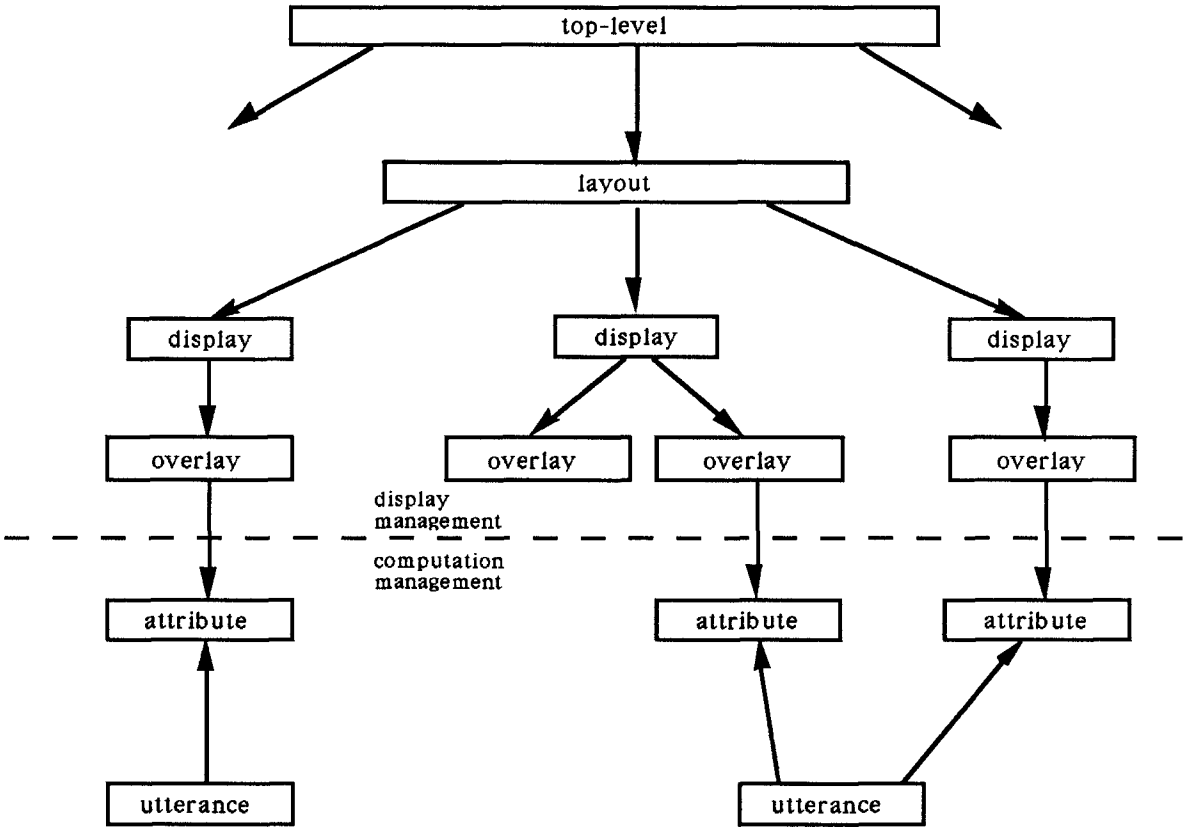

Figure A.3: Example of a Display system.

A.3.3 Display system

The display system gives a complete, interactive control over the computation system and utterances. It is based on the window system of the Lisp Machine which is a very powerful tool for dealing with displays.

The system has three different window levels. The higher one is called the *layout*, the middle one the *display*, and the bottom one the *overlay*. The data of the attributes is displayed on the overlays. Figure A.3 shows a typical organization of the display system.

The overlays are the simplest display objects that the program manipulates. They describe how the values are drawn on the screen. They come in two varieties:

1. attribute overlays, which draw the values of an attribute
2. background overlays, which draw utterance-independent annotations

The background overlays are mainly used to display the two different kinds of cursors that are available plus background grids or axis. The overlays are transparent. Two overlays can occupy the same area of the screen and both are drawn.

A display is a rectangular area of the screen containing one or more overlays. Unlike overlays, displays are not transparent. Partially covered displays are hidden from view.

A display contains information which can be accessed by its overlays. For example, we may want to draw two overlays on a common axis. The scale and position of each overlay's axis default to the values stored in the containing display.

The overall screen is managed through layouts. Each layout specifies a collection of displays and their positions. There can be any number of layouts, but only one is displayed at any time. Some layouts have been designed in advance and come with a set of displays and overlays to be used for a specific task. For example there is a layout to manage the recording of sounds with the A/D converter and another to study the problem of time-domain windowing of waveforms. But the normal layouts are called "blank" layouts. On these, the user defines the structure of displays and overlays interactively during every particular session. FigA.4 shows an example of a layout.

A.4 Conclusions

This appendix has presented the environment on which this dissertation has been developed. It has proved to be an excellent workbench for the processing of sounds and for the development of new signal processing algorithms.

Although this environment has been designed with the objective of developing the algorithms presented in this dissertation, it is also used for various related projects. A particularly relevant application is as a workbench for composers. Using the environment, sounds are transformed with any of the analysis/synthesis systems available, which then, with the help of other programs, are integrated into musical compositions.

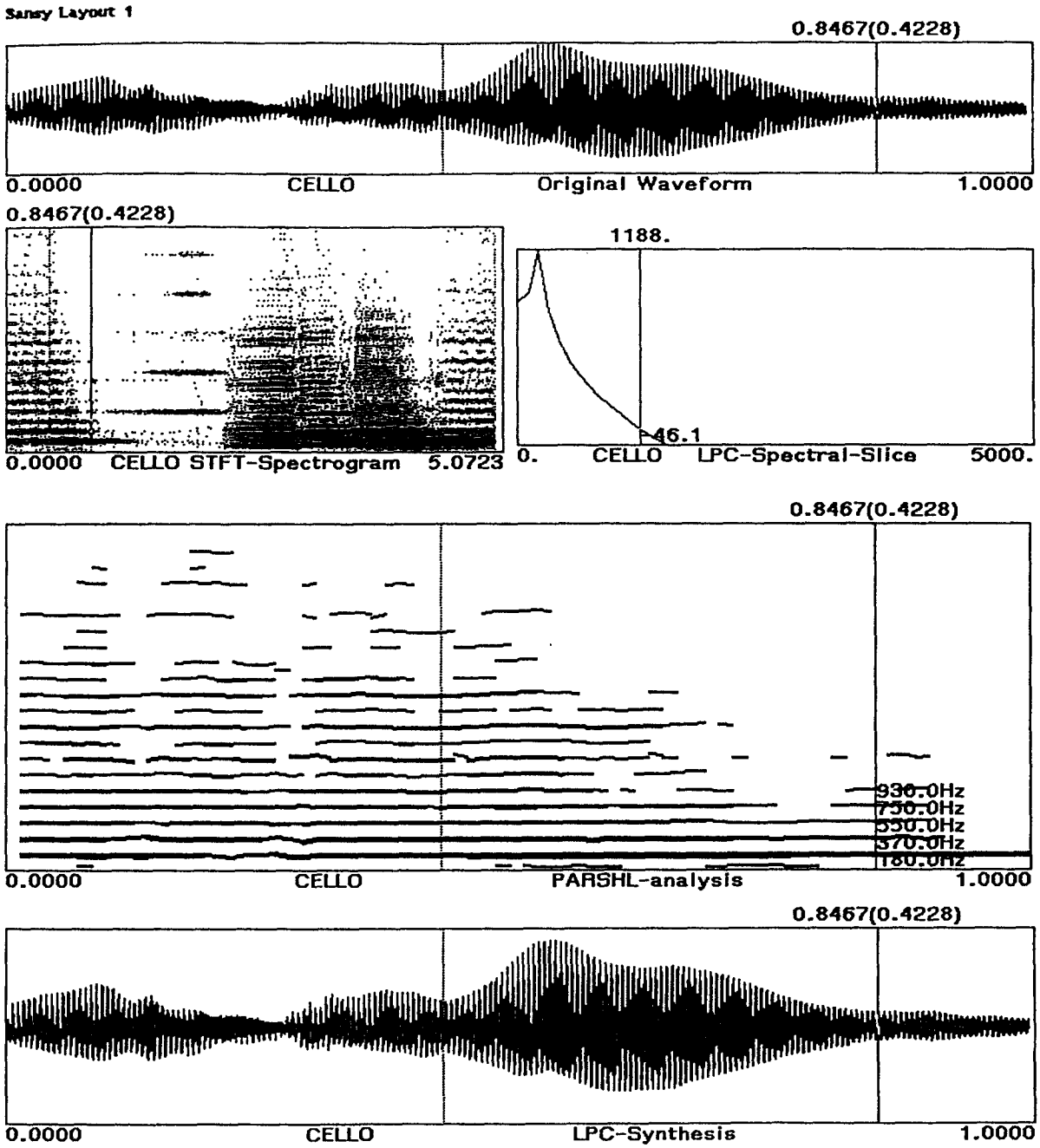

Figure A.4: Example of a layout.

Appendix B

Cross-synthesis

B.1 Introduction

The technique used for the stochastic analysis and synthesis can apply the spectral characteristics of one sound on to another one, thus creating the effect of a hybridization. This process is traditionally called cross-synthesis (Moorer, 1979).

The hybridization of two sounds by applying the spectral characteristics of one sound to another one goes back to the 1930s with the work of Dudley (Dudley, 1939). At that time this was accomplished with an analog vocoder. More recently, the digital method of linear predictive coding, LPC (discussed in the next appendix) has been used for that purpose in music applications.

The cross-synthesis technique presented in this chapter is more flexible than the LPC implementation. It offers a lot of control on the process, allowing the creation of a wide variety of hybridization effects.

B.2 Description of the System

Figure B.1 shows a diagram of the cross-synthesis system and Figure B.2 includes a graphic example. A spectral envelope of a sound is obtained by computing a set of magnitude spectra using the STFT and then performing a line-segment approximation on each spectrum. Each envelope is then applied to the corresponding magnitude spectrum of another sound; where this spectrum has also been obtained from a STFT analysis. This process returns a set of modified magnitude spectra. Then, the hybridized sound results from an inverse STFT of the modified magnitude spectra and the untouched phase spectra of the second sound.

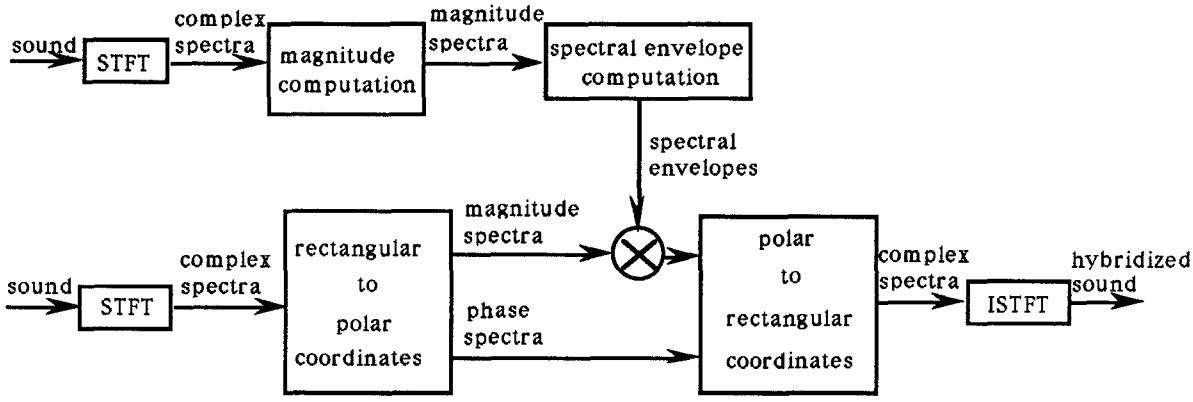

Figure B.1: General diagram of the cross-synthesis system.

B.3 Applying a Spectral Envelope to a Magnitude Spectrum

All the steps involved in this system (Figure B.1) are discussed in previous chapters, except the actual product of the spectral envelope of one sound with the magnitude spectrum of the other one. This process corresponds to the filtering of a sound by a time varying filter. For clarity purposes we call excitation to sound to which the envelopes are applied, and modulating sound to the other sound.

To explain this process let us consider $\hat{E}_l(k)$ a magnitude-spectrum envelope at frame l of the modulating sound, in dB, and $A_l(k)$ a magnitude spectrum, also in dB, of the excitation. Since we are in a log scale (dB), to apply the envelope to the excitation spectrum we simply add the two together. The only problem is the scaling of the result. If what we want is to keep the magnitude of $A_l(k)$ in the result, that is, to maintain the magnitude of the excitation, the envelope $\hat{E}_l(k)$ is modified to have an average of 0dB; done by subtracting its current average,

$$\bar{E}_l(k) = \hat{E}_l(k) - a_l \tag{B.1}$$

where

$$a_l = \frac{1}{N} \sum_{k=0}^{N-1} \hat{E}_l(k) \tag{B.2}$$

is the average dB-level of one spectral frame.

The cross-synthesized magnitude spectrum is then,

$$B_l(k) = A_l(k) + \bar{E}_l(k) \tag{B.3}$$

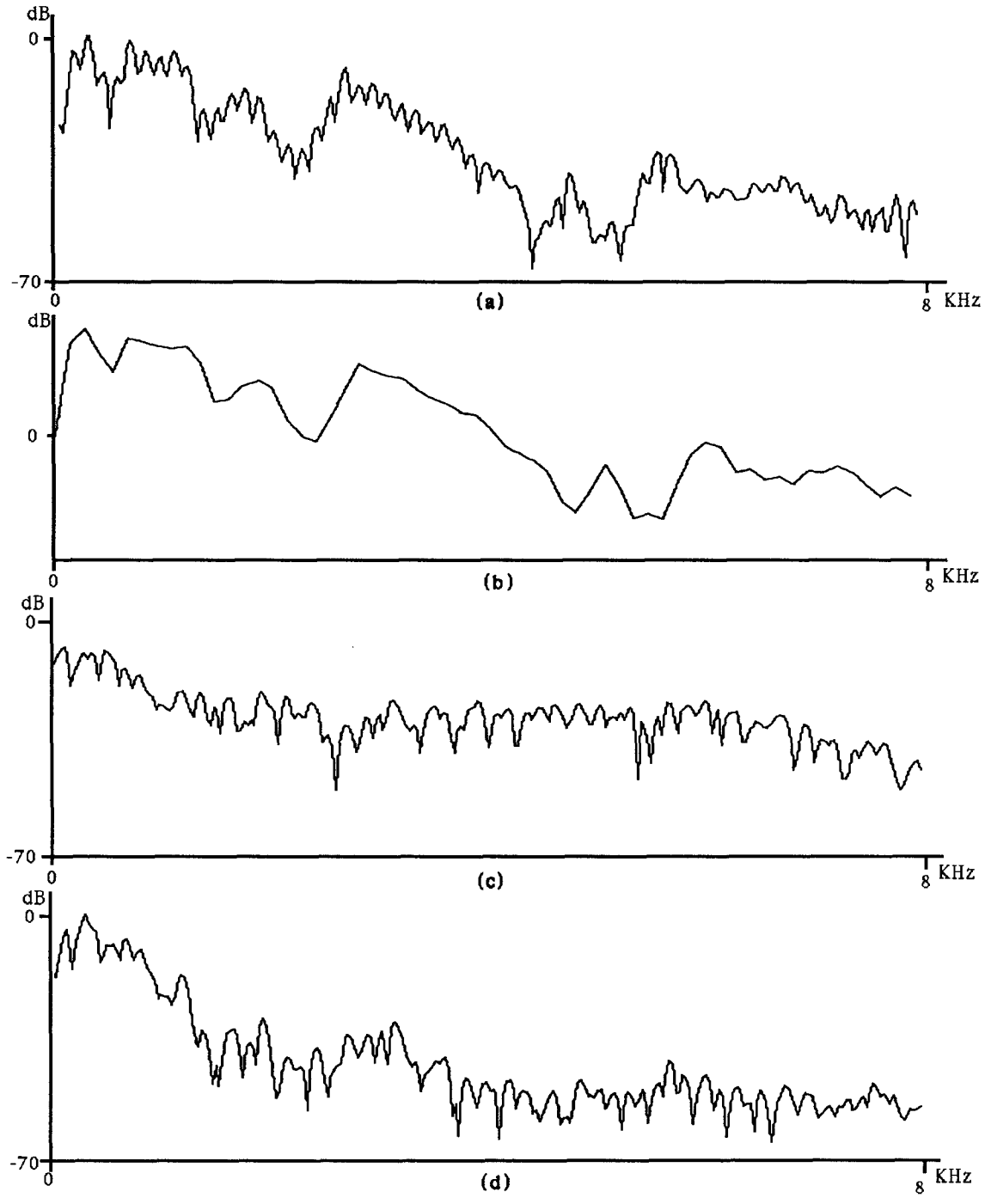

Figure B.2: Cross-synthesis example: (a) magnitude spectrum of a speech sound (modulator), (b) computed spectral envelope, (c) magnitude spectrum of a rain sound (excitation), (d) hybridized spectrum.

This method gives a sound with the amplitude evolution of the excitation. However, it is also possible to maintain the amplitude of the modulating sound in the cross synthesis, or any amplitude in between the two sounds. For this purpose both the envelope and the magnitude spectrum are normalized to have an average of 0dB using equation B.1, then added,

$$\bar{B}_l(k) = \bar{A}_l(k) + \bar{E}_l(k) \tag{B.4}$$

where $\bar{B}_l(k)$ is the normalized result with a 0dB average. Then $\bar{B}_l(k)$ is scaled by

$$c_l = \alpha a_l + \beta b_l, \qquad \alpha\epsilon[0,1],\ \beta\epsilon[0,1] \tag{B.5}$$

where a_l is the average of the envelope at frame l, b_l the average of the corresponding magnitude spectrum, and α and β are parameters that control the balance between the two amplitudes. For example, when $\alpha = 0$ and $\beta = 1$ the cross-synthesized sound has the amplitude of the excitation. Enough headroom has to be left in order not to overflow the result when α and β sum to a number bigger than 1.

B.4 Summary of the Technique

A review of the steps for a computer implementation is given below.

1. Perform the STFT of a sound $x(n)$ (modulating sound) with specific values for *window-type, window-length, FFT-size*, and *hop-size*,

$$X_l(k) \triangleq \sum_{n=0}^{N-1} w(n)x(n+lH)e^{-j\omega_k n}, \tag{B.6}$$

   where $w(m)$ is the analysis window, l the frame number, and H the *hop-size*. The result is a series of complex spectra from which only their magnitude $|X_l(k)|$ is computed.

2. Compute the envelope for each magnitude spectrum by performing a line-segment approximation,

$$\tilde{E}_l(q) = \max_k(|X_l(qM+k)|), \qquad k = -M/2, -M/2+1, \ldots, 0, \ldots, M/2-2, M/2-1, \quad q = 0, 1, \ldots, N/M-1 \tag{B.7}$$

   where M is the step size and the window size (or size of the section), $\tilde{E}_l(q)$ is the maximum of section q and frame l.

3. Perform the STFT of another sound $y(n)$ (i.e., excitation), with same *window-type, window-length, FFT-size*, and *hop-size* as for sound $x(n)$,

$$Y_l(k) \triangleq \sum_{n=0}^{N-1} w(n)y(n+lH)e^{-j\omega_k n}, \tag{B.8}$$

If length of sound $y(n)$ is different from the one of $x(n)$, the *window-length* and *hop-size* are changed in order to have the same number of frames in both analysis.

4. Compute its magnitude spectrum in dB,

$$A_l(k) = 20\log|Y_l(k)| \tag{B.9}$$

5. Interpolate each line-segment approximation $\tilde{E}_l(q)$ to a curve of size $N/2$, where N is the *FFT-size* used for sound $y(n)$, convert it to dB, and normalize it to have a 0dB average.

6. Obtain the cross-synthesized magnitude spectrum $B_l(k)$ by adding the normalized envelope $\bar{E}_l(k)$ (or its modification) and the magnitude spectrum $A_l(k)$,

$$B_l(k) = A_l(k) + \bar{E}_l(k) \tag{B.10}$$

7. Compute the complex spectrum $S_l(k)$ from the phase spectrum of $Y_l(k)$ and the modified magnitude spectrum $B_l(k)$,

$$\begin{aligned} \mathrm{Re}\{S_l(k)\} &\triangleq B_l(k)\cos[\angle Y_l(k)] \\ \mathrm{Im}\{S_l(k)\} &\triangleq B_l(k)\sin[\angle Y_l(k)] \end{aligned} \tag{B.11}$$

8. Compute the inverse-FFT of S_l,

$$s_l(m) = \frac{1}{N}\sum_{k=-N/2}^{N/2-1} S_l(k)e^{j\omega_k m} \tag{B.12}$$

9. Obtain the hybridized sound by overlapping and adding the output frames,

$$s(n) = \sum_{l=0}^{L-1} s_l(n-lH) \tag{B.13}$$

B.5 Examples

Speech is very appropriate as the source for the spectral envelopes. The excitation can be any sound that is spectrally rich. Next, a set of examples are presented that use the same speech phrase as the spectral envelope, and each one uses a different excitation.

B.5.1 Sound example 14

Speech phrase hybridized with other sounds (*sampling-rate* = 18000).

Analysis parameters: *window-type* = Kaiser (β = 2.5), *window-length* = 400 samples (.022 sec.), *FFT-size* = 512 samples, *hop-size* = 100 samples (.005 sec.), *number-segments-in-spectral-approximation* = 50.

1. speech sound
2. cat sound
3. cross-synthesis of cat with speech
4. cow sound
5. cross-synthesis of cow and speech
6. gong sound
7. cross-synthesis of gong and speech
8. plane sound
9. cross-synthesis of plane and speech
10. ship creaking sound
11. cross-synthesis of ship creaking and speech
12. modification of the spectral envelopes on the previous example
13. another modification of the spectral envelopes

In all these examples the length of the excitation is kept unchanged. The sequence of envelopes of the speech signal are stretched or compressed in time to accommodate to the length of the excitation.

B.6 Conclusions

In this appendix we have presented a sound hybridization technique that is more flexible than the traditional cross-synthesis implementations based on LPC. With it, the spectral and/or amplitude characteristics of one sound are applied to another sound. This process is most successful when the modulating sound has a pronounced formant structure, and when the excitation sound has a very rich spectrum.

Appendix C

Use of LPC to Model the Stochastic Component

C.1 Introduction

The representation of the stochastic component was obtained in Chapter 5 by performing a line-segment approximation to the magnitude-spectrum residual. An alternative is to fit an nth-order polynomial to the spectral residual using linear predictive coding, LPC (Makhoul, 1975; Markel and Gray, 1976; Rabiner and Schafer, 1978; Moorer, 1979). Even though this alternative loses some of the flexibility of the line-segment approximation approach, it has the advantage of being a simpler representation (i.e., less data points).

LPC has become the standard technique in speech research for estimating the basic parameters, e.g., pitch, formants, spectra, vocal tract area functions, and for representing speech for low bit-rate transmission and storage. In this appendix LPC is used to obtain a time-varying all-pole filter that matches a set of magnitude-spectrum residuals. This technique is successful even though in our application the characteristics of the spectra are very different from the ones obtained in traditional speech applications.

In this appendix the linear prediction model is first presented. Then a particular solution, called the autocorrelation method, is discussed, followed by its application to the residual approximation problem. This appendix ends with the synthesis of the stochastic signal from the LPC parameters and conclusions.

C.2 Linear Prediction Model

This model assumes that a signal $x(n)$ is the result of a linear combination of past values and some input $u(n)$,

$$x(n) = -\sum_{k=1}^{p} a_k x(n-k) + Gu(n) \tag{C.1}$$

where G is a gain factor.

Given a particular signal $x(n)$, the problem is to determine the predictor coefficients a_k and the gain G in some manner. Because of the time-varying nature of most sounds, the predictor coefficients are estimated from short segments of the signal.

C.3 Solution of the Model

The standard LPC formulation assumes that the input $u(n)$ is unknown white noise or an impulse train whose impulses are separated by more than p samples. In the noise-driven case the signal $x(n)$ can be predicted only approximately from a linearly weighted summation of past samples. Let this approximation of $x(n)$ be $\hat{x}(n)$, where

$$\hat{x}(n) = -\sum_{k=1}^{p} a_k x(n-k) \tag{C.2}$$

Then the error between the actual value $x(n)$ and the predicted value $\hat{x}(n)$ is given by

$$e(n) = x(n) - \hat{x}(n) = x(n) + \sum_{k=1}^{p} a_k x(n-k) \tag{C.3}$$

where the problem is to minimize the error, normally done by minimizing the mean or the total squared error with respect to each of the parameters. There are several ways of doing it. The one discussed here is the autocorrelation method (for other methods see: Makhoul, 1975; Markel and Gray, 1976). This approach assumes that the frame $\tilde{x}_l(n)$ is 0 outside the interval $0 \leq n \leq N-1$, where N is the length of the window used to perform the short-time analysis. The windowing process corresponds to the one used for the STFT, and the discussion of Chapter 2 applies here.

Denote the squared error by E and the windowed frame by $\tilde{x}_l(n)$, then

$$E = \sum_{n=-\infty}^{\infty} e(n)^2 = \sum_{n=-\infty}^{\infty} \left(\tilde{x}_l(n) + \sum_{k=1}^{p} a_k \tilde{x}_l(n-k) \right)^2 \tag{C.4}$$

E is minimized by setting

$$\frac{\partial E}{\partial a_i} = 0, \qquad 1 \leq i \leq p \tag{C.5}$$

From (C.4) and (C.5) we obtain the set of equations:

$$\sum_{k=1}^{p} a_k R(i-k) = -R(i), \qquad 1 \leq i \leq p \tag{C.6}$$

and

$$E_p = R(0) + \sum_{k=1}^{p} a_k R(k) \tag{C.7}$$

where

$$R(i) = \sum_{n=-\infty}^{\infty} \tilde{x}_l(n)\tilde{x}_l(n+i) \tag{C.8}$$

is the autocorrelation of the signal $\tilde{x}_l(n)$.

The predictor coefficients a_k are obtained by solving the set of equations (C.6), p equations with p unknowns. There are many methods of solving these equations. Even though the most popular and well known of these methods are the Levinson and Robinson algorithms (Markel and Gray, 1976), the most efficient method known for solving this system of equations is Durbin's recursive procedure (Makhoul, 1975; Rabiner and Schafer, 1978).

C.4 Modeling the Stochastic Component

In Chapter 5 it is shown how the subtraction of the deterministic component from the original sound results in a set of magnitude-spectrum residuals. The stochastic representation is then obtained by fitting an envelope to each one of these spectra. In Chapter 5 this is done with a line-segment approximation; here LPC is used.

The LPC model and its solution can either be formulated in the time or frequency domain (in the previous section it was presented in the time domain for simplicity). The only practical difference between the two is whether the autocorrelation function $R(i)$, which is an intermediate step, is calculated from the time signal $x(n)$ or from the spectrum $X(k)$. Since the residual to be approximated is obtained in the frequency domain, it is convenient to calculate $R(i)$ from the spectrum. This is done by performing the inverse Fourier transform of the power spectrum,

$$R_l(n) = \frac{1}{N} \sum_{k=-N/2}^{N/2-1} |X_l(k)|^2 e^{j\omega_k n} \tag{C.9}$$

where $|X(k)_l|^2$ is proportional to an estimate of the power spectrum of a windowed portion of $x(n)$.

The LPC coefficients a_k are then obtained by solving the set of equations (C.6). Figure C.1 shows how these coefficients match the residual spectrum. (The frequency domain envelope shown in Figure C.1 is the Fourier transform of the coefficients.)

C.5 Synthesis of the Stochastic Component

The stochastic signal is synthesized by filtering a white noise signal with the time-varying all-pole filter represented by the LPC coefficients. The simplest implementation is to use equation (C.1), where $u(n)$ is unit variance white noise with zero-mean, and G, the amplitude of the noise, is given by equation (C.7). The resulting synthesis structure, shown in Fig. C.2, is the direct form of the all-pole filter. The gain G is linearly interpolated from frame to frame, and the coefficients are updated at every frame.

The direct form structure may have stability problems, specially when the coefficients are interpolated from frame to frame. A very attractive alternative is to use lattice filters (Markel and Gray, 1976). An example is shown in Fig. C.3. The multipliers, k_i, are reflection coefficients and can be interpolated without fear of instabilities. The conversion from direct-form coefficients to reflection coefficients is performed by a recursive procedure which is discussed by Markel and Gray (Markel and Gray, 1976).

C.6 Conclusions

In this appendix an alternative representation for the stochastic component of the sound has been presented. This alternative uses the well known LPC method to fit an nth-order pole filter on every magnitude-spectrum residual. Compared with the line-segment approximation approach, the LPC method has the advantage that it reduces the number of data points in the stochastic representation. However, it has an important drawback, it is not a flexible representation. Compared with the line-segments, the LPC-coefficients are very difficult to modify. Another disadvantage is that the LPC analysis and synthesis process is more sensitive to numerical errors than a Fourier-based technique.

The decision as to which stochastic representation to use depends on the application. If the main concern is flexibility, the line-segment approximation is clearly the choice, however, if data reduction is the main priority, the LPC method may be a better choice.

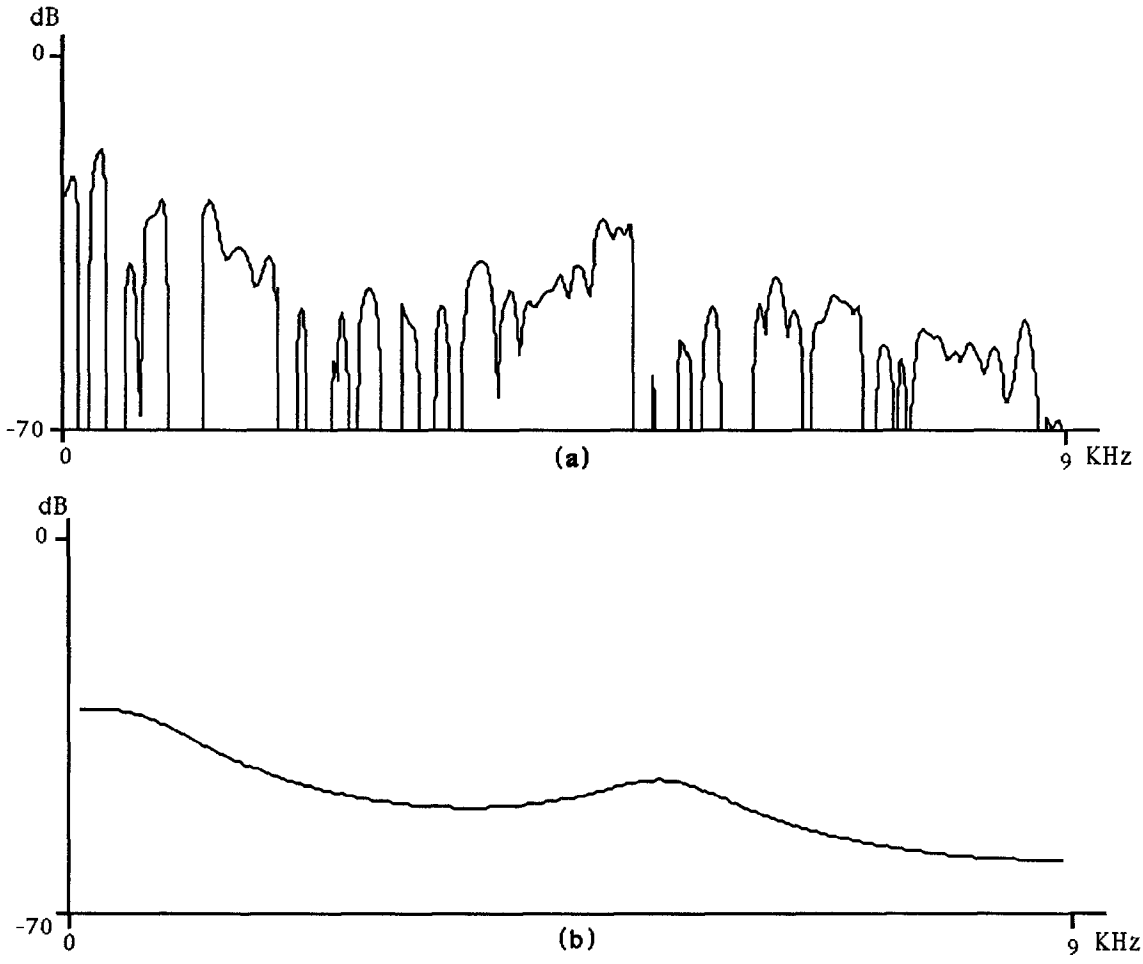

Figure C.1: Approximation of a magnitude-spectrum residual using LPC: (a) residual spectrum from a piano note attack, (b) LPC approximation of the residual.

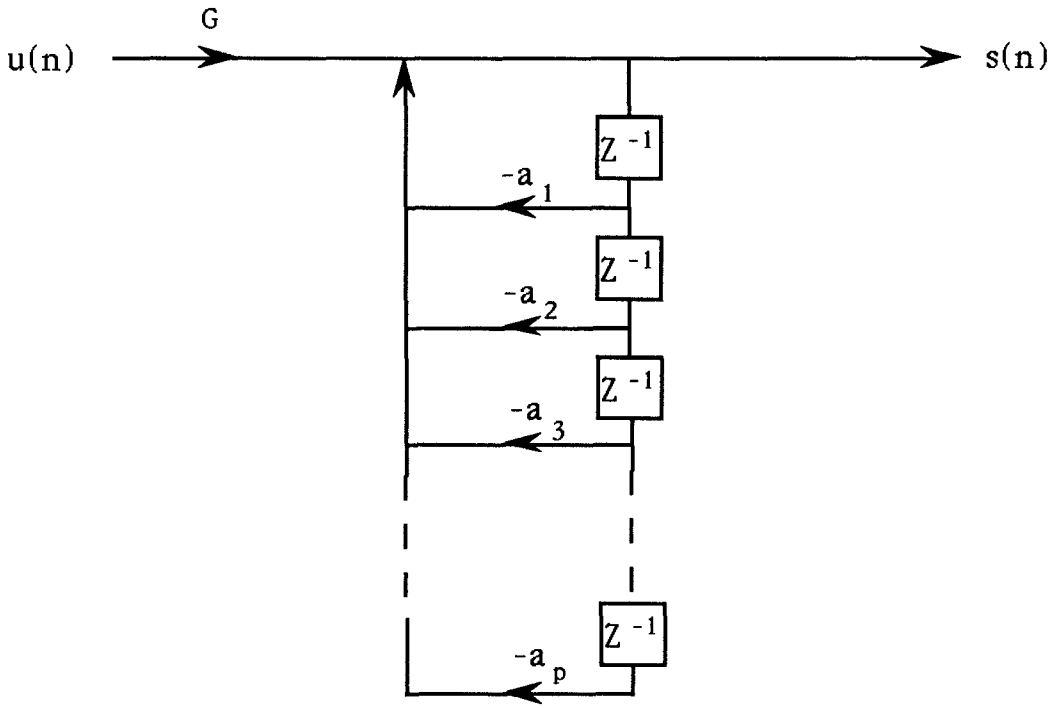

Figure C.2: Direct form realization of an all-pole filter.

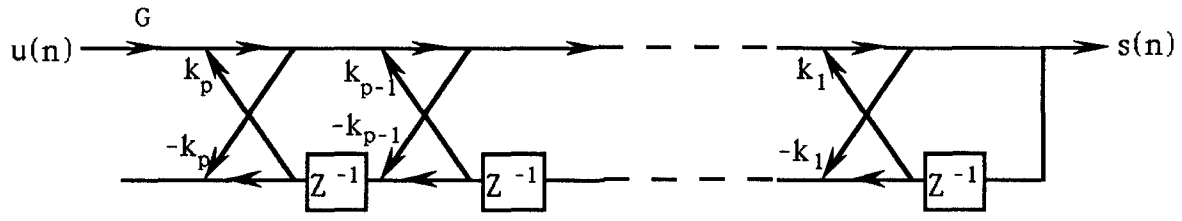

Figure C.3: Lattice form realization of an all-pole filter.

Appendix D

Deterministic Synthesis with Original Attack

D.1 Introduction

A simple extension of the deterministic analysis/synthesis performed in Chapter 4 is the splicing of an original attack into the deterministic synthesis.

Most instrumental sounds have a noisy attack and a fairly periodic steady-state and decay portions. Therefore, except for the attack, the deterministic analysis is able to capture most of the sound characteristics. Then, a more realistic attack can be obtained by splicing the original attack into the synthesized sound. This is not possible with most synthesis techniques because the synthesized waveform does not match the phases of the original sound. But in the deterministic analysis performed in Chapter 4 the splicing is successful because the amplitude, frequency, and phase of every partial are tracked. With this technique the synthesized sound preserves the phase of the original waveform during the steady-state and it is possible to splice both sounds together at a single sample (Figure D.1).

When the part of the waveform where the splice is done is not stable enough, a cross-fade of a few samples, done with a simple Hanning window, may be necessary. In fact, the cross-fade is even successful when the phase is not tracked (magnitude-only analysis/synthesis) if the original sound is very periodic and the cross-fade is chosen to be a few periods long.

D.2 Examples

The splicing of an original attack into its synthesized version is most effective with sounds that have very sharp attacks. Here examples are shown with a marimba and a piano.

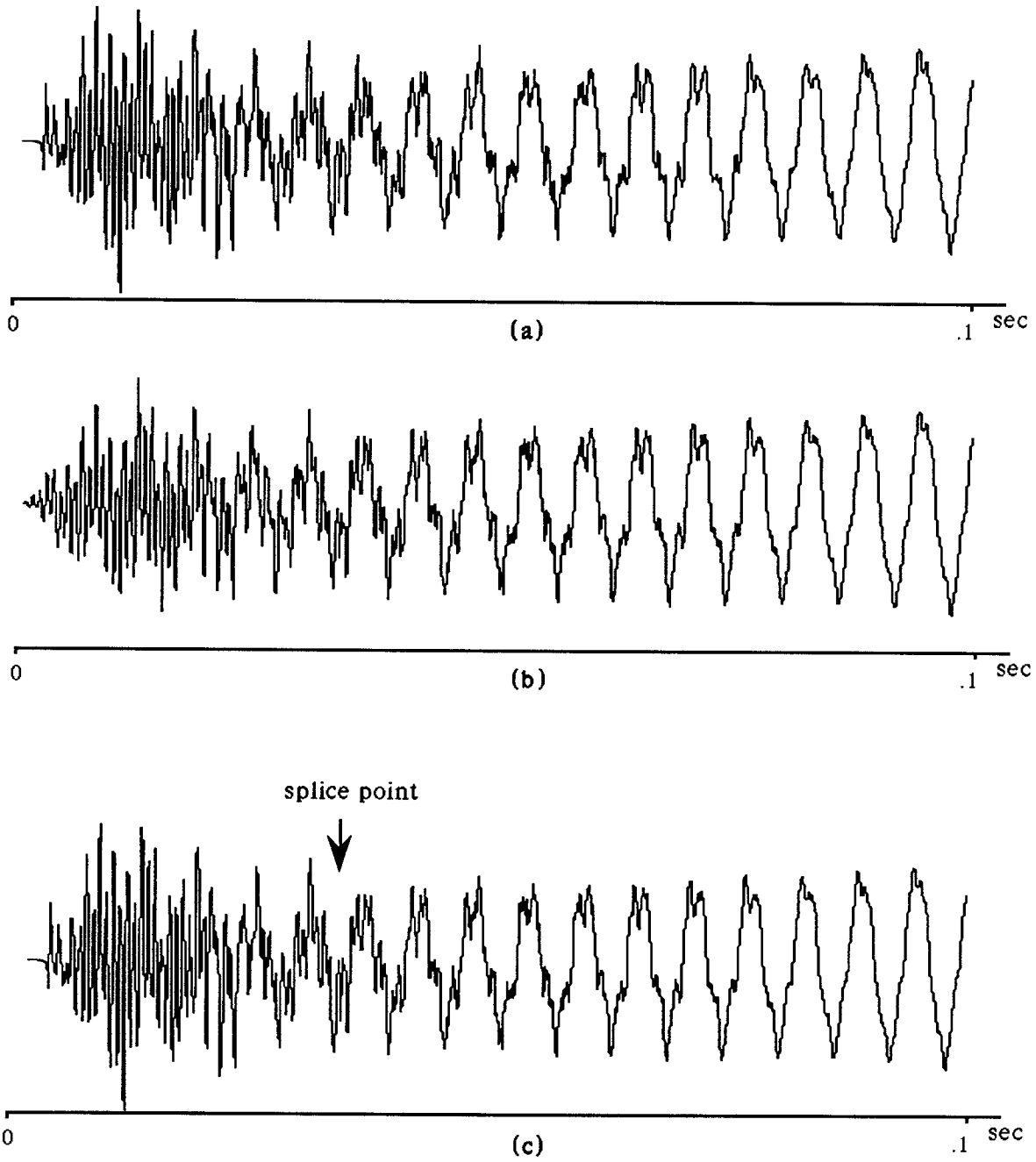

Figure D.1: Example of deterministic synthesis with original attack: (a) marimba tone, (a) deterministic synthesis with phase tracking, (c) deterministic synthesis with original attack.

D.2.1 Sound example 15

Marimba note. (*sampling-rate* = 34000, length = 2 sec.)

STFT parameters: *window-type* = Kaiser (β = 2.8), *window-length* = 1001 samples (.03 sec.), *FFT-size* = 2048 samples, *hop-size* = 500 samples (.014 sec.).

Peak detection parameters: *local-dB-range* = 75dB, *general-dB-range* = 85dB, *minimum-peak-height* = 2dB, *frequency-range* = 80Hz–10KHz.

Peak continuation parameters: *maximum-peak-deviation* = 80Hz, *peak-contribution-to-guide* = .2, *maximum-number-of-guides* = 20, *minimum-starting-guide-separation* = 120Hz, *maximum-sleeping-time* = 0 frames, *length-of-filled-gaps* = 0 frames, *minimum-trajectory-length* = 5 frames (.037 sec.).

1. original sound
2. deterministic synthesis
3. deterministic synthesis with original attack

Note that the *hop-size* is 1/2 the *window-length*, not 1/4 as in the examples of the previous chapters.

D.2.2 Sound example 16

Piano note. (*sampling-rate* = 34000, length = 1 sec.)

STFT parameters: *window-type* = Kaiser (β = 2.8), *window-length* = 1201 samples (.035 sec.), *FFT-size* = 2048 samples, *hop-size* = 600 samples (.018 sec.).

Peak detection parameters: *local-dB-range* = 75dB, *general-dB-range* = 85dB, *minimum-peak-height* = 2dB, *frequency-range* = 80Hz–12KHz.

Peak continuation parameters: *maximum-peak-deviation* = 50Hz, *peak-contribution-to-guide* = .3, *maximum-number-of-guides* = 50, *minimum-starting-guide-separation* = 100Hz, *maximum-sleeping-time* = 0 frames, *length-of-filled-gaps* = 0 frames, *minimum-trajectory-length* = 5 frames (.04 sec.).

1. original sound
2. deterministic synthesis
3. deterministic synthesis with original attack

D.3 Conclusions

This appendix has presented a use of the deterministic analysis/synthesis process which permits splicing the attack of an original sound onto its synthesized version. Since in most instrumental sounds the attack is the main non-linear portion, and thus hard to synthesize, and since it is very important perceptually, the use of a real attack improves the quality of the sound enormously. Some transformations are still possible in the synthesized component of the sound.

This process can also be used to splice attacks of one sound onto steady states of other sounds. Thus obtaining hybrid sounds. For example, it is possible to splice the attack of a trumpet onto the decay of a piano. This is done by matching the phases of the synthesized piano with the real trumpet sound at the splice point. The phases of the trumpet are detected by performing a Fourier transform at the splice point and then used as phase values in the deterministic analysis of the piano tone at that same point.

Appendix E

Sound Examples Index

This appendix includes an index of the sound examples that accompany this dissertation.

An audio tape with the examples can be ordered from: Center for Computer Research in Music and Acoustics, Department of Music, Stanford University, Stanford, CA. 94305.

E.1 STFT Examples

1. Excerpt from the Gloria of the Mass in C minor, K427, by Wolfgang Amadeus Mozart.

1. original sound
2. synthesis from the STFT analysis
3. synthesis with a time expansion by a factor of 2

E.2 Sinusoidal Model Examples

2. Excerpt from "El Amor Brujo" by Manuel de Falla.

1. original sound
2. synthesis with phase
3. synthesis without phase
4. synthesis with time expansion by factor of 1.68
5. synthesis with frequency transposition by factor of 1.4
6. synthesis with frequency transposition by factor of .8

3. Guitar passage.

1. original sound
2. synthesis with phase tracking
3. synthesis without phase tracking
4. synthesis with time expansion by a factor of 1.45

E.3 Sinusoidal plus Residual Model Examples

4. Guitar passage.

1. original sound
2. deterministic synthesis
3. residual
4. deterministic synthesis plus residual

5. Flute sound.

1. original sound
2. deterministic synthesis
3. residual
4. deterministic synthesis plus residual

6. Vocal sound.

1. original sound
2. deterministic synthesis
3. residual
4. deterministic synthesis plus residual

7. Piano passage.

1. original sound
2. deterministic synthesis
3. residual
4. deterministic synthesis plus residual

E.4 Sinusoidal plus Stochastic Model Examples

8. Guitar passage.

1. original sound
2. deterministic synthesis
3. stochastic synthesis
4. deterministic plus stochastic synthesis
5. frequency transposition by a factor of .3
6. frequency transposition by .7 and stretching of partials
7. compression of the frequency evolution
8. inversion of the frequency evolution
9. time-varying glissando and stretching of partials
10. time-varying time-scale
11. time expansion by 2.3
12. time expansion by 2.3 with time-varying time-scale and stretching of partials
13. time compression by .5 with time-varying time-scale and stretching of partials
14. time compression by .5 and frequency transposition by a factor of .4
15. time compression by .5 and glissando down

9. Speech phrase.

1. original sound
2. deterministic synthesis

3. stochastic synthesis
4. deterministic plus stochastic synthesis
5. frequency transposition by a factor of .6
6. compression of the frequency evolution and frequency transposition by a factor of .4
7. frequency transposition by .4 and stretching of partials
8. time-varying glissando and stretching of partials
9. time-varying time-scale and time-varying compression of the frequency evolution
10. from deterministic to stochastic signal
11. time compression by .3, compression of the frequency, and frequency transposition by a factor of .4
12. time compression by .3 and compression of the frequency
13. time expansion by 3
14. time expansion by 3 of only stochastic component and time-varying time-scale

10. Conga passage.

1. original sound
2. deterministic synthesis
3. stochastic synthesis
4. deterministic plus stochastic synthesis
5. compression of the frequency evolution
6. compression of the frequency evolution and frequency transposition by .3
7. compression of the frequency evolution and frequency transposition by 2
8. stretch partials
9. glissando down
10. glissando up
11. time-varying change of noise component
12. time-varying time-scale

13. time-varying time-scale (inverse of previous example)
14. time-varying time-scale and time-varying stretch partials
15. change of the frequency evolution
16. inverse of the previous example
17. time expansion by 3

11. Flute passage.

1. original sound
2. deterministic synthesis
3. stochastic synthesis
4. deterministic plus stochastic synthesis
5. compression of the frequency evolution
6. frequency transposition by .5 and stretch partials
7. compression of the frequency evolution, frequency transposition by .8, time-varying time-scale, and stretch partials
8. inversion of frequency evolution
9. time compression by .4
10. time compression by .4, frequency transposition by .8, and compression of the frequency evolution

12. Piano passage.

1. original sound
2. deterministic synthesis
3. stochastic synthesis
4. deterministic plus stochastic synthesis
5. frequency transposition by .5
6. compression of the frequency evolution
7. time-varying stretch partials

13. Singing-voice passage.

1. original sound
2. deterministic synthesis
3. stochastic synthesis
4. deterministic plus stochastic synthesis
5. frequency transposition by .8 and stretch partials
6. frequency transposition by .8 and compress partials
7. glissando and expansion of the frequency evolution
8. time-varying time-scaling
9. more percentage of stochastic component and increasing with time

E.5 Cross-synthesis Examples

14. Speech phrase hybridized with other sounds.

1. speech sound
2. cat sound
3. cross-synthesis of cat with speech
4. cow sound
5. cross-synthesis of cow and speech
6. gong sound
7. cross-synthesis of gong and speech
8. plane sound
9. cross-synthesis of plane and speech
10. ship creaking sound
11. cross-synthesis of ship creaking and speech
12. modification of the spectral envelopes on the previous example
13. another modification of the spectral envelopes

E.6 Deterministic Synthesis with Original Attack Examples

15. Marimba note.

1. original sound
2. deterministic synthesis
3. deterministic synthesis with original attack

16. Piano note.

1. original sound
2. deterministic synthesis
3. deterministic synthesis with original attack

Bibliography

Adrien, Jean Marie, René Causse and Xavier Rodet. 1987. "Sound synthesis by physical models, application to strings," *Proc. of the International Computer Music Conference,* 1987, Urbana, Illinois, pp. 264–269.

Adrien, Jean Marie, René Causse and Eric Ducasse. 1988. "Dynamic modeling of stringed and wind instruments, sound synthesis by physical models," *Proc. of the International Computer Music Conference,* 1988, Cologne, Germany, pp. 265–276.

Allen, Jont B. 1977. "Short term spectral analysis, synthesis, and modification by discrete fourier transform," *IEEE Trans. on Acoust., Speech, and Signal Proc.*, vol. ASSP–25, pp. 235–238.

Allen, Jont B. and Lawrence R. Rabiner. 1977. "A Unified approach to short-time fourier analysis and synthesis," *Proc. IEEE*, vol. 65, pp. 1558–1564.

Almeida, Luis B. and Fernando M. Silva. 1984. "Variable-frequency synthesis: An improved harmonic coding scheme," *1984 Proc. IEEE Int. Conf. Acoust. Speech and Sig. Proc.*, San Diego, CA, pp. 27.5.1–27.5.4.

Amuedo, John. 1985. "Periodicity estimation by hypothesis-directed search," *1985 Proc. IEEE Int. Conf. Acoust. Speech and Sig. Proc.* Tampa, Florida, pp. 11.6.1–11.6.4.

Atal, B. S. and M. R. Schroeder. 1967. "Predictive coding of speech signals," Proc. 1967 Conf. Commun. and Process., pp. 360–361.

Atal, B. S. 1970. "Speech analysis and synthesis by linear prediction of the speech wave," *J. Acoust. Soc. Amer.*, vol. 47, no. 1, pp. 65.

Atal, Bishnu S. and Suzanne L. Hanauer. 1971. "Speech analysis and synthesis by linear prediction of the speech wave," *J. Acoust. Soc. Amer.*, vol. 50, no. 2, pp. 637–655.

Atal, Bishnu S. and Joel R. Remde. 1982. "A new model of LPC excitation for producing natural-sounding speech at low bit rates," *1982 Proc. IEEE Int. Conf. Acoust. Speech and Sig. Proc.*, pp. 614–617.

Beauchamp, James W. 1969. "A computer system for time-variant harmonic analysis and synthesis of musical tones," in Heinz von Foerster and James W. Beauchamp, eds. *Music by computers,* New York: Wiley, pp. 19–62.

Bennett, Gerald and Xavier Rodet. 1989. "Synthesis of the singing voice," *Current directions in computer music research,* Max Mathews and John Pierce ed., Cambridge, Massachusetts: The MIT Press, pp. 19–33.

Berio, Luciano. 1958. "Thema (Omaggio a Joyce)," Turnabout 34177. (disk)

Bracewell, Ronald N. 1978. *The Fourier transform and its applications,* New York: McGraw-Hill.

Cage, John. 1952. "Williams Mix," Avakian JC-1. (disk)

Cann, Richard. 1979-1980. "An analysis/synthesis tutorial," (parts 1, 2, 3). *Computer Music Journal,* vol. 3, no. 3, pp. 6–11; vol. 3, no. 4, pp. 9–13; vol. 4, no.1, pp. 36–42.

Chafe, Chris. 1983. "Solera," *Computer Music from CCRMA,* vol. 1, CCRMA, Stanford University. (cassette)

Chafe, Chris. 1989. "Pulsed noise in self-sustained oscillations of musical instruments," submitted to ICASSP-90.

Cox, M. G. 1971. "An algorithm for approximating convex functions by means of first-degree splines," *Computer Journal,* vol. 14, pp. 272–275.

Crochiere, R. E. 1980. "A weighted overlap-add method of fourier analysis-synthesis," *IEEE Trans. on Acoust., Speech, and Signal Proc.*, vol. ASSP–28, pp. 55–69.

Cyphers, David S. 1985. "Spire: A speech research tool," Master's Thesis, Elec. Eng. Dept., Massachusetts Institute of Technology.

Davies, Hugh. 1968. *International electronic music catalog,* Cambridge, Massachussets: The M.I.T. Press.

Dodge, Charles. 1975. "Synthetic speech music," Composer's Recordings, Inc., New York, CRI-SD-348. (disk)

Dodge, Charles. 1983. "Cascando," Composer's Recordings, Inc., New York, CRI-SD-454. (disk)

Dodge, Charles. 1985. "In Celebration: The composition and its realization in synthetic speech," *Composers and the Computer,* Curtis Roads ed., Los Altos, California: William Kaufmann, Inc. pp. 47–74.

Dodge, Charles and Thomas A. Jerse. 1985. *Computer music,* New York: Schirmer Books.

Dodge, Charles. 1989. "On Speech Songs," *Current directions in computer music research,* Max Mathews and John Pierce ed., Cambridge, Massachusetts: The MIT Press, pp. 9–17.

Dolson, Mark B. 1983a. "A tracking phase vocoder and its use in the analysis of ensemble sounds," Ph.D. Dissertation, California Institute of Technology.

Dolson, Mark B. 1983b. "Musical applications of the phase vocoder," *Proc. of the International Computer Music Conference,* 1983, Rochester, New York, pp. 99–102.

Dolson, Mark B. 1984. "Refinements in the phase-vocoder-based modification in music," *Proc. of the International Computer Music Conference,* 1984, Paris, France, pp. 65–66.

Dolson, Mark B. 1985. "Recent advances in musique concrete at CARL," *Proc. of the International Computer Music Conference,* 1985, Burnaby, B.C., Canada, pp. 55–60.

Dolson, Mark B. 1986. "The phase vocoder: A tutorial," *Computer Music J.*, vol. 10, no. 4, pp. 14–27.

Dolson, Mark B. 1989. "Fourier-transform-based timbral manipulations," *Current directions in computer music research,* Max Mathews and John Pierce ed., Cambridge, Massachusetts: The MIT Press, pp. 105–112.

Dudley, Homer. 1939. "The vocoder," Bell Labs Rececord, vol. 18, pp. 122–126.

Ernst, David. 1977. *The evolution of electronic music,* New York: Schirmer Books.

Flanagan, J. L., and R. M. Golden. 1966. "Phase vocoder," Bell System Technical Journal, vol. 45, pp. 1493–1509.

Fletcher, Harvey and W. A. Munson. 1933. "Loudness, its definition, measurement and calculation," *J. Acoust. Soc. Amer.*, vol. 5, pp. 82–108.

Freedman, M. D. 1965. *A technique for analysis of musical instrument tones,* Ph.D. Dissertation, University of Illinois.

Freedman, M. D. 1967. "Analysis of musical instrument tones," *J. Acoust. Soc. Amer.*, vol. 41, pp. 793–806.

Freedman, M. D. 1968. "A method for analyzing musical tones," *J. Audio Eng. Soc.*, vol. 16, no. 4, pp. 419–425.

Gordon, John W. and John Strawn. 1985. "An introduction to the phase vocoder," *Digital audio signal processing: An anthology,* J. Strawn, ed., Los Altos, CA: William Kaufmann, Inc.

Grey, John M. 1975. *An exploration of musical timbre,* Ph.D. Dissertation, Stanford University.

Grey, John M. and James A. Moorer. 1977. "Perceptual evaluations of synthesized musical instrument tones," *J. Acoust. Soc. Amer.*, vol. 62, no. 3, pp. 454–462.

Grey, John M. and John W. Gordon. 1978. "Perceptual effects of spectral modifications on musical timbres," *J. Acoust. Soc. Amer.*, vol. 63, no. 5, pp. 1493–1500.

Griffin, Daniel W. and Jae S. Lim. 1988. "Multiband excitation vocoder," *IEEE Trans. on Acoust., Speech, and Signal Proc.*, vol. ASSP–36, pp. 1223–1235.

Griffin, Daniel W. and Jae S. Lim. 1984. "Signal estimation from modified short-time fourier transform," *IEEE Trans. on Acoust., Speech, and Signal Proc.*, vol. ASSP–32, pp. 236–242.

Harris, Fredric J. 1978. "On the use of windows for harmonic analysis with the discrete fourier transform," *Proc. IEEE*, vol. 66, pp. 51–83.

Harvey, Jonathan. 1980. "Mortuos Plango, Vivos Voco," Digital Music Digital, Wergo, WER 2025-50. (CD)

Henry, Pierre. 1952. "Vocalise," Ducretet-Thomson-9. (disk)

Hess, Wolfgang. 1983. *Pitch determination of speech signals,* New York: Springer-Verlag.

Jaffe, David. 1987a. "Spectrum analysis tutorial, Part 1: The discrete Fourier transform," *Computer Music J.*, vol. 11, no. 2, pp. 9–24.

Jaffe, David. 1987b. "Spectrum analysis tutorial, Part 2: Properties and applications of the discrete Fourier transform," *Computer Music J.*, vol. 11, no. 3, pp. 17–35.

Janssen, Jos and Heinerich Kaegi. 1986. "MIDIM–Duplication of a central-javanese sound concept," Interface, Vol. 15, pp. 185–229.

Kaegi, Werner and S. Tempelaars. 1978. "VOSIM–A new sound synthesis system," *J. Audio Eng. Soc.*, vol. 26, no. 6, pp. 418–425.

Kaiser, J. F. 1974. "Nonrecursive digital filter design using the I_0-sinh window function," Proc. 1974 IEEE Int. Symp. on Circuits and Syst., pp. 20–23.

Kassel, Robert H. 1986. *A user's guide to SPIRE,* Speech Communication Group, Massachusetts Institute of Technology.

Kronland-Martinet, Richard. 1988. "The wavelet transform for analysis, synthesis, and processing of speech and music sounds," *Computer Music J.*, vol. 12, no. 4, pp. 11–20.

Lansky, Paul and Kenneth Steiglitz. 1981. "Synthesis of timbral families by warped linear prediction," *Computer Music J.*, vol. 5, no. 3, pp. 45–49.

Lansky, Paul. 1989. "Compositional applications of linear predictive coding," *Current directions in computer music research,* Max Mathews and John Pierce ed., Cambridge, Massachusetts: The MIT Press, pp. 5–8.

Luce, David A. 1963. *Physical correlates of nonpercussive musical instrument tones,* Ph.D. Dissertation, Department of Physics, Massachusetts Institute of Technology.

Maher, Robert C. 1989. *An approach for the separation of voices in composite musical signals,* Ph.D. Dissertation, University of Illinois at Urbana-Champaign.

Makhoul, John. 1975. "Linear prediction: A tutorial review," *Proc. IEEE,* vol. 63, pp. 561–580.

Manning, Peter. 1985. *Electronic and computer music,* London: Oxford University Press.

Markel, J. D. and A. H. Gray. 1976. *Linear prediction of speech,* New York: Springer-Verlag.

McAulay, Robert J. and Thomas F. Quatieri. 1984. "Magnitude-only reconstruction using a sinusoidal speech model," *1984 Proc. IEEE Int. Conf. Acoust. Speech and Sig. Proc.* San Diego, California, pp. 27.6.1–27.6.4.

McAulay, Robert J. and Thomas F. Quatieri. 1986. "Speech analysis/synthesis based on a sinusoidal representation," *IEEE Trans. on Acoust., Speech, and Signal Proc.*, vol. ASSP–34, pp. 744–754.

Moorer, James A. 1973. "The heterodyne filter as a tool for analysis of transient waveforms," Report No. STAN-CS-73-379, Computer Science Department, Stanford University.

Moorer, James A. 1975. *On the segmentation and analysis of continuous musical sound by digital computer,* Ph.D. Dissertation, Stanford University.

Moorer, James A. 1977. "Signal processing aspects of computer music: A survey," *Proc. IEEE*, vol. 65, pp. 1108–1137.

Moorer, James A. 1978. "The use of the phase vocoder in computer music applications," *J. Acoust. Soc. Amer.*, vol. 26, no. 3/2, pp. 42–45.

Moorer, James A. 1979. "The use of linear prediction of speech in computer music applications," *J. Acoust. Soc. Amer.*, vol. 27, no. 3, pp. 134–140.

Moorer, James A. 1983. "Lions are growing," *Computer Music from CCRMA,* vol. 1. CCRMA, Stanford University. (cassette)

Nawab, S. Hamid, Thomas F. Quatieri and Jae S. Lim. 1983. "Signal reconstruction from short-time fourier transform magnitude," *IEEE Trans. on Acoust., Speech, and Signal Proc.*, vol. ASSP–31, pp. 986–998.

Nuttall, Albert H. 1981. "Some windows with very good sidelobe behavior," *IEEE Trans. on Acoust., Speech, and Signal Proc.*, vol. ASSP–29, pp. 84–91.

Oppenheim, Alan V. and Ronald W. Schafer. 1975. *Digital signal processing,* Englewood Cliffs, New Jersey: Prentice-Hall.

Petersen, Tracy L. 1975. "Voices," Tulsa, Okla.: Tulsa Studios.

Petersen, Tracy L. 1976. "Vocal tract modulation of instrumental sounds by digital filtering," Proceedings of the Int. C.M.C., part I, 1975, Urbana, Illinois, pp. 33–41.

Phillips, G. M. 1968. "Algorithms for piecewise straight line approximation," *Computer Journal*, vol. 11, pp. 211–212.

Piszczalski, M. and B. A. Galler. 1979. "Predicting musical pitch from component frequency ratios," *J. Acoust. Soc. Amer.*, vol. 66, no. 3, pp. 710–720.

Plomp, R. 1966. *Experiments on tone perception,* Institute for perception RVO-TNO, Soesterberg, The Netherlands.

Portnoff, Michael. R. 1976. "Implementation of the digital phase vocoder using the fast fourier transform," *IEEE Trans. on Acoust., Speech, and Signal Proc.*, vol. ASSP–24, pp. 243–248.

Portnoff, Michael R. 1980. "Time-frequency representation of digital signals and systems based on short-time fourier analysis," *IEEE Trans. on Acoust., Speech, and Signal Proc.*, vol. ASSP–28, pp. 55–69.

Portnoff, Michael. R. 1981. "Time-scale modification of speech based on short-time fourier analysis," *IEEE Trans. on Acoust., Speech, and Signal Proc.*, vol. ASSP–29, pp. 374–390.

Quatieri, Thomas F. and Robert J. McAulay. 1986. "Speech transformations based on a sinusoidal representation," *IEEE Trans. on Acoust., Speech, and Signal Proc.*, vol. ASSP–34, pp. 1449–1464.

Rabiner, Lawrence R. and Bernard Gold. 1975. *Theory and applications of digital signal processing,* Englewood Cliffs, New Jersey: Prentice-Hall.

Rabiner, Lawrence R. and Ronald W. Schafer. 1978. *Digital processing of speech signals,* Englewood Cliffs, New Jersey: Prentice-Hall.

Reich, Steve. 1966. "Come Out," Odyssey 32160160. (disk)

Risset, Jean-Claude. 1988. "Songes," Digital Music Digital, Wergo, WER 2013-50. (CD)

Risset, Jean-Claude, and Max V. Mathews. 1969. "Analysis of musical-instrument tones," Physics Today, vol. 22, no. 2, pp. 23–30.

Roads, Curtis. 1983. "A report on SPIRE: An interactive audio processing environment," *Computer Music J.*, vol. 7, no. 2, pp. 70–74.

Rodet, Xavier. 1984. "Time-domain formant-wave-function synthesis," *Computer Music J.*, vol. 8, no. 3, pp. 9–14.

Rodet, Xavier, Y. Potard, and J. B. Barrière. 1984. "The CHANT project: From synthesis of the singing voice to synthesis in general," *Computer Music J.*, vol. 8, no. 3, pp. 15–31.

Roederer, Juan G. 1979. *Introduction to the physics and psychophysics of music,* New York: Springer-Verlag.

Rosenfeld, Azriel. 1969. *Picture processing by computer,* New York: Academic Press.

Saito, S. and F. Itakura. 1966. "The theoretical consideration of statistically optimum methods for speech spectral density," Report No. 3107, Electrical Communication Laboratory, N.T.T., Tokyo. (In Japanese).

Schaeffer, Pierre. 1966. *Traité des objets musicaux,* Paris: Éditions de Seuil.

Schaeffer, Pierre and Pierre Henry. 1949. "Symphonie pour un homme seul," Philips 6510-012. (disk)

Schafer, Ronald W. and Lawrence R. Rabiner. 1975. "Digital representations of speech signals," *Proc. IEEE*, vol. 63, pp. 662–677.

Scharf, Bertrand. 1970. "Critical bands," in Tobias, *Foundations of modern auditory theory,* vol. I, New York: Academic Press.

Schroeder, Manfred R. 1966. "Vocoders: Analysis and synthesis of speech," *Proc. IEEE*, vol. 54, pp. 720–734.

Schroeder, Manfred R. 1968. "Period histogram and product spectrum: new methods for fundamental frequency measurement," *J. Acoust. Soc. Amer.*, vol. 43, no. 4, pp. 829–834.

Schumacher, Robert T., and Chris Chafe. 1989. "Detection of aperiodicity in nearly periodic signals," submitted to ICASSP-90.

Sedgewick, Robert. 1988. *Algorithms,* Reading, Massachusetts: Addison-Wesley.

Serra, Marie-Hélène, Dean Rubine and Roger B. Dannenberg. 1988. "The analysis and resynthesis of tones via spectral interpolation," *Proc. of the International Computer Music Conference,* 1988, Cologne, Germany, pp. 322–332.

Serra, Xavier. 1986. "A computer model for bar percussion instruments," *Proc. of the International Computer Music Conference,* 1986, The Hague, Netherlands, pp. 257–262.

Shipman, David W. 1982. "Development of speech research software on the MIT Lisp machine," The Journal of the Acoustic Society of America, 103rd Meeting, Chicago, Illinois, 26-30 April 1982.

Shipman, David W. 1982. "The use of the FPS-100 from the MIT Lisp machine," Research Laboratory of Electronics, Massachusetts Institute of Technology.

Smith, Julius O. 1983. "Methods for digital filter design and system identification with application to the violin," Ph.D. Dissertation, Elec. Eng. Dept., Stanford University.

Smith, Julius O. 1985. "Introduction to digital filter theory," *Digital audio signal processing: An anthology,* J. Strawn, ed., Los Altos, California: William Kaufmann, Inc.

Smith, Julius O. 1987. "Musical applications of digital waveguides," Report No. STAN-M-39, Music Department, Stanford University.

Smith, Julius and Xavier Serra. 1987. "PARSHL: An analysis/synthesis program for non-harmonic sounds based on a sinusoidal representation," *Proc. of the International Computer Music Conference,* 1987, Urbana, Illinois, pp. 290–297.

Smith, M. J. T. and T. P. Barnwell. 1985. "A unifying framework for analysis/synthesis systems based on maximally decimated filter banks," *85 Proc. IEEE Int. Conf. Acoust. Speech and Sig. Proc.*, Tampa, Florida, pp. 521–524.

Stockhausen, Karlheinz. 1956. "Gesang der Jünglinge," Deutsche Grammophon Gesellschaft 138811 SLPM. (disk)

Strawn, John. 1980. "Approximation and syntactic analysis of amplitude and frequency functions for digital sound synthesis," *Computer Music J.*, vol. 4, no. 3, pp. 3–24.

Terhardt, Ernst. 1979. "On the perception of spectral information in speech," in O. Creutzfeld et alt. ed. *Hearing Mechanisms and Speech,* 1979, New York: Springer-Verlag.

Terhardt, Ernst, Gerhard Stoll, and Manfred Seewann. 1982a. "Pitch of complex signals according to virtual-pitch theory: Tests, examples, and predictions," *J. Acoust. Soc. Amer.*, vol. 71, no. 3, pp. 671–678.

Terhardt, Ernst, Gerhard Stoll, and Manfred Seewann. 1982b. "Algorithm for extraction of pitch and pitch salience from complex tonal signals," *J. Acoust. Soc. Amer.*, vol. 71, no. 3, pp. 679–688.

Tou, Julius T., and Rafael C. Gonzales. 1974. *Pattern recognition principles,* Reading, Massachusetts: Addison-Wesley.

Tribolet, Jose M. and Ronald E. Crochiere. 1979. "Frequency domain coding of speech," *IEEE Trans. on Acoust., Speech, and Signal Proc.*, vol. ASSP–27, pp. 512–530.

Van Trees, Harry L. 1968. *Detection, estimation, and modulation theory,* New York: John Wiley & Sons.

Varèse, Edgard. 1958. "Poème Electronique," Columbia MS-6146. (disk)

Wahl, Friedrich M. 1987. *Digital image signal processing,* Boston: Artech House.

Weinreb, Daniel and David Moon. 1981. *Lisp machine manual,* Symbolics, Inc.

Wolcin, Joseph J. 1980. "Maximum *a posteriori* estimation of narrow-band signal parameters," *J. Acoust. Soc. Amer.*, vol. 68, no. 1, pp. 174–178.

Wolman, Amnon. 1989. Composition in progress.

Xenakis, Iannis. 1960. "Orient-Occident I," revised and published in 1968 as "Orient-Occident III," Nonesuch 71246. (disk)

Yost, William A. and Donald W. Nielsen. 1977. *Fundamentals of hearing,* New York: Holt, Rinehart and Winston.

Zwicker, E. and B. Scharf. 1965. "A model for loudness summation," *Psych. Rev.*, vol. 72, pp. 3–26.

Zwicker, E., G. Flottorp and S. S. Stevens. 1957. "Critical band width in loudness summation," *J. Acoust. Soc. Amer.*, vol. 29, pp. 548–557.