Issues on Modeling the Singing Voice

by

Àlex Loscos Mira

Submitted in partial fulfillment of the requirements for
the degree of Diploma of Advanced Studies
Doctorate in Computer Science and Digital Communication
Department of Technology

Tutor: Dr. Xavier Serra

Universitat Pompeu Fabra

Barcelona, September 2003

Abstract

This work presents and puts into context the research contained in all publications in which the author has been involved so far. This set of gathered publications are mainly focused on the field of singing voice processing; more precisely, on spectral processing techniques and voice modeling for singing voice analysis, transformation and synthesis. The final goal of this work is to set the future directions of the research on top of which the final thesis work will stand.

Acknowledgments

Although this is one of the shortest sections of this work, it is one of the most important, especially when the work presented is the result of team collaboration.

I would like to thank all the people from the Music Technology Group, especially to Jordi, Pedro, Perfe, Ramon, Maarten, Xavier, Oscar, Jaume, Mark, Zeta, Lars, Emilia, Fabien, Enric, Joana, Cristina, and Professor Xavier Serra. Thanks as well to all the people from Yamaha Corp. I have worked with.

Contents

Chapter 1

Introduction

1.1 The work

The present document gathers all the publications in which I have been involved so far since 1998. This collection includes a set of conference papers, a journal article, and a book chapter.

In order to define the framework and the context in which the publications where published, the current work opens with a personal introduction of my trajectory as a researcher and goes on with two main chapters: Spectral Processing Techniques, and Voice Modeling and Synthesis. The aim of such chapters is to introduce the knowledge fields where the research has been carried out and explain the techniques that have sustained it.

The Spectral Processing Techniques chapter introduces the already mentioned SMS and SPP techniques, and describes their corresponding analysis, transformation and synthesis methods. The Voice Modeling and Synthesis chapter introduces the voice acoustics theory, presents the peculiarities of the singing voice in comparison to normal speech, and introduces the EpR voice model and our singing voice synthesizer system.

Next, all the publications are presented with its title, its abstract, and a description of my contribution to it. A conclusions chapter closes the work and presents which might be the basis and the directions to point out for the thesis work. Finally, the publications and patents that have resulted from my collaboration with Yamaha are annexed.

1.2 The place

The Music Technology Group (MTG) is a research group placed in Barcelona dedicated to sound processing and synthesis, audio identification, audio content analysis, classification and transformation, interactive systems, multimedia applications and other subjects related with audio technology research and experimentation. Inside the MTG, a reduced group of people conform the Singing Voice Processing Group. This group is in charge of the voice related projects and research.

The MTG belongs to the Institut Universitari Audiovisual (IUA) of the Universitat Pompeu Fabra (UPF). The MTG was founded in 1994 by its current director Xavier Serra and actually has more than 30 researches working in it. The research is funded by private companies such as Yamaha (the world's largest manufacturer of musical instruments and a leader in digital audio) or SDAE (the Spanish digital authoring society), and by public and governmental institutions (Generalitat de Catalunya, Ministerio Español de Ciencia y Teconlogía and European Comissioin). On 2001, the MTG became a member of the Technology Innovation Network of the Generalitat de Catalunya. The Xarxa IT is a network of the research groups of the Catalan universities that was created to help them get closer to the industrial sector and to create technology transfer opportunities.

1.3 The author

Here is a brief summary of my academic trajectory.

1992-1996
I study Telecommunications engineering in the Escola d' Enginyeria Tècnica Superior de Telecomunicacions de Barcelona (ETSETB), Universitat Politecnica de Catalunya (UPC)

1997-1999
While already working with my graduate thesis inside the UPC's voice processing group, I get a scholarship to join the Musical Technology Group (MTG) of the Institut Universitari Audiovisual (IUA) of the Universitat Pompeu Fabra (UPF).

1999
Distinction in the graduate thesis *"Singing Voice Morphing System based on SMS"* presented in the UPC.
I graduate as a Telecommunication Engineer specialized in Signal Processing at the ETSETB, UPC.

1999-2000
Researcher in the Music Technology Group of the IUA

1999-2000
Programming and Signal Processing teacher in the Informatics Engineering courses of the Enginyeria Superior en Informàtica (ESI) of the UPF.

2000
Synthesis and Audio Processing teacher in the Digital Arts Master courses of the UPF.

2000-2001
Ph.D. student in the Doctorate in Computer Science and Digital Communication at the UPF

2001-2002
Audio Effects teacher in the Digital Audio Master courses of La Salle, Universitat Ramon LLull, Barcelona.

However, my academic curriculum is not the only responsible of ending up in the Musical Technology Group doing research on singing voice synthesis. My relationship with music has some relevance here. I have been a music passionate enthusiast since kid and though I have not learned music in an academic way I started my amateur career as a musician and song-writer about ten years ago. All these years round I have been learning to play guitar, electric bass and keyboards and I have started singing as well. Unlike the rest of the instruments, the exercise of learning to sing brought me about a wide curiosity for all what was underneath singing and a wide interest for all it could be placed on top of it.

1.4 The projects

Here is a brief presentation of the projects I have been involved so far and in which I have carried out all the research presented in this work. The projects are presented in chronological order.

1.4.1 Elvis

Elvis was a collaboration project with Yamaha. The aim of the project was to develop an automatic singing voice impersonator for the Japanese karaoke customers. The system had to be able to change some voice characteristics in real-time in order to make the karaoke user's singing resemble a famous pop-rock singer (as for example Elvis) while performing. To do so, a frame-based real-time singing voice analysis-transformation-synthesis engine was implemented.

The analysis-transformation-synthesis engine was built on top of the Spectral Modeling Synthesis (SMS), an analysis by synthesis technique based on the sinusoidal plus residual decomposition of the sound. The analysis is in charge of parameterizing the voice. The transformation step is in charge of modifying the parameters obtained in the analysis. The synthesis step is in charge of generating the synthetic voice out of the modified parameterization.

The transformation step is the critical step. There the user's voice is morphed with the famous singer's by interpolating their corresponding voice parameters. Morph is a technique by which out of two or more elements you can generate others with hybrid properties. In this context, we use morphing techniques to change the personality of the singing voice. Moreover, in order to have an appropriate morph, the performances of both the user and the famous singer have to be synchronized somehow so that the system will not morph a long sustained user's vowel with a short nasal famous singer's consonant. For this reason Automatic Speech Recognition (ASR) techniques were included in the system adapting them to the case of the singing voice, real-time, and since we know the lyrics that have to be sung, to the case of matching.

Following a chronological order of the tasks that are involved in an automatic real-time singing-voice impersonation, the first thing is to record in a dry environment (that is without reverb and without any music accompaniment) a group of professional Japanese singers singing some of the most successful Japanese songs. All this audio is used to train our ASR system and build the 47 phonetic units models that make up the Japanese phonetic dictionary. The last task to do before the user takes the microphone in the karaoke place is analyze, parameterize, segmentate in phonemes, and store the professional singer performance of the song the user will sing.

The real-time processes start digitalizing the singing voice signal the system receives through the sound card of the computer. The digitalized signal is analyzed and parameterized with the SMS spectral based techniques. The analysis gives out two different types of parameters. One group (Mel coefficients, delta Mel coefficients , zero crossing, …) will feed the ASR system in order to perform the matching between user's and professional singer's performances, so the system knows which part of the song the user is singing, and more precisely, which phoneme is being uttered. Once located this phoneme, the other group (energy, fundamental frequency, spectral shape envelope) will be used to generate the synthetic hybrid voice by interpolating this attributes of the user's voice with the professional singer's, which were previously obtained and stored. The process concludes when the synthetic voice is sent to the sound card. All these real-time processes occur in less than 30 milliseconds.

The prototype was implemented as a graphical interface using C++ on Linux. At that time Linux was improving Windows audio I/O latency in more than 40 milliseconds. The graphical interface, which emulates the classic karaoke interfaces, shows a score of the user's performance with an Elvis animation, and allows the user control the morphing values of the synthesis in real-time with a set of sliders.

The main research done for Elvis project has been published in [LCB99, CLB99, CLBBS00, BBCLS00].

1.4.2 SMSPerformer

SMSPerformer was an MTG inside project. The aim of the project was to come out with a graphical interface for the real-time SMS synthesis engine that could work from already SMS analyzed sounds. Althought its use was originally focused to test in real-time all synthesis and morph parameters in the voice impersonator context (with the purpose of determining the most suitable parameter values), the sound manipulation possibilities of the SMSPerformer and the interest shown by some composers in using it leaded to an extension to its original purposes and turned the test-plattform into a composition tool.

Since the program was going to be used as a composition and performance tool, it was decided it should include programmable time-varying transformations, MIDI control for the synthesis parameters, and performance loading and saving options. All these capabilities brought new possibilities to the SMS synthesis with real-time transformations and an improved graphical interface implemented with Visual C++ on Windows.

Most of the research done in this project has been published in [LR98]. Moreover, the SMSPerformer was used by the argentinian composer Ricardo Ventura to perform his electro acoustic piece "Desnudo de mujer en sofá azul". The work was presented in a concert organized by *Sala HAL* in which the composer transformed a set of sounds in real-time over a pre-recorded composition.

1.4.3 Daisy

Daisy was a collaboration project with Yamaha. The aim of the project was to develop a singing voice synthesizer, that is, a synthesizer in which the user would input the lyrics and the notes of a vocal melody and obtain a synthetic performance of a virtual singer. To synthesize such performance the system concatenates a chain of elemental synthesis units. These units are obtained by transposing and time-scaling samples from singers databases. These databases are created out of recording, analyzing, labeling and storing as many different musical and phonetic contexts as possible of the singer's performances.

The concatenation of the samples is performed by spreading out along a certain number of boundary frames the spectral shape and phase discontinuities. The expression of the singing is then applied using the Excitation plus Residual (EpR) singing voice model, built up on top of the Spectral Peak Processing (SPP).

The EpR is a voice model based on an extension of the source / filter approach in which the differentiated glottal pulse response and the vocal tract response are modeled in frequency with a set of resonances, an exponential function, and a differential envelope. The SPP is a spectral modeling technique that considers the spectrum to be composed of spectral regions, each of which contains a harmonic spectral peak and its surroundings. Both SPP and EpR will be described in details along the following chapters of this work.

For the prototype version, the singer database creation and the synthesis itself were implemented on top of the SMSTools2 graphical interface. The SMSTools2, despite of the name and despite it was originally conceived as the SMS software interface, has been used by the Singing Voice Processing group as the development platform for most of their try-outs and improvements.

Out of this collaboration project Yamaha has released a product named Vocaloid. Vocaloid was presented by Yamaha at the Musikmesse in Frankfurt, and at the 114th Audio Engineering Society (AES) Convention in Amsterdam. Their advertisement claims: "*by just inputting the melody and words on their PCs, users can produce the vocal parts for their pieces with no further work overcoming a major hurdle composers have faced until now due to the limitations that technology has placed on their ability to freely create songs incorporating singing. The software runs on Windows-based PCs and synthesizes the sound from "vocal libraries" of recordings of actual singers, retaining the vocal qualities of the original singing voices to reproduce real-sounding vocals. The software also features simple commands that allow users to add expressive effects. Currently, Vocaloid can generate singing in Japanese and English*."

The research carried out for Daisy is mainly described in [BL03, BLMK03, BCLOS01, BLCS01].

1.4.4 Telefónica

Telefónica was a collaboration project with the so called company, leader in telecommunication services in Spain and Latin America. The aim of the project was to evaluate the speech voice quality obtained from MTG's synthesis techniques based on SPP and study the feasibility of porting them to Telefonica's automatic text-to-speech machines. This evaluation was based on comparing the quality of the synthetic voices obtained from both Telefonica's and MTG's synthesis engines.

Since the MTG had no previous experience on speech synthesis and since the only element to be evaluated was the synthesis technique, we defined an interface through which Telefonica transferred to the MTG all the information required to synthesize a sentence. These were the speaker's database (segmented and labeled into phonetic units), the list of the phonetic units that conformed the utterance, and the prosody to be applied (syllable durations and intonation envelope).

The database conversion and the synthesis process were implemented on top of the SMSTools2, using the authoring system that it had been developed for Daisy's singers database creation. The evaluation was carried out through a web test questionnaire based on ITU-T P.85 standard that used *php* and *MySQL* as programming and management tools.

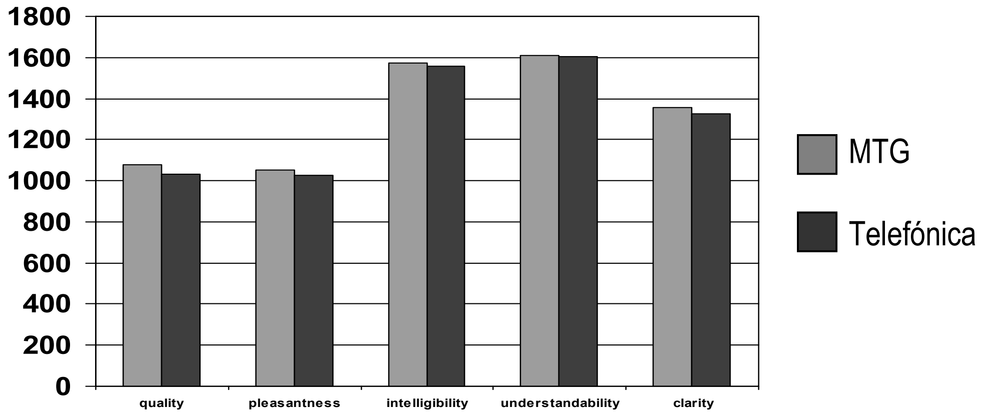


Figure 1.1 Global scoring of different synthesis aspects (quality, pleasantness, intelligibility, understandability, clarity)

Figure 1.1 shows the total scoring (obtained by summing the scores given to each sentence of the questionnaire) for both synthesis techniques and reveal that MTG's synthesis and Telefonica's highest quality synthesis were evaluated to be alike.

1.4.5 Object Processing

Object Processing was an MTG inside project. The aim of this project was to process a single audio object of a mix independently of the rest of the sound. With this we could modify the sound of a violin in an ensemble recording or we could apply transformations to the lead voice in a whole band recording.

The main steps to process a single audio object are: analyze the mixed audio using SPP analysis techniques, assign which peaks belong to each audio object, and process all the peaks that belong to the audio object to be modified. The project did not focus on how to discriminate which peaks belong to which instrument but on how to modify the characteristics of a certain instrument by modifying its corresponding peaks.

The implementation of Object Processing was done on top of the SMSTools2, using Visual C++.

1.4.6 Vocal Processor

Vocal Processor is a collaboration project with YAMAHA. The aim of the project is the implementation of a real-time singing voice effect processor. This effect processor is not a generic-like processor with which you can apply distortion, delay or flanger to a singing voice but a specific processor able to modify the singing voice characteristics in terms of singing characteristic attributes such as timbre, vibrato, intonation, tuning, breathiness, roughness, harmonies, and others.

The Vocal Processor uses the SPP technique and it is being implemented as a Virtual Studio Technology (VST) plug-in using the MTG's CLAM (C++ Libraries for Audio and Music) libraries.

Part of the research done for Vocal Processor is published in [ABLS01, ABLS02], and [ABLAV03].

Chapter 2

Spectral Processing Techniques

The aim of this chapter is to introduce the two main techniques on top of which the research presented in this work has been carried out: the Spectral Modeling Synthesis (SMS), and the Spectral Peak Processing (SPP).

2.1 Introducing the techniques

In this first section the background of such techniques (SMS and SPP) is introduced.

2.1.1 Phase Vocoder

The phase-vocoder [FG66] is a frequency-domain technique based on the Short Time Fourier Transform (STFT). The STFT characterizes the spectral behavior of a signal *x(n)* along time and can be written as:

$$X(n,k)=\sum_{m=-\infty}^{m=\infty} x(m)\cdot h(n-m)\cdot e^{-j\frac{2\pi mk}{N}}, \qquad k=0,1,\ldots N-1 \qquad (2.1)$$

where *X(n, k)* is the complex time-varying spectrum with the frequency bin *k* and time index *n*. At each time index, the input signal *x(m)* is weighted by a finite length window *h(n-m)* and then computed its spectrum. There is a more detailed explanation of the STFT in section 2.2.1.

The most common understanding of the phase-vocoder technique is the filter bank interpretation. This filter bank is a parallel bank of *N* bandpass filters with the following impulse response

$$h_k(n)=h(n)\cdot e^{j\frac{2\pi nk}{N}}, \qquad k=0,1,\ldots N-1 \qquad (2.2)$$

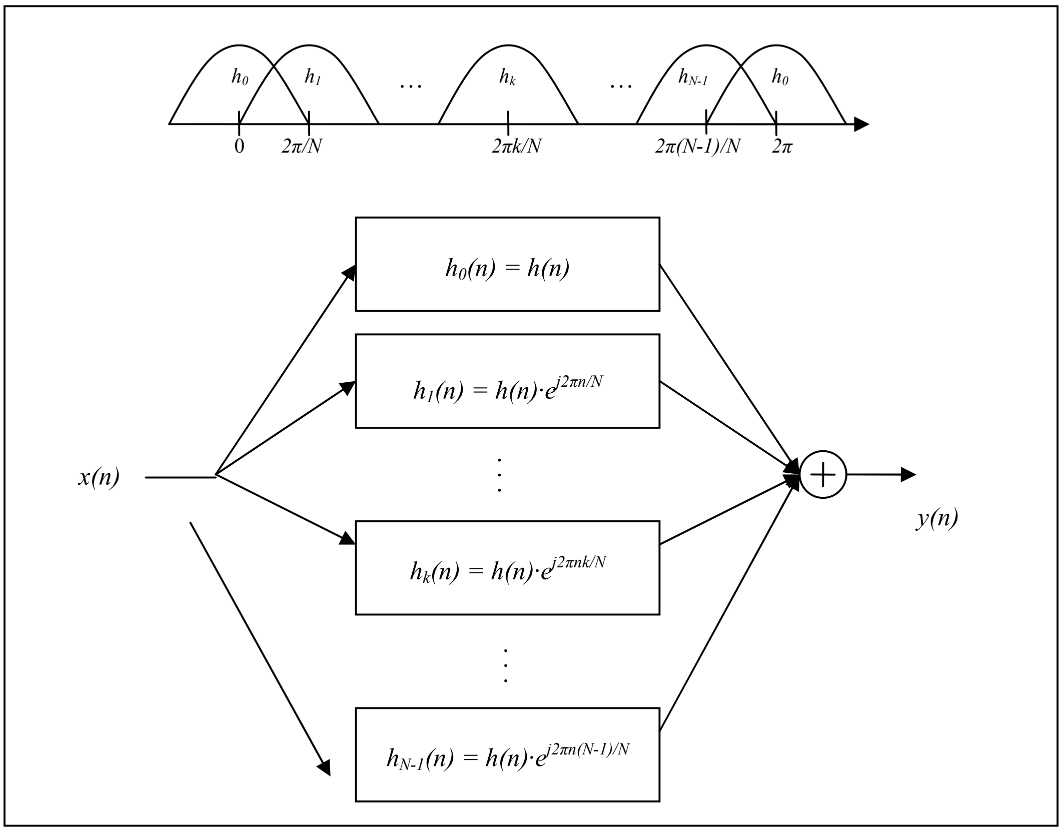
which means the same filter shape is shifted along frequency by $2\pi/N$ radians steps.

The bandpass signal of band *k*, $y_k(n)$, is obtained by filtering the input signal *x(n)* with filter $h_k(n)$ as showed in the following expression

$$\begin{aligned}
y_k(n) &= x(m) * h_k(n) = \sum_{m=-\infty}^{m=\infty} x(m)\cdot h_k(n-m) \\
&= \sum_{m=-\infty}^{m=\infty} x(m)\cdot h(n-m)\cdot e^{j\frac{2\pi(n-m)k}{N}} = \\
&= e^{j\frac{2\pi nk}{N}} \cdot \sum_{m=-\infty}^{m=\infty} x(m)\cdot e^{-j\frac{2\pi mk}{N}} \cdot h(n-m) = \\
&= e^{j\frac{2\pi nk}{N}} \cdot X(n,k)
\end{aligned} \tag{2.3}$$

and the output signal *y(n)* is the sum of all bandpass signals

$$y(n) = \sum_{k=0}^{k=N-1} x(n) * h_k(n) = \sum_{k=0}^{k=N-1} X(n,k)\cdot e^{j\frac{2\pi nk}{N}} \tag{2.4}$$



h_0
h_1
…
h_k
…
h_{N-1}
h_0
0
$2\pi/N$
$2\pi k/N$
$2\pi(N-1)/N$
2π
$h_0(n) = h(n)$
$h_1(n) = h(n)\cdot e^{j2\pi n/N}$
$h_k(n) = h(n)\cdot e^{j2\pi nk/N}$
$h_{N-1}(n) = h(n)\cdot e^{j2\pi n(N-1)/N}$
$x(n)$
$y(n)$

Figure 2.1 Filter bank description of the short time Fourier transform. The frequency bands on the top show the displacement of each of the bandpass filters.

Thus, for a certain time index *n*, *y(n)* will be defined by a *N* length vector containing an estimation of each band's energy

$$y_k(n) = e^{j\frac{2\pi nk}{N}} \cdot X(n,k) = |X(n,k)| \cdot e^{j\varphi_X(n,k)} \cdot e^{j\frac{2\pi nk}{N}} \tag{2.5}$$

so

$$\begin{aligned} |y_k(n)| &= |X(n,k)| \\ \varphi_{y_k}(n) &= \varphi_X(n,k) + \frac{2\pi \cdot n \cdot k}{N} \end{aligned} \tag{2.6}$$

Before synthesis, spectral modifications can be applied to the resulting analysis data. If we term the modified data $y_k^{\mathrm{mod}}(n)$, the synthesis window *w(n)*, the synthesis hop size *H*, and *$y_m(n)$* is the Inverse Fourier Transform of the modified short time spectra

$$y_m(n) = \frac{1}{N} \sum_{k=0}^{k=N-1} y_k^{\mathrm{mod}}(n) \cdot e^{j\frac{2\pi nk}{N}} = \frac{1}{N} \sum_{k=0}^{k=N-1} \left(e^{j\frac{2\pi nH}{N}} \cdot X(mH,k) \right)^{\mathrm{mod}} \cdot e^{j\frac{2\pi nk}{N}} \tag{2.7}$$

the resulting synthesis signal *s(n)* can be written as the overlap and add of the windowed *$y_m(n)$*'s

$$s(n) = \sum_{m=-\infty}^{m=\infty} w(n - m \cdot H) \cdot y_m(n) \tag{2.8}$$

Improved phase vocoder techniques have appeared more recently [Puck95,LD99]. These new techniques try to solve the phase unwrapping problem. This problem is caused by the fact that the phase obtained for each bin *k* depends on a term that is multiple of *2π* and that is not the same for all bins, see equation (2.6). These phase differences are dispersed and cause synthesis artifacts when applying spectral transformations. Out of these new techniques named phase-locked vocoders, a rigid phase locking method proposed by Laroche and Dolson called the identity phase locking, has an important relevance in this work.

The identity phase locking vocoder it is based on spectrum peak detection and segmentation and on the assumption that spectrum peaks are sinusoids. In this technique a simple peak detection algorithm detects all spectra local maximums and considers them peaks. This series of peaks are then used to chop the spectrum into regions. Each of these regions contains one peak and sets the boundaries with the neighboring regions to the middle frequency between adjacent peaks or to the lowest amplitude between the two peaks. The proposal states that only the phases of the peaks are calculated while the phases of the rest of the bins are locked to their corresponding region peak.

The identity locking asserts that the synthesis phases around a spectral peak are related with that peak's phase in the same way they are in the analysis. So, being k_p the bin index of the dominant peak, φ_s the synthesis phases, and φ_a the analysis phases, the phases of the bins k under the region of influence of peak k_p will be

$$\varphi_s(k) = \varphi_s(k_p) + \varphi_a(k) - \varphi_a(k_p) \text{ where } k \in Region(k_p) \tag{2.9}$$

2.1.2 Signal Modeling

The term signal modeling refers to the task of describing a signal with respect to its inherent characteristics and underlying structure - a model of the signal fundamental's behavior. In this context, analysis is the process of fitting such model to a particular signal and synthesis is the process by which a signal is reconstructed using the model and the analysis data. Among the numerous approaches that have appeared in the recent years, two model types are the most commonly used nowadays in musical sound generation: *physical models* and *spectrum models*.

Physical models attempt to parameterize the process of the sound generation at its source, e.g. the excitation mechanism of a flute and the resulting self-sustained oscillation produced inside its bore. The purpose of the analysis is to derive a general physical description of the instrument in question. That physical description is used to construct a synthesis system that mimics the instrument behavior. In a guitar model, for instance, the model parameters derived form the analysis include the length, tension, and various wave propagation characteristics of the strings, the acoustic resonances of the guitar body, and the transfer properties of the string-body coupling. These physical parameters can be used to build a system that, when driven by a modeled excitation such as a string pluck, synthesizes a realistic guitar sound. In speech processing, the classical examples of physical modeling are the human vocal tract models, which are source-filter approaches. The source mimics the glottal excitation while the filter models the shape of the vocal tract, meaning that the source-filter approach is designed to mirror the actual underlying physical system from which the speech signal originated.

Spectrum models are not taking this physical sound generation into account but rather attempt to parameterize the sound itself, as it is perceived by the basilar membrane of the ear, discarding whatever information the ear seems to discard in the spectrum. Both SMS and SPP are based on spectrum models.

2.1.2.1 The Sinusoidal Model

Within the spectrum models there are the additive models, which are based on the basic idea that a complex sound can be constructed by a large number of simple sounds. These models try to represent the spectral characteristics of the sound as a weighted sum of basic components, so-called basis expansions.

$$x(n) = \sum_{j=1}^{J} \alpha_j g_j(n) \tag{2.10}$$

where $g_j(n)$ is the j^{th} basis function and α_j is its appropriate chosen weight. There is no restriction to their number J. A detailed discussion on the properties of these basis expansions can be found in [Good97].

When the additive synthesis is the summation of time varying sinusoidal components rooting in Fourier's theorem, which states that any periodic waveform can be modeled as a sum of sinusoids of various amplitudes and harmonic frequencies, we talk about sinusoidal modeling. The sinusoidal model and in general additive models have been under consideration in the field of computer music since its inception.

One of the widely applied models in speech coding and processing, or audio analysis-synthesis systems is the sinusoidal model as proposed by McAulay and Quatieri in [MQ86]. At this, the signal is modeled as a sum of sinusoids with time-varying amplitudes $A_p(t)$, frequencies $\omega_p(t)$, and phases $\theta_p(t)$. Therefore we estimate the signal as

$$\begin{aligned} \hat{x}(t) &= \sum_{p=1}^{P} A_p(t) \sin\left[\theta_p(t)\right] \\ \theta_p(t) &= \int_0^t \omega_p(\sigma)\partial\sigma + \eta_p + \phi_p\left[\omega_p(t)\right] \end{aligned} \qquad (2.11)$$

where P is the number of sinusoids, which can also be a function of time in some applications. Here, the index p denotes that the parameter belongs to one time-evolving sinusoid, called p^{th} partial or track. The time-varying phase $\theta p(t)$ contains the time-varying frequency contribution, a fixed phase offset η_p, which accounts for the fact that the sinusoid will generally not be in phase, and a time-varying frequency-dependent term $\phi_p\left\lfloor\omega_p(t)\right\rfloor$. In the speech production model proposed in [MQ86], $\phi_p\left\lfloor\omega_p(t)\right\rfloor$ is determined by the phase function of the vocal tract.

Sinusoidal synthesis is accepted as perhaps the most powerful and flexible method. Because independent control of every component is available in sinusoidal synthesis, it allows the pitch and length of sounds to be varied independently, as well as it is possible to implement models of perceptually significant features of sound such as inharmonicity and roughness. Another important aspect is the simplicity of the mapping of frequency and amplitude parameters into the human perceptual space. These parameters are meaningful and easily understood by musicians. Recently, a new sinusoidal synthesis method based on spectral envelopes and Fast Fourier Transform has been developed [RD92]. The use of the inverse FFT reduces the computation cost by a factor of 15 compared to oscillators. This technique renders possible the design of low cost real-time synthesizers allowing processing of recorded and live sounds synthesis of instruments and synthesis of speech and the singing voice.

This model makes strong assumptions on the behavior of the time-evolving sinusoidal components. Because of the fact that this system is a frame-based approach, the signal parameters are assumed to be slowly varying in time compared to the analysis frame-rate. In real-world signals, such as a musical note played by a flute, this assumption is almost suited by the property of pseudo-periodicity. These sounds consist mainly of stable sinusoids, which can be perfectly modeled by equation (2.11). But modeling audio signals only as a sum of sinusoids suffers from a major limitation. If the signal contains noisy or impulsive components, an excessive number of sinusoids are needed to model them. The resulting residual signal shows that these broadband processes are present in every natural sound, e.g. the sound of the breath stream in a wind-driven instrument or the sliding of the bow against a string of a cello.

2.1.2.2 The Deterministic plus Stochastic Model

X. Serra and J.O. Smith proposed in [SS90] an extension of the sinusoidal model known as the deterministic-plus-stochastic decomposition.

The use of the estimation of the signal $\hat{x}(t)$ in equation (2.11) is to imply that the sum-of-partials model does not provide an exact reconstruction of the signal *x(t)*. Because of the fact that a sum of sinusoids is ineffective for modeling impulsive events or highly uncorrelated noise, the residual consists of such broadband processes, which correspond to musical important features, such as the turbulent streaming inside the bore of a flute. The stable sinusoids represent the main modes of the vibrating system, and the residual the excitation mechanism and non-linear behaviors. In the case of bowed strings the stable sinusoids are the result of the main modes of vibration of the strings and the noise is generated by the sliding of the bow against the string, plus by other non-linear behavior of the bow-string-resonator system. Since these features are needed to synthesize a natural sound, the additional stochastic component, the residual, should be included in the signal model [SS90,Ser97].

$$x(t) = x_{\det}(t) + x_{stoch}(t) \tag{2.12}$$

The model assumes that the sinusoids are stable partials of the sound and that each one has a slowly changing amplitude and frequency. The deterministic part of the model is the same proposed by the sinusoidal model in equation (2.11).

$$\begin{aligned} x_{\det}(t) &= \sum_{p=1}^{P} A_p(t)\sin\left[\theta_p(t)\right] \\ \theta_p(t) &= \int_0^t \omega_p(\sigma)\partial\sigma + \eta_p + \phi_p\left[\omega_p(t)\right] \end{aligned} \tag{2.13}$$

By assuming that $x_{stoch}(t)$ is a stochastic signal, it can be described as filtered white noise,

$$x_{stoch}(t) = \int_0^t h(t,\tau)u(\tau)\partial\tau \tag{2.14}$$

where *u(τ)* is white noise and *h(t, τ)* is the response of a time varying filter to an impulse at time t . That is, the residual is modeled by the convolution of white noise with a time-varying frequency-shaping filter. This deterministic-plus-stochastic decomposition has been discussed in several later works [Rod97,DQ97,Good97,VM98].

Using the terms deterministic and stochastic brings up the question about the theoretical distinction between these two kinds of processes. Deterministic signals are not only restricted to the sum of sinusoids. However, in this model the class of deterministic signals considered is restricted to quasi-sinusoidal components and the stochastic part is likened to residual.

2.1.3 Techniques employed

2.1.3.1 Spectral Modeling Synthesis (SMS)

The original SMS (Spectral Modeling Synthesis) System is based on the deterministic-plus-stochastic decomposition and has been originally developed by X. Serra. Its basic features are described in [Ser97], but it is still part of the current work of the Music Technology Group (MTG). However, in this section we will present the original system.

This system is restricted to model monophonic sounds and in this work, it will also be restricted to pseudo-harmonic modeling due to the nature of the singing voice approach explored in chapter 3. Both the analysis and synthesis are frame-based processes.

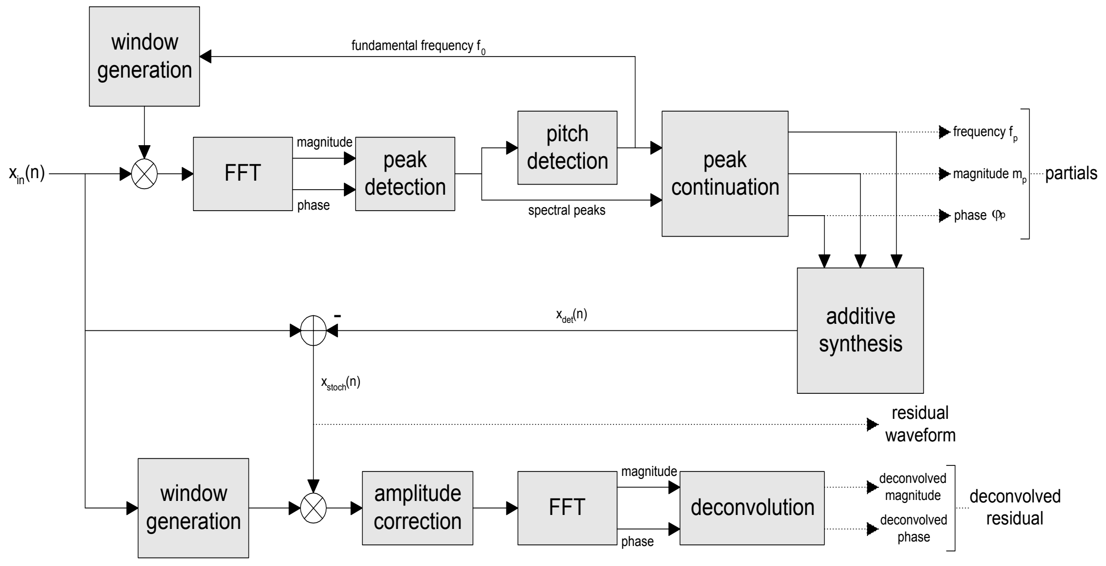


Figure 2.2 Block diagram of the SMS analysis process

Figure 2.2 shows the block diagram of the basic SMS analysis. The input sound $x_{in}(n)$ is processed frame-by-frame at a fixed analysis-frame rate. First, a new section of the sound to be analyzed is multiplied with an analysis window adjusted to the estimated fundamental frequency f_0 in order to achieve a good time-frequency trade-off in the analysis. The Fast Fourier Transform (FFT) obtains its spectrum and the prominent spectral peaks are detected in the magnitude. The relevance of this algorithm is that it detects the magnitude, frequency and phase of the partials present in the original sound. These parameters describe the deterministic part of the input signal. In the pseudo-harmonic case pitch detection is performed to set the window size (pitch-synchronous analysis) and to set also the frequencies of the guides used to find the time-evolving partials. Then, the detected peaks are incorporated in the existing partial trajectories of the last analysis step by means of a peak continuation algorithm.

To calculate the stochastic component $x_{stoch}(n)$ of the current frame, the deterministic component $x_{det}(n)$ is first synthesized and then subtracted from the original input signal $x_{in}(n)$. This is possible because the phases of the original sound are matched and therefore the shape of the time-domain waveform preserved. The stochastic representation is then obtained by performing a spectral fitting of the residual signal. Newest versions of the system perform this subtraction in the frequency domain with a less precise but more efficient method.

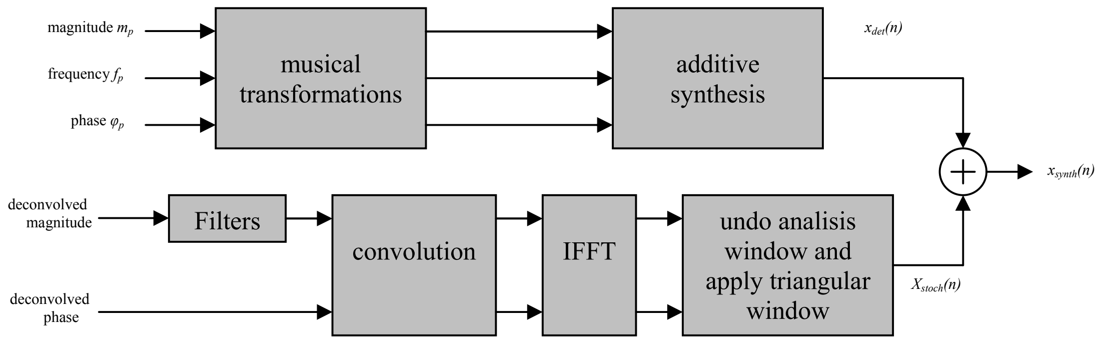


Figure 2.3 Block diagram of the SMS synthesis process

In figure 2.3 the block diagram of the synthesis process is shown. The deterministic part of the signal $x_{det}(n)$ results from the magnitude and frequency trajectories, or their transformation, by generating a sine wave for each trajectory and the superposition of these components (additive synthesis). This can either be done in time or frequency domain, using the inverse FFT approach.

The synthesized stochastic signal $x_{stoch}(n)$ is the result of the inverse-FFT of the possibly modified spectrum. Then, the effects of the analysis window are undone and a triangular window is applied to each frame for a half overlap-add synthesis. Finally, the stochastic and deterministic parts are combined to the sum synthesized sound $x_{synth}(n)$.

2.1.3.2 Spectral Peak Processing (SPP)

The Spectral Peak Processing (SPP) is a technique based on the rigid locked phase vocoders. As pointed out in section 2.1.1, this sort of technique can be somehow related with the sinusoidal models. The main difference with the rigid locked-phase vocoder is that in the SPP not all spectrum local maximums are considered to be peaks and thus sinusoids. SPP only considers as peaks those local maximums that the analysis classifies as harmonic peaks. However, mimicking the locked phase vocoders, the SPP approach considers the spectrum compounded of spectral peak regions each of which consist of a harmonic spectral peak and its surrounding spectra which we assume to contain all the non-perfect harmonic behavior information of the corresponding harmonic peak. The goal of such technique is to preserve the convolution of the analysis window after transposition and equalization transformations.

The analysis and synthesis procedures and dataflow are just the same as in the SMS except from those procedures that specifically regard the residual component, which do not exist here. Moreover, the SPP analysis includes spectrum segmentation into regions out of the harmonic spectral peaks as shown in figure 2.4.

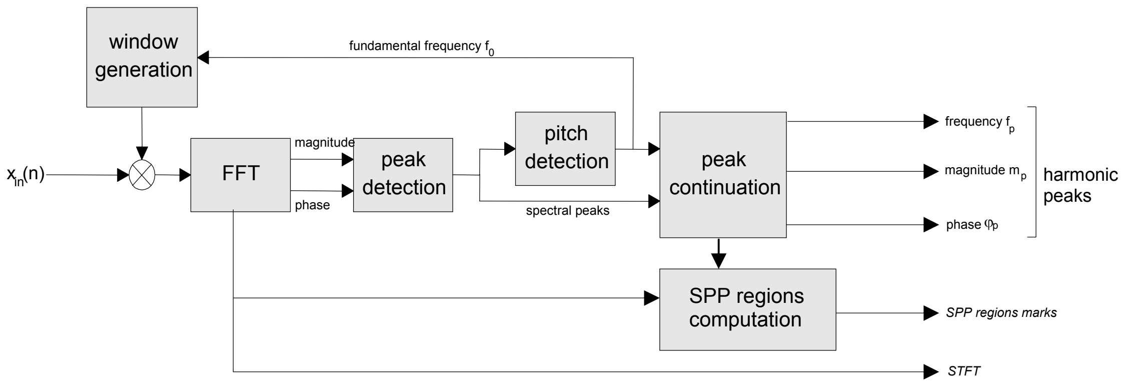


Figure 2.4 Block diagram of the SPP analysis process

The synthesis in the SPP includes the STFT and the SPP spectral peak regions marks aside from the frequency, magnitude and phase of the harmonic spectral peaks.

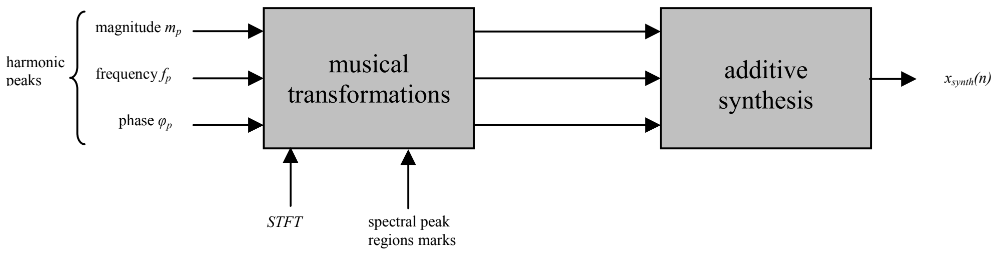


Figure 2.5 Block diagram of the SPP synthesis process

Details on analysis, and transformation / synthesis processes are given in sections 2.2 and 2.4 respectively. From now on, when talking about SPP, we will do it in the context of pseudo-harmonic monophonic sounds.

2.2 Analysis techniques

2.2.1 Magnitude and Phase Spectra Computation

The first step in analyzing the input signal is the computation of the magnitude and phase spectra. This is done by applying a sliding time window to the input signal $x_{in}(n)$ and by performing the short time Fourier transform, STFT. This process isolates time-localized parts of the signal, which are then analyzed using the Discrete Fourier Transformation. The STFT is mathematically defined as

$$X(k,n) = \sum_{m=0}^{N-1} w(m)x(n+m)W_K^{km} \tag{2.15}$$

with

$$W_K = \exp\left(-j\frac{2\pi}{K}\right) \text{ and } K \geq N \tag{2.16}$$

where the DFT is of size K and *w(m)* is a time-domain window with zero value outside the interval *[0,N-1]*. The window and the DFT size do not have to be necessarily of the same size.

The control parameters of the STFT (window size, window type, DFT size and frame rate) have to be chosen according to the characteristics of the sound that will be processed. First, a good frequency resolution is needed to detect the spectral peaks that correspond to the deterministic component. But the phase information is also required for the subtraction of the deterministic part from the original to get the residual.

Figure 2.6 shows a portion of a cello sound before (a) and after applying a Kaiser-Bessel window (b), which is described in [Har78].

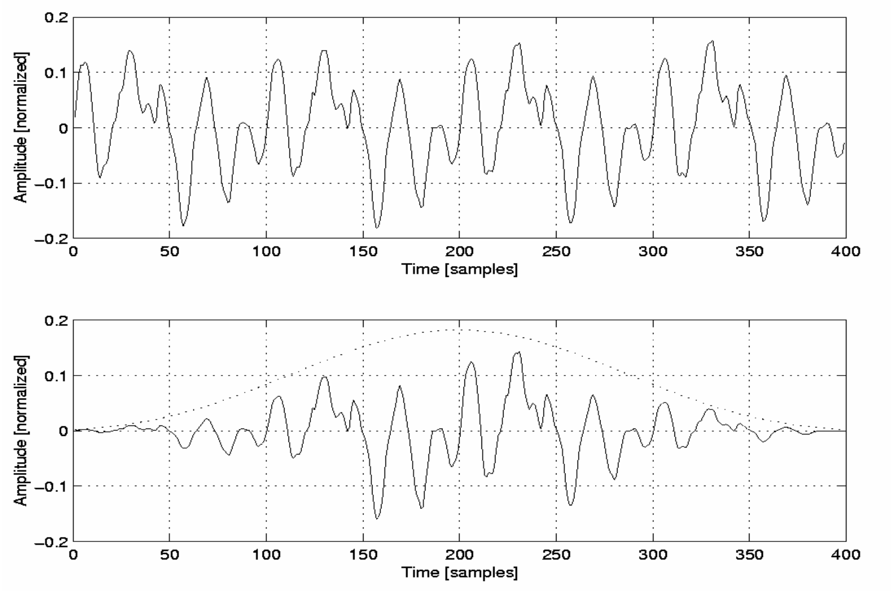


Figure 2.6 Original (a) and windowed (b) portion of a cello sound playing an A_4

The time-domain data should be centered on zero in order to obtain a spectrum free of any linear phase trend caused by the windowing process. Therefore, an odd-length window should be used to achieve this symmetry. Before centering, the data is zero-padded, so that its length is at least 4 times as long as its original length. It should also be a power of 2, so that the fast algorithm of the DFT, the FFT, can be used. The reason for the zero padding is motivated by the precision of the peak detection and is explained in detail in the next section.

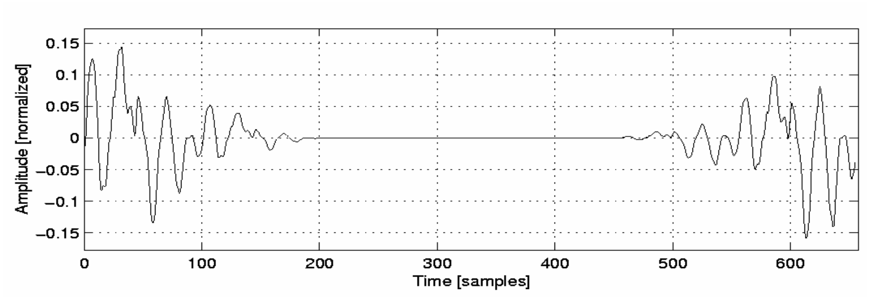


Figure 2.7 Portion of a cello sound zero-padded and centered around zero

The time-frequency compromise of the STFT is very important since it limits the precision of the detection, or even the detectability of partials or impulses in the frequency-domain as well as in the time-domain. For example, it is necessary to have enough frequency resolution if a time-evolving sinusoid should be resolved of the sound. Considering impulses in the stochastic part of the signal, the frequency resolution is not that important compared to the detection of the precise time position. This can be accomplished by using different FFT parameters for the deterministic and stochastic analysis.

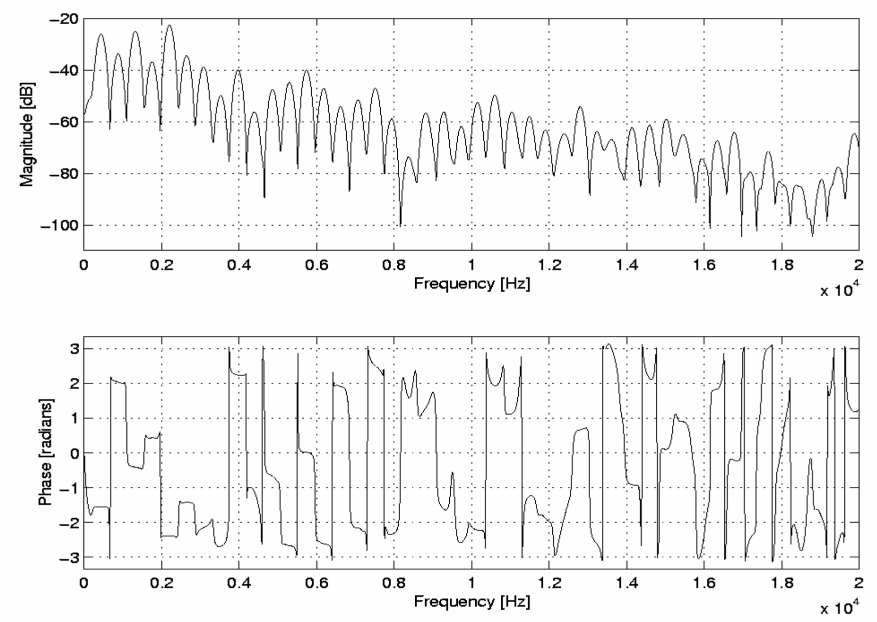


Figure 2.8 Magnitude (a) and phase (b) spectrum of a portion of a cello sound

2.2.2 Peak Detection

Once the spectrum of the current frame is computed, the next step is the detection of the prominent magnitude peaks as the ones observable in figure 2.6. A stable sinusoid has a well-defined frequency representation, depending on the window applied to the time-domain data before computing the DFT. To illustrate the explanation, we will suppose we use a Kaiser-Bessel window, with parameter α depending on the estimated fundamental frequency, for the deterministic analysis. This window is defined as

$$w(n) = \frac{I_0\left(\pi\alpha\sqrt{1-\left(\frac{n}{N/2}\right)^2}\right)}{I_0(\pi\alpha)} \qquad \text{with} \qquad 0 \le |n| \le \frac{N}{2} \tag{2.17}$$

where

$$I_0(x) = \sum_{k=0}^{\infty}\left(\frac{\left(\frac{x}{2}\right)^2}{k!}\right) \tag{2.18}$$

The parameter $\pi\alpha$ is half of the time-bandwidth product. As presented in [Har78], the transform is approximately that of

$$W(\Omega) \approx \frac{N}{I_0(\pi\alpha)}\frac{\sinh\left(\sqrt{\alpha^2\pi^2-\left(N\Omega/2\right)^2}\right)}{\sqrt{\alpha^2\pi^2-\left(N\Omega/2\right)}} \qquad \text{with} \qquad 0 \le \Omega \le 2\pi \tag{2.19}$$

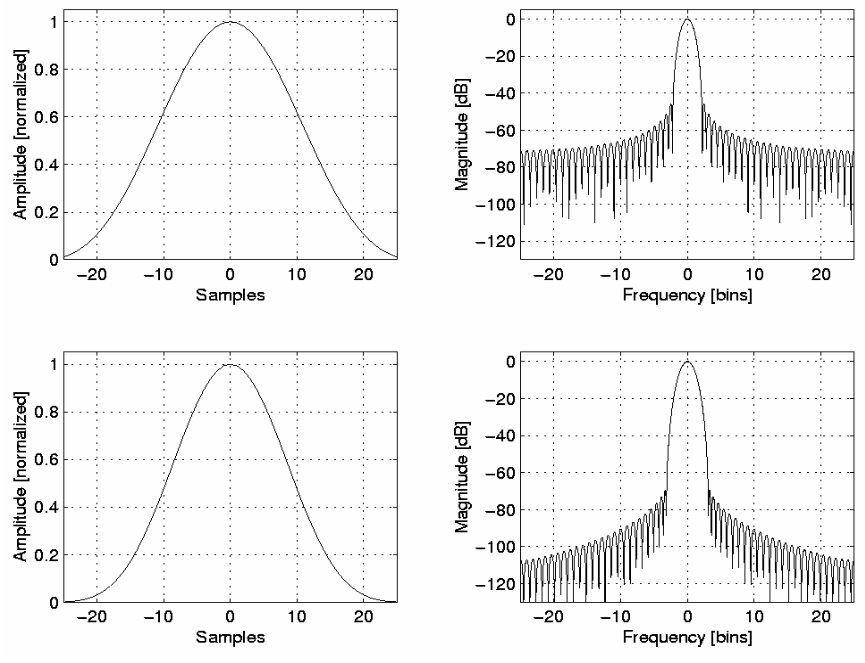


Figure 2.9 Kaiser-Bessel window with α =2.0 (a) and its log-magnitude (b) and with α =3.0 (c), (d)

This windows and their spectra for the parameters α=2.0 and α=3.0 are shown in figure 2.9 and a stable sinusoid should ideally have the same spectral shape. However, in practice this is rarely the case since most of the sounds are not perfectly periodic and the spectrum of the window is not band limited, causing a superposition of the sidelobes of different sinusoidal components and thus violating the main-lobe shape. Due to this effect, it is difficult to rely on the shape of a peak as a criterion whether it is a stable sinusoid or not without tolerating some mismatch. On the other hand, a sinusoid without constant amplitude inside the frame boundaries widens this main-lobe significantly and adds an in- or decreasing trend to the corresponding phase. This can be easily shown for the simple case of a rectangular window.

The rectangular window of the size N samples and its Fourier transform are given by

$$w(n) = \begin{cases} 1 & \text{for} \quad |n| \le \dfrac{N-1}{2} \\ 0 & \text{for} \quad |n| > \dfrac{N-1}{2} \end{cases} \tag{2.20}$$

$$W(\Omega) = \frac{\sin\left(N\Omega/2\right)}{\sin\left(\Omega/2\right)} \qquad \text{with} \quad \Omega = \omega T \tag{2.21}$$

and a simple time-domain signal, such as a cosine function with time-varying amplitude, can be written as

$$x(n) = A(n)\cos(\Omega_0 n) = \frac{A(n)}{2}\left[\exp(j\Omega_0 n) + \exp(-j\Omega_0 n)\right] \tag{2.22}$$

and its spectrum as

$$X(\Omega) = F\{x(n)\} = A(\Omega) * \pi\left[\delta(\Omega - \Omega_0) + \delta(\Omega + \Omega_0)\right] \tag{2.23}$$

where $A(\Omega)$ is the Fourier transform of the time-varying modulation *A(n)*. The windowing in the time-domain refers to a convolution in the frequency-domain, so that the windowed signal is given by

$$x_w(n) = A(n)\cos(\Omega_0 n) w(n) \tag{2.24}$$

and its spectrum by

$$X_w(\Omega) = A(\Omega) * \pi\left[\delta(\Omega - \Omega_0) + \delta(\Omega + \Omega_0)\right] * \frac{\sin\left(N\Omega/2\right)}{\sin\left(\Omega/2\right)} \tag{2.25}$$

If *A(n)=A=const.*, meaning that the signal is not amplitude modulated inside the frame boundaries, the spectrum of *A(n)* can be written as $A(\Omega) = 2\pi A\delta(\Omega)$ and thus represents the ideal case where the main-lobe of the window is not widened. But if $A(n) \neq const.$, then its appropriate Fourier transform $A(\Omega)$ has a significant bandwidth that causes the widening due to the convolution process. $A(\Omega)$ Also adds an in- or decreasing phase inside the main-lobe range of the peak.

Figure 2.8 shows that the phase progression inside the main-lobe is not flat in every case and this indicates that the measured amplitude or frequency is not constant over the frame duration. This observation can also be used as a stability criterion for the estimated parameters. A practical solution to this problem is to detect as many peaks as possible, measure their main-lobe width, flatness of the phase, and delay the decision of what is a deterministic, stable sinusoid, to the peak continuation algorithm following the peak detection.

The detection of peaks in the magnitude spectrum is very simple. A peak is defined as a local maximum in the magnitude spectrum, which is mathematically given by two conditions, the necessary condition

$$|X'(k_{max})|=0 \tag{2.26}$$

and the sufficient condition

$$|X'(k_{max})|=|X''(k_{max})|=...=|X^{(n-1)}(k_{max})|=0, \qquad |X^{(n)}(k_{max})| \neq 0 \tag{2.27}$$

If n is odd, then |X(k)| has no local extremum. But if n is even and $|X^{(n)}(k_{max})| <0$, then |X(k)| has a strict local maximum.

Due to the fact that $|X(k)|$ is a vector of discrete elements, the difference between two values can be computed instead of the derivation. Figure 2.10 shows the magnitude and phase spectrum of the cello sound with the detected peaks at the bin indices $k_i^{\max}$. The amplitude $A(k_i^{\max})$ can be directly written as

$$A(k_i^{\max}) = \frac{2}{\xi}\left|X(k_i^{\max})\right| \qquad \text{with } i = 1, 2, \ldots \tag{2.28}$$

and

$$\xi = \sum_{n=0}^{N-1} w(n) \tag{2.29}$$

The phases are given by

$$\varphi_i = \arctan\left(\frac{\operatorname{Im}\{X(k_i^{\max})\}}{\operatorname{Re}\{X(k_i^{\max})\}}\right) \quad \text{so that} \quad -\frac{\pi}{2} < \varphi_i \leq \frac{\pi}{2} \tag{2.30}$$

Due to the sampled nature of the spectra of the FFT, each peak is only accurate within half a sample. A spectral sample represents a frequency interval of f_s / K Hz, where f_s is the sampling rate and K is the FFT size. Thus, if the frequency $f_{\sin e}$ of a sinusoid corresponds to a bin frequency $i \cdot f_s / K$ with $i \in [0, K-1]$, it can be perfectly identified and its magnitude and phase can be exactly extracted. In the off-bin case $f_{\sin e} \neq i \cdot f_s / K$, the peak suffers from spectral leakage and cannot be detected correctly. However, it is possible to refine the estimation accuracy of this peak by interpolation.

First, an oversampled FFT, meaning $K>N$ in equation (2.16), is one possibility to improve the accuracy of the detection. But to obtain a high frequency accuracy on the level of 0.1% of the distance from the top of an ideal peak to its first zero-crossing (in the case of the rectangular window), using only zero-padding, the required factor would be $K/N \approx 1000$ [Ser97]. A practical solution to this problem is to zero-pad only enough so that quadratic spectral interpolation, using only samples surrounding the maximum-magnitude sample, suffices to refine to the required accuracy.

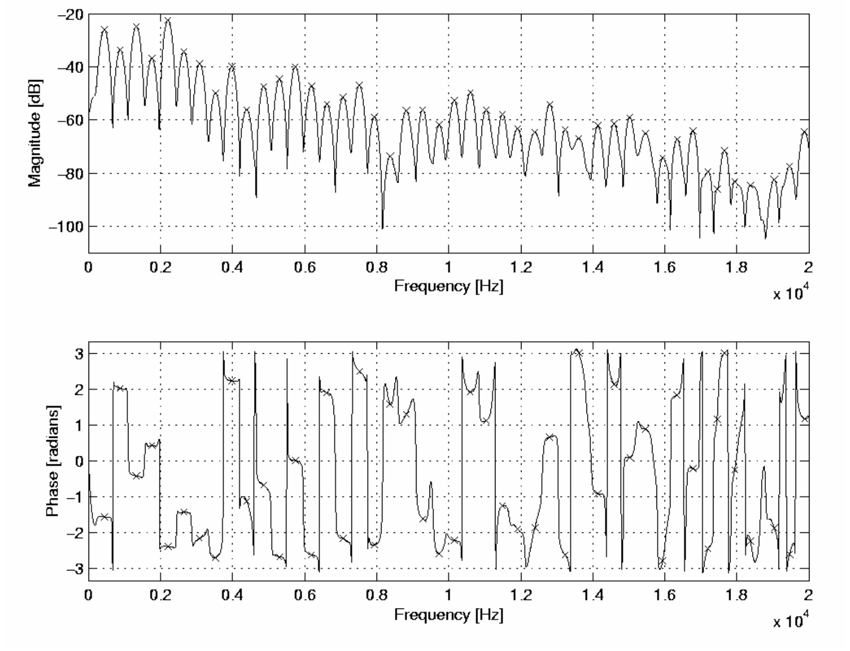


Figure 2.10 Peaks detected in the magnitude of the cello sound

Thus, the estimate is refined by quadratic interpolation, as given by

$$M(k+\alpha) = \sum_{i=-1}^{1} c_i(\alpha) M(n+i) \tag{2.31}$$

with

$$
\begin{aligned}
c_{-1} &= \frac{1}{2}\alpha(\alpha-1) \\
c_0 &= 1-\alpha^2 \\
c_1 &= \frac{1}{2}\alpha(\alpha+1)
\end{aligned}
\tag{2.32}
$$

where $M(k)$ is the magnitude at the bin k and α denotes the point that should be interpolated. To calculate the appropriate $\alpha_i^{\max}$ for the local maximum of each detected peak $k_i^{\max}$, the following equation must be solved,

$$
\frac{\partial M\left(k_i^{\max}+\alpha_i^{\max}\right)}{\partial \alpha_i^{\max}} = 0 \tag{2.33}
$$

so that

$$
\alpha_i = \frac{1}{2}\left(\frac{M\left(k_i^{\max}-1\right)-M\left(k_i^{\max}+1\right)}{M\left(k_i^{\max}-1\right)-2M\left(k_i^{\max}\right)+M\left(\left(k_i^{\max}+1\right)\right)}\right) \tag{2.34}
$$

Finally, the refined estimate can be written as

$$
\hat{k}_i^{\max} = k_i^{\max} + \alpha_i^{\max} \tag{2.35}
$$

But it is important to notice that this spectral interpolation helps only identifying off-bin frequencies that are widely separated, but does not improve the resolution of closely spaced components. Two closely spaced sinusoids can only be resolved if their frequency difference is at least one bin in an N-point FFT, using a rectangular window. For the Kaiser-Bessel window with α =2.0, the main-lobe is about twice as wide, see figure 2.9 (a), as that of the rectangular window, and as a result, this Kaiser-Bessel window must span two signal periods, *2N* samples, to achieve a resolution of harmonic components. If α =3.0, the necessary number of signal periods increases to three.

2.2.3 Pitch Detection

The term fundamental frequency can be defined as the common divisor of the harmonic series that best explains the spectral peaks found in the spectrum. Pitch is the equivalent term in the psychoacoustics context. Thus pitch does not get its value from mathematical formulas but from perception based rules. However in this work we will refer to both terms indiscriminately mostly because our goal is to detect the pitch rather than the fundamental frequency. It is easy to understand the differences between these two terms when listening to a loud hoarse voice or to long note releases, where the pitch perceived differs from the fundamental frequency over 12 semitones.

There have been different approaches to the problem of fundamental frequency estimation. Time-domain based techniques, frequency-domain techniques, and also other techniques, based on cepstrum or linear predictive coding, have been proposed. Because both SMS and SPP are based on the estimation of parameters in the frequency-domain, it is more efficient to use the results of the previous calculations in a frequency-domain approach. The Two-Way Mismatch Procedure proposed by Maher and Beauchamp in [MB94] is such an algorithm, measuring the goodness of a possible fundamental frequency f_0 by calculating a mismatch error between the estimated peak frequencies and the ones of the ideal harmonic series and vice versa.

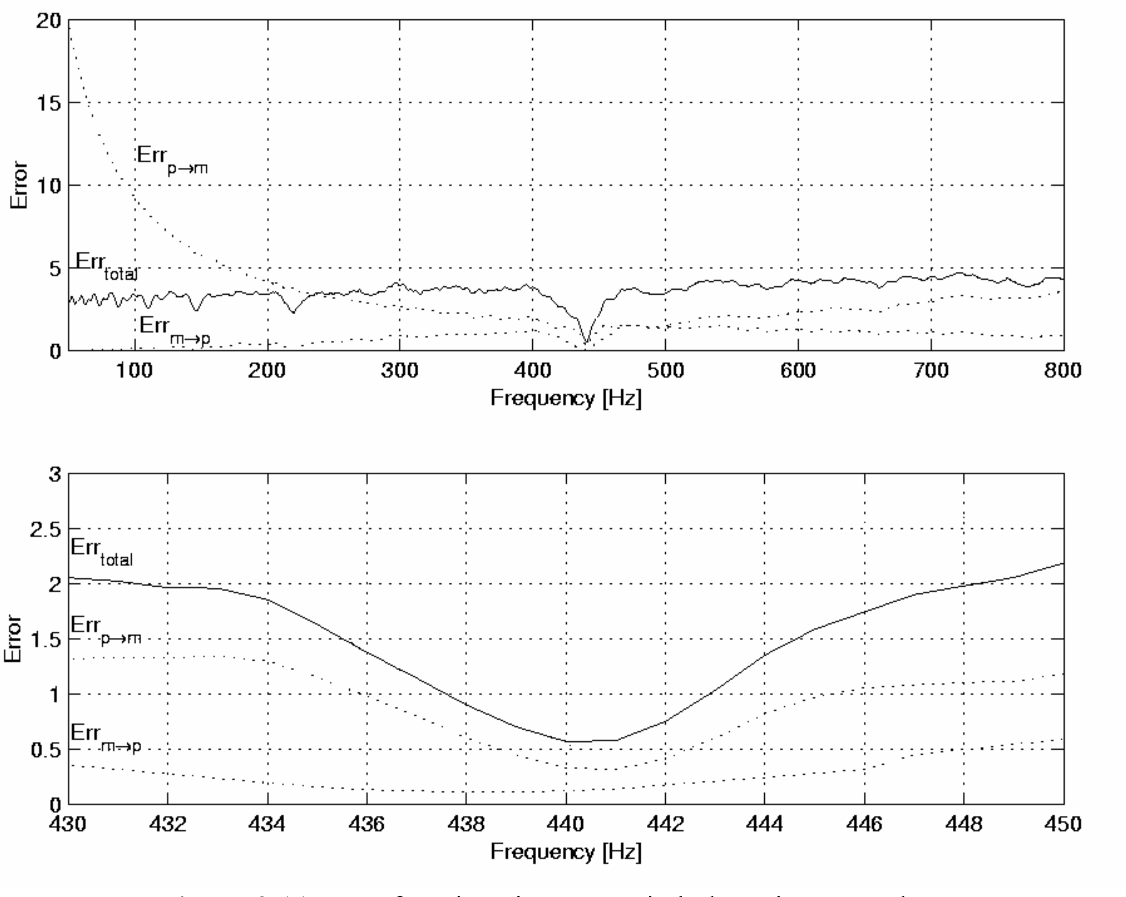


Figure 2.11 Error functions in TWM pitch detection procedure

This two-way mismatch helps to avoid octave errors in the estimation by applying a penalty for partials that are present in the measured data but are not predicted, and also for partials whose presence is predicted but do not actually appear in the measured sequence. The predicted-to-measured error is defined as:

$$Err_{p\to m} = \sum_{n=1}^{N} \Delta f_n (f_n)^{-p} + (\frac{a_n}{A_{\max}}) \times \left[q \Delta f_n \cdot (f_n)^{-p} - r \right] \qquad (2.36)$$

where *Δf_n* is the difference between a predicted and its closest measured peak f_n, and a_n are the frequency and magnitude of the predicted peaks, and A_{max} is maximum peak magnitude.

The measured to predicted error is defined as:

$$Err_{m\rightarrow p} = \sum_{k=1}^{K} \Delta f_k (f_k)^{-p} + (\frac{a_k}{A_{\max}}) \times \left[q\Delta f_k \cdot (f_k)^{-p} - r \right] \qquad (2.37)$$

where Δf_k is the difference between a measured and its closest predicted peak f_k, and a_k are the frequency and magnitude of the measured peaks, and A_{max} is maximum peak magnitude.

The total error is:

$$Err_{total} = Err_{p\rightarrow m} / N + \rho Err_{m\rightarrow p} / K \qquad (2.38)$$

The values of the included parameters are proposed by Maher and Beauchamp to be p=0.5, q=1.4, r=0.5 and ρ=0.33 and proved to work optimally for most of the sounds. Figure 2.11 shows the errors functions for the case of a cello playing an A_4. It can be seen that the total error has a minimum at $f_0 \approx 440$ Hz, which is the fundamental frequency of the played note.

On top of the TWM procedure several improvements and some modifications in order to adapt it to the case of the singing voice have been done. Some of these are described in [Cano98].

2.2.4 Peak Continuation

The peak continuation algorithm is in charge of adding the detected spectral peaks to the incoming peak trajectories. One of the first continuation algorithms was proposed by McAulay and Quartieri in [MQ86]. It matches the closest spectral peak in frequency to each of the incoming trajectories of the last processed frame and thus maybe omitting the musical meaning of these time-evolving sinusoidal tracks. If, for example, a partial changes substantially in frequency from one frame to the next, this algorithm easily switches from the partial that it is tracking to another one, which at that point is closer in frequency.

Restricting the analysis to monophonic sounds, the information of the fundamental frequency f_0 can be used to set musical meaningful guides, advancing in time through the spectral peaks, looking for the appropriate ones (according to specific criteria) and forming trajectories out of them. Taking advantage of the pseudo-periodicity property of the processed signals, the guides can be set to the harmonic frequencies f_i^{harm} of the estimated fundamental f_0.

$$f_i^{guide} = f_i^{harm} = i \cdot f_0 \qquad \text{with} \qquad i = 0, 1, 2, \ldots \qquad (2.39)$$

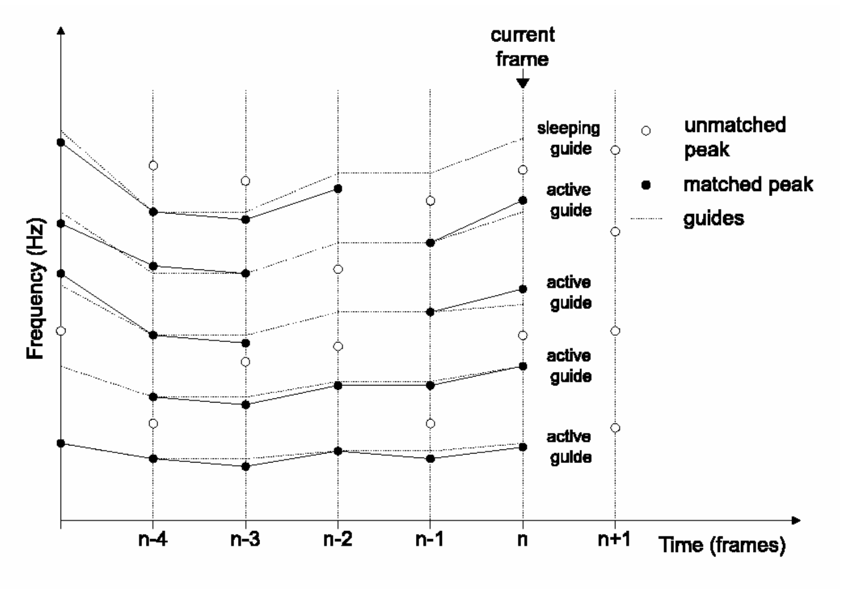


Figure 2.12 Illustration of the peak continuation process

Now, each guide p searches the detected peak frequencies for a possible match according to the following criterion

$$\left| f_j^{peaks} - f_p^{guide} \right| \le \Delta f \tag{2.40}$$

where Δf is the frequency range searched for a match. During this search, the algorithm also has to be aware of not matching peaks that already belong to other trajectories and of avoiding the crossing of trajectories as the result of the continuation process. The criterion in equation (2.40) can be refined by using also the magnitude information of the peak matched in the last frame to this trajectory because it makes no musical sense matching a peak that is close in frequency but with a totally different magnitude when searching for stable partials. The criterion can be extended to the following two conditions

$$\begin{aligned} &\left| f_j^{peaks} - f_p^{guide} \right| \le \Delta f \\ &\left| M(f_j^{peaks}) - M(f_p^{guide}) \le \Delta M \right| \end{aligned} \tag{2.41}$$

with

$$M(f) = \left| X(f) \right| \tag{2.42}$$

where Δf is the frequency range, and ΔM is the magnitude range searched for a match. *M(f)* is the log-magnitude spectrum. If no match is possible in the current frame, the trajectory is turned off in the sense of setting its amplitude to zero. But depending on the duration of this turned-off state, the trajectory can be considered as disappeared, or in case of a short dropout, as a failure in the detection process. If the duration of this dropout is shorter than a fixed duration, for example 30 ms [Vas96], it makes sense to repair this trajectory by linear interpolation of its magnitude and an appropriate calculation of the phase for the missing frames, as given by

$$f_{p,l} = i \cdot f_{0,l} \tag{2.43}$$

$$M_{p,l} = M_{p,k} + \left(M_{p,m} - M_{p,k}\right)\frac{l}{L} \qquad \text{with } l \in \left]k,m\right[\tag{2.44}$$

$$\varphi_{p,l} = \varphi_{p,k} + 2\pi f_{p,l-1}\frac{S}{f_s} + \pi\left|f_{p,l} - f_{p,l-1}\right|\frac{S}{f_s} \tag{2.45}$$

where the index *i* denotes the number of the perfect harmonic corresponding to the trajectory *p*, *l* the number of the current frame, *k* and *m* the appropriate last and next frame with a correctly detected set of estimates, and *S* the frame size in samples. If the trajectory can really be considered as disappeared, its duration is checked in order to keep only the trajectories that are stable partials and thus belong to the deterministic part of the signal.

The algorithm presented here is one approach to the peak continuation problem the but different strategies and techniques can be applied to solve this problem (i.e. Hidden Markov Models [Gar92,DGR93]).

2.2.5 SMS Specific: Stochastic Analysis

The stochastic analysis in the SMS contains two steps, first, the calculation of the stochastic component, also called deterministic subtraction, and second, its modeling. In the deterministic subtraction the sinusoidal components of the estimated deterministic part are subtracted from the original sound in the time domain. The resulting residual can be used to measure how well the deterministic features in the sound have been extracted. If partials still remained in the residual, it should be reanalyzed, using other analysis parameter settings in order to help the algorithm to overcome its problems. This can be done, for example, by forcing the pitch detection to find the fundamental in a smaller frequency range or by changing the minimum duration of a trajectory that is considered as deterministic. This user interaction improves the analysis by using a-priori knowledge. If the sound was not recorded in the ideal situation, the residual will also tend to contain more than just the stochastic part of the sound, such as reverberation or background noise. But ideally, it should be as close as possible to a stochastic signal.

The deterministic subtraction contains two steps, first, the synthesis of the deterministic part from the set of peak trajectories, and finally, the frame-by-frame subtraction. A series of sinusoids can be synthesized from the trajectories that reproduce the instantaneous phase and amplitude of the original partials. Thus, it is possible to obtain a residual part by subtracting the synthesized sinusoids from the original sound.

One frame of the deterministic part is generated by

$$x_{\text{det}}(n) = \sum_{p=1}^{P} \hat{A}_p \cos\left[m\hat{\omega}_p + \hat{\varphi}_p\right] \qquad \text{with} \quad n=0,1,2,\ldots,S\text{-}1 \tag{2.46}$$

where P is the number of trajectories present in the current frame and S is the frame size. To avoid clicks caused by discontinuities at the frame boundaries, the frame-rate partial parameters $\{\hat{A}_p, \hat{\omega}_p, \hat{\varphi}_p\}$ are smoothly interpolated from frame to frame, resulting in oscillator control functions $\hat{A}_p(n)$ and $\hat{\omega}_p(n)$ for every trajectory. We interpolate by using low-order polynomial models, such as linear amplitude and cubic total phase. The instantaneous amplitude $\hat{A}_{p,i}(n)$ can be obtained by calculating

$$\hat{A}_{p,i}(n) = \hat{A}_{p,i-1} + \frac{(\hat{A}_{p,i} - \hat{A}_{p,i-1})}{S} n \tag{2.47}$$

where n=0,1,…S-1 is the time sample into the i^{th} frame.

Since frequency and phase values are tied together by the equation $f = \partial\varphi / \partial t$, the instantaneous phase $\hat{\theta}_{p,i}$ is given by

$$\hat{\theta}_{p,i} = \hat{\omega}_{p,i} n + \hat{\varphi}_{p,i} \tag{2.48}$$

Given that four variables affect the instantaneous phase: $\hat{\omega}_{p,i}$, $\hat{\varphi}_{p,i}$, $\hat{\omega}_{p,i-1}$, $\hat{\varphi}_{p,i-1}$, we need three degrees of freedom for its control. A cubic polynomial interpolation function is used as proposed by McAulay and Quartieri in [MQ86]. This is mathematically given by

$$\hat{\theta}(n) = \xi + \kappa n + \eta n^2 + \zeta n^3 \tag{2.49}$$

This equation can be solved as described in [MQ86] and only the results will be presented here.

$$\hat{\theta}_{p,i}(n) = \varphi_{p,i} + \hat{\omega}_{p,i-1} n + \eta n^2 + \zeta n^3 \tag{2.50}$$

where η and ζ are calculated by using the beginning and end conditions at the frame boundaries,

$$\eta = \frac{3}{S^2}(\hat{\varphi}_{p,i} - \hat{\varphi}_{p,i-1} - \hat{\omega}_{p,i-1} S + 2\pi L) - \frac{1}{S}(\hat{\omega}_{p,i} - \hat{\omega}_{p,i-1}) \tag{2.51}$$

$$\zeta = \frac{2}{S^3}(\hat{\varphi}_{p,i} - \hat{\varphi}_{p,i-1} - \hat{\omega}_{p,i-1} S + 2\pi L) - \frac{1}{S^2}(\hat{\omega}_{p,i} - \hat{\omega}_{p,i-1}) \tag{2.52}$$

This gives a set of interpolating functions depending on the value of L, among which we select the maximally smooth function by choosing L to be an integer closest to χ, where χ is

$$\chi = \frac{1}{2\pi}\left[(\hat{\varphi}_{p,i-1} + \hat{\omega}_{p,i-1} S - \hat{\varphi}_{p,i}) + \frac{S}{2}(\hat{\omega}_{p,i} - \hat{\omega}_{p,i-1})\right] \tag{2.53}$$

Including these oscillator control functions in the equations (2.47) and (2.49), the deterministic part can be written as

$$x^i_{\det}(n) = \sum_{p=1}^{P_i} \hat{A}_{p,i}(n)\cos\left[\hat{\theta}_{p,i}(n)\right] \qquad \text{with} \quad n=0,1,2,\dots,S\text{-}1 \qquad (2.54)$$

This deterministic part is interpolated for every parameter set $\{\hat{A}_{p,i-1}, \hat{\omega}_{p,i-1}, \hat{\varphi}_{p,i-1}\}$ of one frame to the one $\{\hat{A}_{p,i}, \hat{\omega}_{p,i}, \hat{\varphi}_{p,i}\}$ of the following frame in a smooth way. Once this deterministic component is calculated, it can be subtracted from the original sound to obtain the residual part. This is simply done by

$$x_{stoch}(n) = x(n) - x_{\det}(n) \qquad \text{with} \quad n=0,1,2,\dots,S\text{-}1 \qquad (2.55)$$

For the example of the cello sound the deterministic component and the calculated residual are shown in figure 2.13. To obtain the spectrum, a window *w(n)* is applied to the stochastic component $x_{stoch}(n)$ and a FFT of the size *N* is calculated. This is mathematically given by

$$\hat{x}_{stoch}(n) = w(n)\cdot(x(n) - x_{\det}(n)) \qquad \text{with} \quad n=0,1,2,\dots,N\text{-}1 \qquad (2.56)$$

$$X_{stoch}(k) = DFT\{x_{stoch}(n)\} \qquad \text{with} \quad k=0,1,2,\dots,K\text{-}1 \qquad (2.57)$$

The window used in this approach is a Blackman-Harris 92dB as described in [Har78], and its size is *N* samples. The magnitudes of the deterministic and stochastic component can be seen in figure 2.14 (a), (b). The magnitude of the residual is very close to a white spectrum and the only components left in the residual are the magnitude peaks with frequencies greater than 18 kHz and a low frequent part, which are both not stable in time and thus considered to be not deterministic.

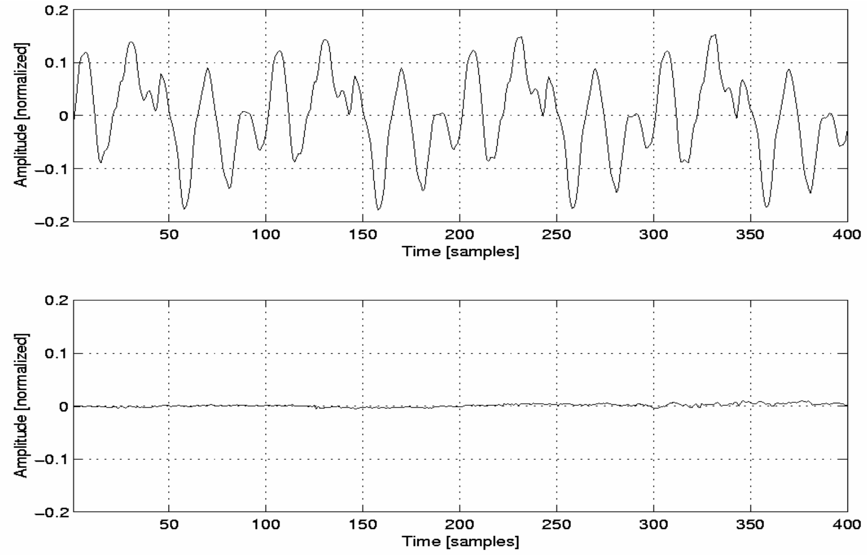


Figure 2.13 Deterministic subtraction: deterministic part of a cello sound (a) and the residual (b)

In [SS90,Ser97] a stochastic approximation has been proposed that models the residual component as filtered white noise. Therefore, a line-segment approximation is fitted to the log-magnitude spectrum curve. Since then, other approximation techniques have been proposed to improve the modeling, such as a speech coding technique called MPLP [DQ97] or a parametric approach, using perceptual criteria in [Good97].

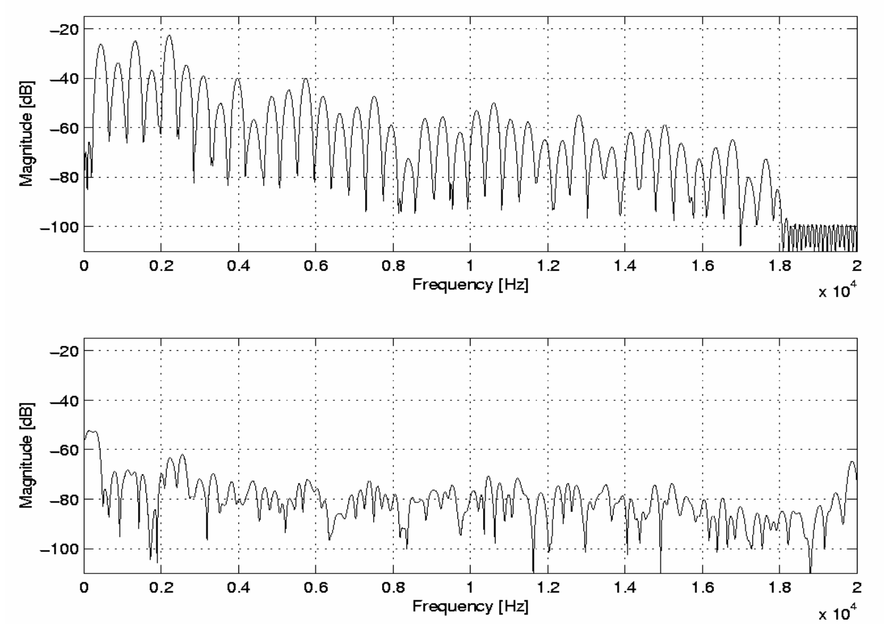


Figure 2.14 Magnitude of the deterministic part of a cello sound (a) and the residual (b)

As already mentioned in section 2.1.3 the residual part of the sinusoidal plus stochastic decomposition can be computed in the frequency domain [Bon97]. Due to the windowing process in the time-domain the calculated spectrum suffers from limited resolution and the sinusoid is also not represented by an ideal Dirac impulse (this is correct for the continuous Fourier transform, but considering the sampled nature of the discrete Fourier transform, this would only be the case for a frequency lying on a bin-frequency) but rather by the modulated Fourier transform of the used window. In the case of the Blackman-Harris window the window spectrum can be written as

$$W(\Omega) = \sum_{m=0}^{N/2} (-1)^m \frac{a_m}{2} \left[D\left(\Omega - \frac{2\pi}{N} m\right) + D\left(\Omega + \frac{2\pi}{N} m\right) \right] \qquad (2.58)$$

with the Dirichlet kernel

$$D(\Omega) = \exp\left(j \frac{\Omega}{2}\right) \frac{\sin\left[\frac{N}{2}\Omega\right]}{\sin\left[\frac{1}{2}\Omega\right]} \qquad \text{with} \qquad \Omega = \omega T \qquad (2.59)$$

subject to constraint

$$\sum_{m=0}^{N/2} a_m = 1 \tag{2.60}$$

For the Blackman-Harris 92 dB window the parameters a_i are a_1=0.35875, a_2=0.48829, a_3=0.14128 and a_4 =0.01168, satisfying the condition (2.50).

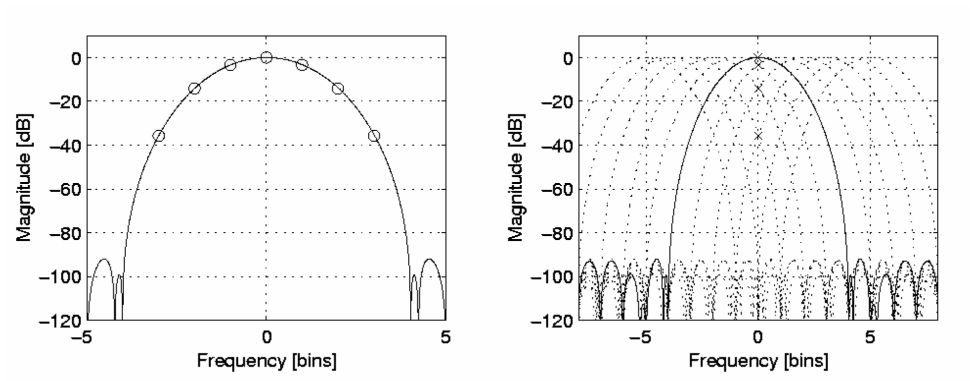


Figure 2.15 Spectral values of the main-lobe (a) and their effect in the superposition (b)

The windowing process in the time-domain corresponds to a convolution in the frequency-domain, so that the ideal spectrum is convolved with the spectrum of the window. The discrete convolution is mathematically given by

$$X_w(k) = \sum_m X(k-m)W(m) \tag{2.61}$$

where *W(n)*, *X(n)* and $X_w(n)$ are the spectra of the window, the signal without and with the windowing effect, respectively. Assuming that a signal *x(n)* consists only of ideal sinusoids with different frequencies, the resulting spectrum will be a superposition of window spectra modulated with the different sinusoidal frequencies.

$$X_w(k) = \sum_{i=0}^{P-1} W(k-k_i) \tag{2.62}$$

with

$$k_i = \frac{\Omega_i K}{2\pi} \tag{2.63}$$

where *K* is the DFT size, f_s is the sampling rate and Ω_i the normalized frequency of the i^{th} sinusoid.

The Blackman-Harris 92dB window has a main-lobe width of 9 bins, as seen in figure 2.15 (a), and their second lobes are -92 dB below. Thus, if we consider a sinusoid is represented in the spectrum by the main-lobe bins as in [RD92, GR94], we can subtract the sinusoidal component by just filling a spectrum buffer with the 9 main-lobe bins for each sinusoid and then subtract this spectrum buffer from the original spectrum buffer. The phase of these sinusoids is set to the sinusoidal peak phase found in the analysis in all main-lobe bins. The subtraction operation is performed in the complex plane.

2.2.6 SPP Specific: Spectrum Segmentation

The outcome of the SPP analysis are the harmonic peaks (frequency, amplitude and phase), the STFT, and the SPP regions marks. Remember SPP divides the spectrum into a set of regions, each of which belongs to one harmonic spectral peak and its surroundings.

Two different algorithms were introduced in section 2.1.1 to segmentate the spectrum into regions out of peaks information. These techniques can also be applied in the SPP case. The region boundary is set to be at the lowest local minimum spectral amplitude between two consecutive peaks as illustrated in figure 2.16(b). If there are no local minimums, then the boundary is set to be at the middle frequency between two consecutive peaks as illustrated in figure 2.16(a).

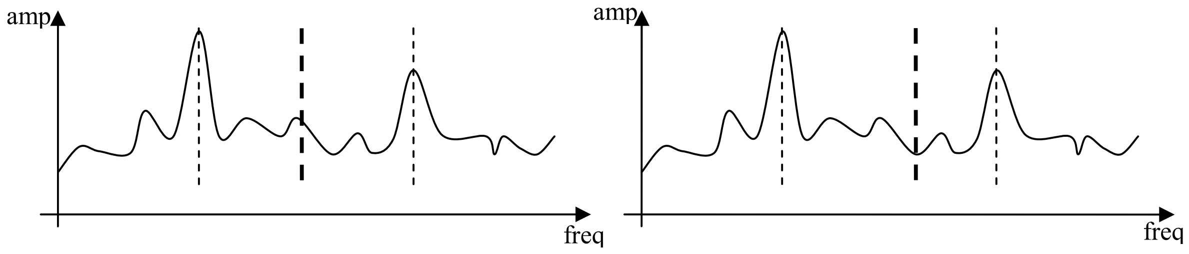


Figure 2.16 SPP region boundaries computation techniques: (a) mid frequency value between consecutive harmonic peaks (b) min amplitude value between consecutive harmonic peak's corresponding frequency

2.3 Transformation techniques

Voice transformation is a very popular issue nowadays and not only in the computer music community. An example of this popularity is the fact that it is hard to find these days a popular music production without any vocal alienation. In this chapter the methods used for voice transformation will be introduced.

2.3.1 SMS based transformations

For the SMS case, in the deterministic representation each function pair, amplitude and frequency, accounts for a partial of the original sound. The manipulation of these functions is easy and musically intuitive. In the stochastic representation we can change the shape of each of the envelopes and the time-varying magnitude or gain to modify the data. This corresponds to a filtering of the stochastic signal, whose 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.

2.3.2 SPP based transformations

For the SPP case, the main drawback of the SMS technique, which is the critical discrimination between sinusoidal and residual (there are always remaining residual components in the sinusoidal part and vice versa) is avoided. This makes transformations using SPP technique more robust and consistent without losing SMS flexibility. With SPP the local behaviour of the peak region has to be preserved both in amplitude and phase. To do so, the delta amplitude relative to the peak's amplitude and the delta phase relative to the peak's phase are kept unchanged after spectral transformations.

2.3.2.1 Equalization

Equalizing entails timbre changing. From the SPP analysis we get a spectrum with the harmonic peaks and the regions marks. The harmonic peaks define the original spectral envelope which is shown with a continuous line in the figure 2.17. On the other hand, the desired spectral shape is drawn as a dashed line in same figure.

The SPP equalization is done by calculating for each region the amount needed so that each harmonic's amplitude matches the timbre envelope. Then these amounts (drawn as arrows in figure 2.17) are added to all the bins that belong to a region (amplitude dB addition). Therefore, the spectrum amplitude of a region is just shifted up or down and it keeps its local behavior.

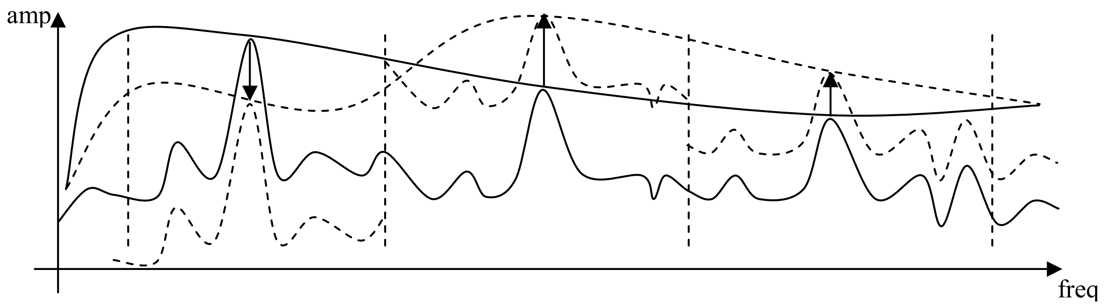


Figure 2.17 SPP equalization

2.3.2.2 Transposition

Transposition implies multiplying the harmonic's frequencies by a constant value. This value is called transposition factor and from now on we will refer to it as *transp*. SPP transposition is accomplished by shifting the SPP regions in frequency. The amount of frequency shift Δf_{Transp} calculated for each harmonic peak is applied as a constant to its whole region (the linear frequency displacement for all the bins of a region is the same). Therefore, the local amplitude spectrum of each region is kept as it is, thus preserving the window convolution with each harmonic peak. For the i^{th} harmonic peak, the new frequency value is

$$f_i^{new} = transp \cdot f_i \tag{2.64}$$

and the shift to be applied

$$\Delta f_{Transp} = \left| f_i^{new} - f_i \right| = f_i \cdot \left| transp - 1 \right| \tag{2.65}$$

In figure2.17 a transposition (with *transp* > 1) is illustrated. The original SPP spectrum is shown on the top, the arrows show the displacement in frequency that is applied to each harmonic, and at the bottom we can see the resulting spectrum. The gaps between SPP regions are filled with -200 dB constant spectral amplitude. In those cases in which *transp* < 1 (we are transposing to a lower pitch) the SPP regions overlap in the resulting spectrum. This overlapping is performed by adding the complex values at each spectral bin.

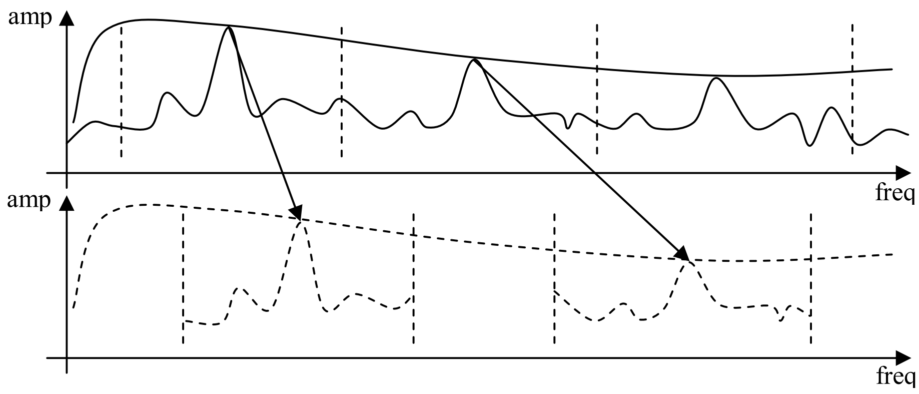


Figure 2.17 Example of an SPP transposition with transposition factor > 1 preserving the original spectral shape

It's important to take care of the spectrum boundaries in the transposition process, because it's possible that in a transposition to lower pitch the fundamental region overlaps with the negative frequency fundamental region. And a similar thing can happen with the high harmonics when we are transposing to a higher pitch. This is not yet contemplated in the current implementation of SPP in Daisy. Also, in most of the cases the frequency displacement will be a non integer value and therefore the spectrum will have to be interpolated. As pointed out in [LD99] time-limited interpolation is ideal, but this I highly impractical since it involves the convolution with a long impulse response. SPP technique uses a 3rd order spline interpolation to deal with that.

After frequency shifting the SPP regions, there are a couple of things left to be done. One is to equalize the resulting spectra in order to preserve the original spectral shape; otherwise, the resulting transposition has the undesired smurf-like effect. The other is to correct the phase in order to preserve it.

When one SPP region is shifted in frequency in a frame by frame process, the phase needs to be corrected in order to continue the harmonic quasi-sinusoidal waves. For a constant harmonic i^{th}, the phase increment between two consecutive frames is $\Delta\varphi_i = 2\pi f_i \Delta t$, where the time increment between frames is Δt. If we transpose by a factor *transp*, then the ideal phase increment should be $\Delta\varphi_i = 2\pi f_i{\cdot}transp{\cdot}\Delta t$. Therefore, the amount of phase that should be added to the spectrum phase in order to continue the harmonic i^{th} is

$$\Delta\varphi_i = 2\pi f_i{\cdot}transp{\cdot}\Delta t - 2\pi f_i\Delta t = 2\pi f_i\left(transp - 1\right)\Delta t \tag{2.66}$$

This phase increment is added to all the bins that belong to the i^{th} region, as we can see in figure 2.18. This way we can preserve the local convolution of the window with the harmonic.

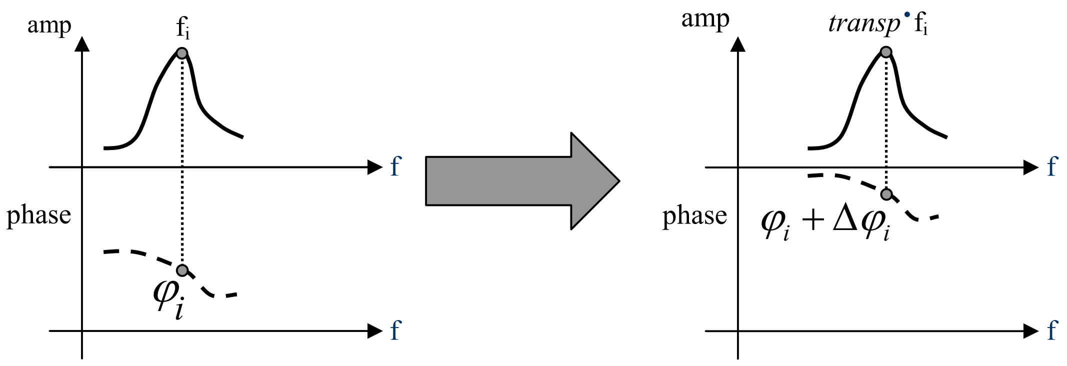


Figure 2.18 Phase shift in SPP transposition

2.3.2.3 Others

Apart from transformations based on transposition and equalization, some other transformations can be applied on top of the SPP technique by taking profit of the different character of the two components that share the SPP region. On one side, the harmonic peak can be somehow labeled as the sinusoidal component of the SPP region. On the other side, the surrounding bins can be related with the noisy component. With such sinusoidal plus residual like decomposition inside the SPP region both elements can be treated separately.

An example of such decomposition is the breathiness effect where, in order to give the voice a noisier and breathy character, the energy of the bins considered being the surroundings of the harmonic peak inside an SPP region is increased by a certain factor while the energy of the bins considered being the harmonic peak bins are lowered.

2.4 Synthesis techniques

2.4.1 SMS Synthesis

2.4.1.1 Deterministic Synthesis

The deterministic component is generated with additive synthesis, similar to the sinusoidal synthesis that was part of the analysis, with the difference that now the phase information is discarded. By not considering phase, this synthesis can either be done in the time domain or in the frequency domain. We will first present the more traditional time domain implementation.

The instantaneous amplitude $\hat{A}(m)$ of a particular partial is obtained by linear interpolation,

$$\hat{A}(m) = \hat{A}^{l-1} + \frac{(\hat{A}^{l} - \hat{A}^{l-1})}{S}m \qquad (2.67)$$

where *m=0,1,...,S-1* is the time sample in the l^{th} frame.

The instantaneous phase is taken to be the integral of the instantaneous frequency, where the instantaneous radian frequency $\hat{\omega}(m)$ is obtained by linear interpolation,

$$\hat{\omega}(m) = \hat{\omega}^{l-1} + \frac{(\hat{\omega}^{l} - \hat{\omega}^{l-1})}{S}m \qquad (2.68)$$

and the instantaneous phase for the r^{th} partial is

$$\hat{\theta}_r(m) = \hat{\theta}_r(l-1) + \hat{\omega}_r(m) \qquad (2.69)$$

Finally, the synthesis equation becomes

$$d^{l}(m) = \sum_{r=1}^{R} \hat{A}_r^{l}(m)\cos\left[\hat{\theta}_r^{l}(m)\right] \qquad (2.70)$$

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

A very efficient implementation of additive synthesis based on the inverse-FFT [RD92, GR94] can be implemented in the frequency domain when the instantaneous phase is not preserved. While this approach looses some of the flexibility of the traditional oscillator bank implementation, especially the instantaneous control of frequency and magnitude, the gain in speed is significant. To generate a sinusoid in the spectral domain it is sufficient to calculate the samples of the main lobe of the window transform, with the appropriate magnitude, frequency and phase values. We can then synthesize as many sinusoids as we want by adding these main lobes in the FFT buffer and performing an IFFT to obtain the resulting time-domain signal. By an overlap-add process we then get the time-varying characteristics of the sound.

The deterministic synthesis frame rate is completely independent of the analysis one. In the implementation using the IFFT we want to have a high frame rate, so that there is no need to interpolate the frequencies and magnitudes inside a frame. As in all short-time based processes we have the problem of having to make a compromise between time and frequency resolution. The window transform should have the fewest possible significant bins since this will be the number of points to generate per sinusoid. A good window choice is the Blackman-Harris 92dB because its main lobe includes most of the energy. However the problem is that such a window does not overlap perfectly to a constant in the time domain. A solution to this problem [RD92] is to undo the effect of the window by dividing by it in the time domain and applying a triangular window before performing the overlap-add process. This will give a good time-frequency compromise.

2.4.1.2 Stochastic Synthesis

The synthesis of the stochastic component can be understood as the filtering of white noise with the spectral envelopes of the stochastic representation. That is performing a time varying filtering of white noise with the envelopes that describe the frequency and amplitude characteristics of the residual. This is generally implemented by the time-domain convolution of white noise with the impulse response of the filter.. But in practice, the easiest and most flexible implementation is to generate the stochastic signal by an inverse-FFT of the spectral envelopes. As in the deterministic synthesis, we can then get the time-varying characteristics of the stochastic signal by an overlap-add process.

Before the inverse-FFT is performed, a complex spectrum (magnitude and phase spectra) has to be obtained from each frequency envelope. We generate the magnitude spectrum by linear interpolating between the spectral envelope points to obtain an envelope curve of length N/2, where N is the FFT-size, and multiplying it by the average magnitude gain that was extracted in the analysis.

There is no phase information in the stochastic representation, but since the phase spectrum of noise is a random signal the phase can be created with a random number generator. To avoid a periodicity at the frame rate, new values are generated at every frame.

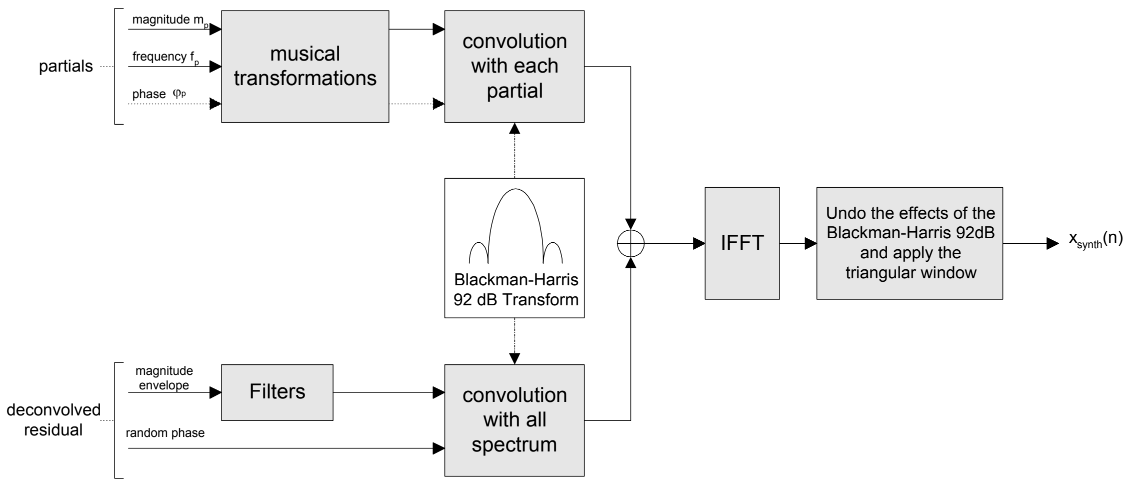

partials
magnitude m_p
frequency f_p
phase φ_p
musical transformations
convolution with each partial
Blackman-Harris 92 dB Transform
IFFT
Undo the effects of the Blackman-Harris 92dB and apply the triangular window
$x_{synth}(n)$
deconvolved residual
magnitude envelope
random phase
Filters
convolution with all spectrum

Figure 2.19 Synthesis block diagram when deterministic and stochastic part share the FFT buffer

By using the IFFT method for both the deterministic and the stochastic synthesis it is possible to use a single IFFT to generate both components as it is shown in figure 2.19.

That is, adding the two spectra in the frequency domain and computing the IFFT once per frame. Since in the noise spectrum there has not been any window applied and in the deterministic synthesis we have used a Blackman-Harris 92dB, we apply this window in the noise spectrum before adding it to the deterministic spectrum and perform the IFFT to the whole spectrum.

2.4.2 SPP Synthesis

The SPP synthesis method is based on the additive synthesis method proposed by [RD92] in which a spectrum buffer is filled and then its inverse FFT is computed. In the SPP case, the spectrum buffer is filled with the transformed harmonic peaks.

The two main transformations processes that take place in the SPP technique are equalization and transposition, which means moving the SPP regions along the amplitude axis and along the frequency axis. When moving a region in any of these two spectrum directions, all the bins in the regions are treated as a block and shifted all in the same way. To exemplify a possible resulting spectrum built with transformed harmonic peaks, we can see figure 2.20 below.

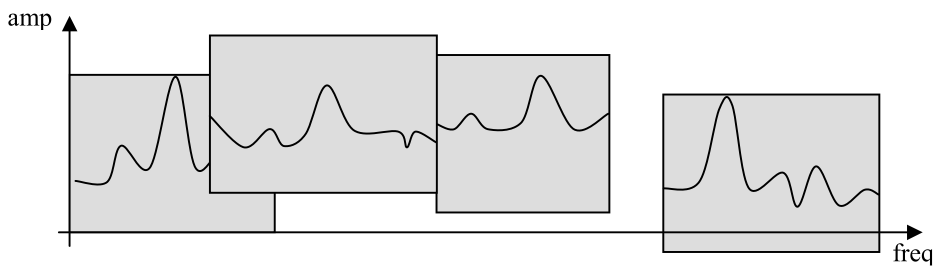


Figure 2.20 SPP transformed spectra example

If no transformation has been applied, the synthesis spectrum will be exactly the same as in the analysis since SPP regions will be on its original position and no overlap, no gaps, and no discontinuities will appear in the synthesis spectrum buffer. If transformations have been applied, the SPP regions are placed on its corresponding new amplitude and frequency positions and the synthesis spectrum buffer is filled by summing their bins in complex. As in the SMS sinusoidal part, the IFFT is applied and the overlap-add gives out the resulting time domain varying signal.

It is important to notice the synthesis buffer spectrum may not be showing a real spectrum since a real spectrum can not show discontinuities or gaps as the ones shown in the example illustrated in figure 2.20. The spectrum of a time-domain real signal can not have such discontinuities. Imagine we wanted to synthesize a simple sinusoid. What we would do is fill the spectrum buffer only with the main lobe bins of the window as shown in the figure 2.21(a).

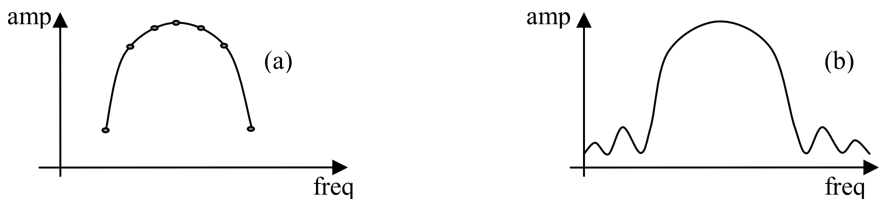


Figure 2.21 The spectrum of a single harmonic spectral peak in (a) the IFFT buffer and (b) the synthesized sound spectral view

Such frequency finite signal corresponds to an infinite signal in time domain. Since we are not preserving this time-infiniteness because we are applying an IFFT and we are windowing to overlap, we are not preserving the finiteness of the signal in the spectrum domain. That's why the real spectrum of the synthesized sound, see figure 2.21(b) would look like the figure 2.21(b) above.

Chapter 3

Voice Modeling and Synthesis

The aim of this chapter is to explain, once the spectral processing techniques have been introduced, our approach to voice modeling and singing voice synthesis problems. Although the singing voice model and synthesizer issues are already described in [BL03, BLMK03, BCLOS01, BLCS01], they are introduced here in order to have a coherent general view of the work.

3.1 Introduction to Voice Production Acoustic Theory

The voice is typically explained as a source / filter system, in which an excitation is filtered by the vocal tract resonances. The voice excitation can be rather voiced, unvoiced, or a combination of both. The unvoiced excitation corresponds to the turbulent airflow that arises from the lungs and the voiced excitation corresponds to the glottal pulses that originate the vocal fold vibrations. The voice filter is characterized by a set of resonances called formants that have their origin in the voice organs lengths and shapes (trachea, esophagus, larynx, pharyngeal cavity, velum, hard palate, jaw, nasal cavity, nostril, lip, tongue, teeth, and oral cavity). This filter modulates the timbre of the sound by dynamically moving the voice organs and thus changing the amplitude, center frequencies and bandwidths of the formants.

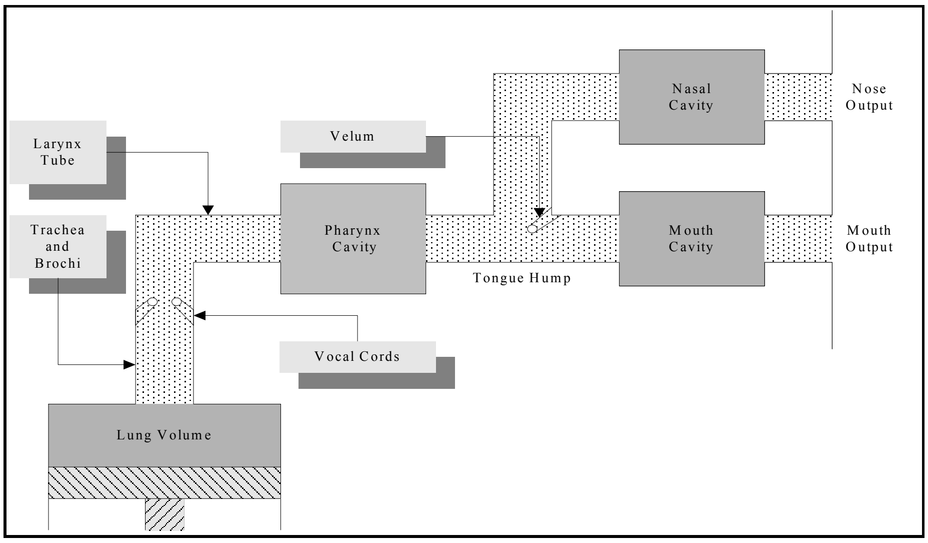


Figure 3.1 Schematic representation of the speech production system after Flanagan [Fla72]

3.1.1 The Speech Production Mechanism

Both speech and singing voice sounds are generated from the same production mechanism. A simplified representation of the physiological speech production mechanism is shown in figure 1.1. Air enters the lungs via the normal breathing mechanism and when pushed out this air in the lungs excites the vocal mechanism. The muscle force pushes air out of the lungs and through the bronchi and trachea. When the vocal folds are tensed, the airflow causes them to vibrate, producing so-called voiced speech sound. In this case the airflow is chopped by the vocal folds into quasi-periodic pulses called phonation.

Figure 1.2 shows a transversal view of the throat including the vocal folds in the opened and closed state. The resulting air-pressure function is shown in figure 1.4 (a) for a vibration rate of f_0=30 Hz and in figure 1.5 (a) for f_0=100 Hz. This rate at which the vocal folds open and close determines the harmonic frequencies of the spectrum *S(f)* of the glottal airflow and thus the perceived fundamental frequency f_0 of the resulting speech sound.

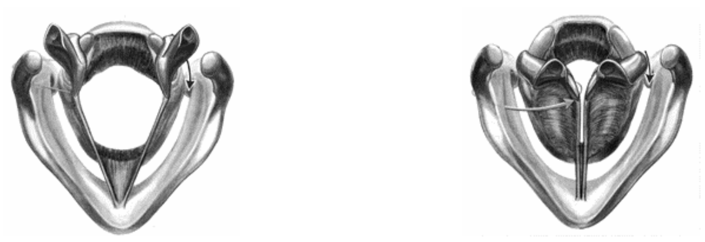


Figure 3.2 Transversal view of the throat with vocal folds opened (a) and closed (b) [h1]

Two idealized source spectra of the glottal excitation are shown in figure 1.3. In this example the fundamental frequencies are f_0=100 Hz (a) and f_0=400 Hz (b). The magnitude of the harmonics gradually falls. For normal voicing this decrease is about -12 dB / octave [Kla80]. When the vocal folds are relaxed, in order to produce a noise, the airflow either must pass through a constriction in the vocal tract and thereby become turbulent, producing so-called unvoiced sound, or it can build up pressure behind a point of total closure within the vocal tract (e.g. the lips), and when the closure is opened, the pressure is suddenly released, causing a brief transient sound. The spectrum of such a turbulent noise source is stochastic with a slight magnitude decrease of -6 dB / octave [Kla80]. An example of this kind of source spectrum is shown in figure3.6.

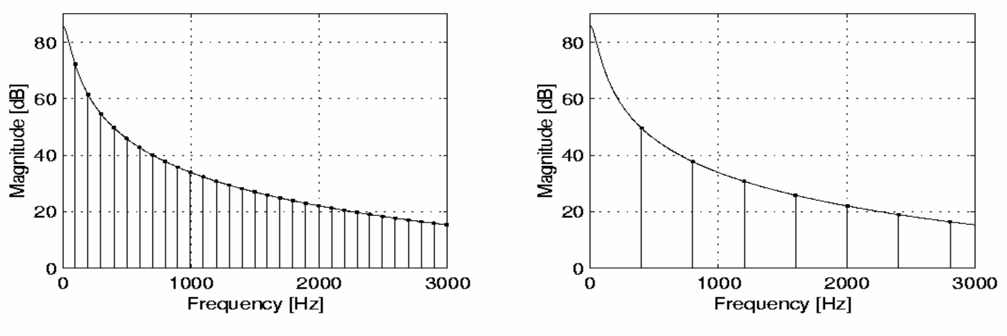


Figure 3.3 Glottal excitation *S(f)* with f_0=100 Hz (a) and f_0=400 Hz (b)

The produced source spectra *S(f)*, regardless it is voiced and unvoiced, is then modulated in amplitude in passing through the pharynx (the throat cavity), the mouth cavity, and possibly the nasal cavity. Depending on the positions of the various articulators (i.e., jaw, tongue, velum, lips, mouth), different sounds are produced. This modification originated by the different cavities and articulators is called supralaryngeal filter [LB88] because it can be modeled as a linear system with a transfer function *T(f)*, modifying the source spectrum *S(f)*.

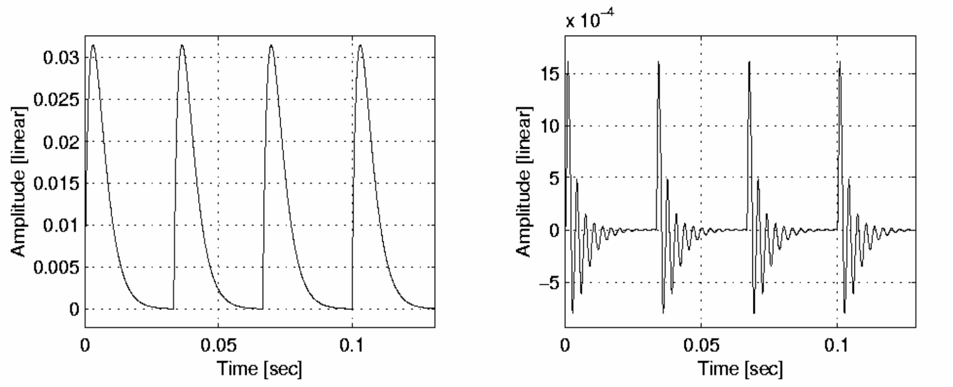


Figure3.4 Air-pressure function for a glottal excitation of f_0=30 Hz before (a) and after passing the supralaryngeal filter (b)

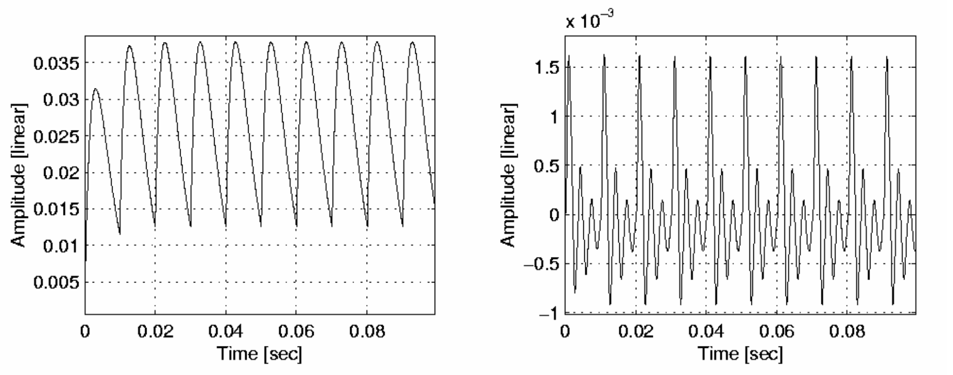


Figure3.5 Air-pressure function for a glottal excitation of f_0=100 Hz before (a) and after passing the supralaryngeal filter (b)

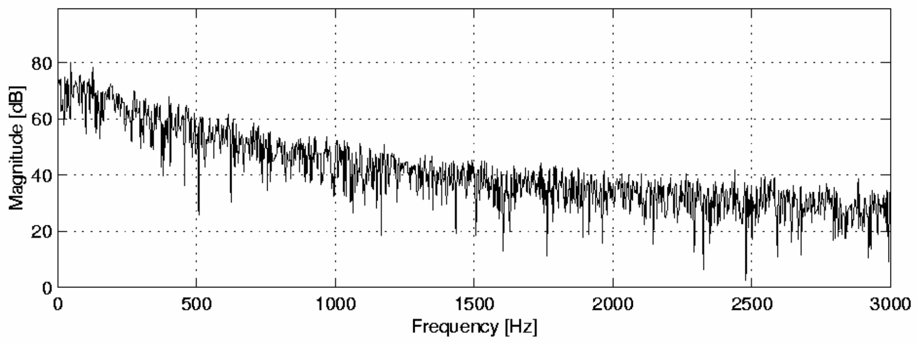


Figure3.6 Unvoiced excitation

The modified spectrum *U(f)* can be written as

$$U(f) = S(f) \cdot T(f) \tag{3.1}$$

During the production of human speech the shape of the supralaryngeal airway and thus the transfer function *T(f)* changes continually. This acoustical filter suppresses the transfer of sound energy at certain frequencies and lets maximum energy through at other frequencies. In speech processing the frequencies at which local energy maximum may pass through the supralaryngeal airway are called formant frequencies F_i. They are determined by the length (speaker dependent) and the shape of the vocal tract (more articulation than speaker dependent). The larynx including the vocal folds and the subglottal system has only minor effects on the formant frequencies F_i [Fla72]. Different vowels owe their phonetic difference, or so-called quality, to their different formant frequencies.

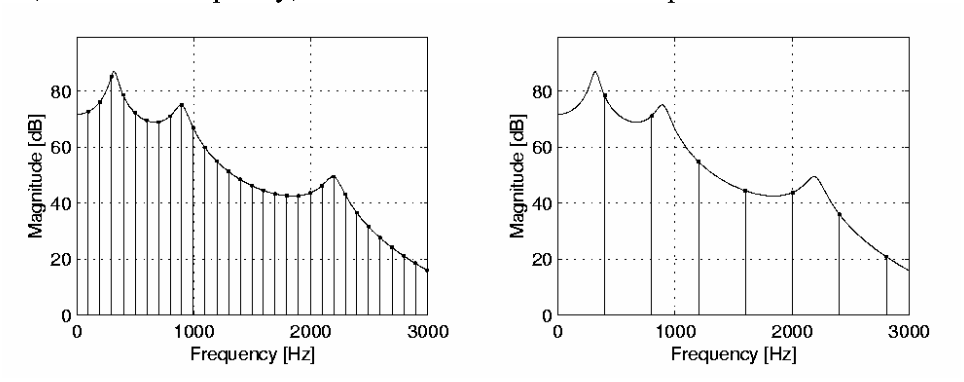


Figure 3.7 Idealized transfer function *T(f)* of the vocal tract for the phoneme [u] with f_0=100 Hz (a) and f_0=400 Hz (b)

As an example, figure 1.7 shows an idealized transfer function of the supralaryngeal airway for the vowel [u] (the phoneme in "shoe``) including a grid of the harmonic frequencies for f_0=100 Hz (a) and f_0=400 Hz (b). The first three formant frequencies of this vowel sound are F_1=500 Hz, F_2=900 Hz, and F_3=2200 Hz. The bandwidth of each formant varies from 60 to 150 Hz. The first three formants of vowels play a major role in specifying these sounds. Higher formants exist, but they are not necessary for the perception of vowel qualities [Fla72].

The resulting spectrum $U(f) = S(f) \cdot T(f)$ describes the speech signal at the end of the airways of the vocal tract. Sound energy is present at each of the harmonics of the glottal source, but the amplitude of each harmonic will be a function of *T(f)* and *S(f)*. Finally, the spectrum *U(f)* is modified by the radiation characteristic *R(f)* in a way that the resulting spectrum can be written as

$$P(f) = S(f) \cdot T(f) \cdot R(f) \tag{3.2}$$

with the frequency *f* in Hz. The radiation characteristic *R(f)* models the effect of the directivity patterns of sound radiation from the head as a function of frequency. This effect depends on the lip opening and can be modeled as a high-pass filter shown in figure 1.8 [Kla80]. The resulting spectra *P(f)* of the previous examples of f_0=100 Hz (a) and f_0=400 Hz (b) are shown in figure 1.9.

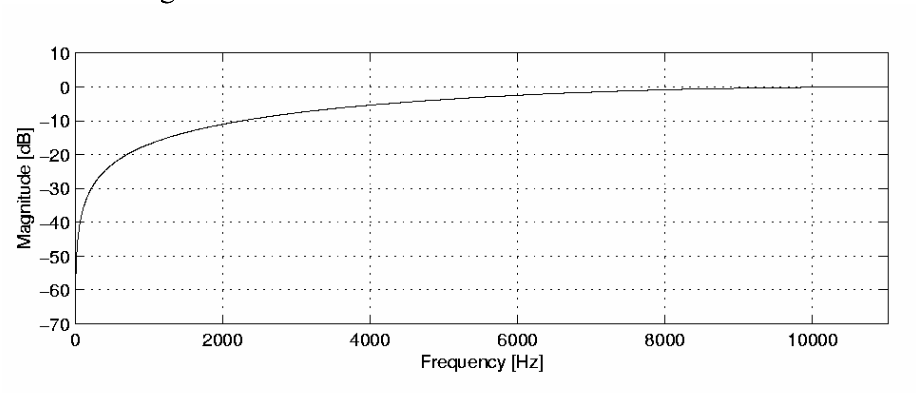


Figure 3.8 Idealized radiation characteristic *R(f)* of the mouth

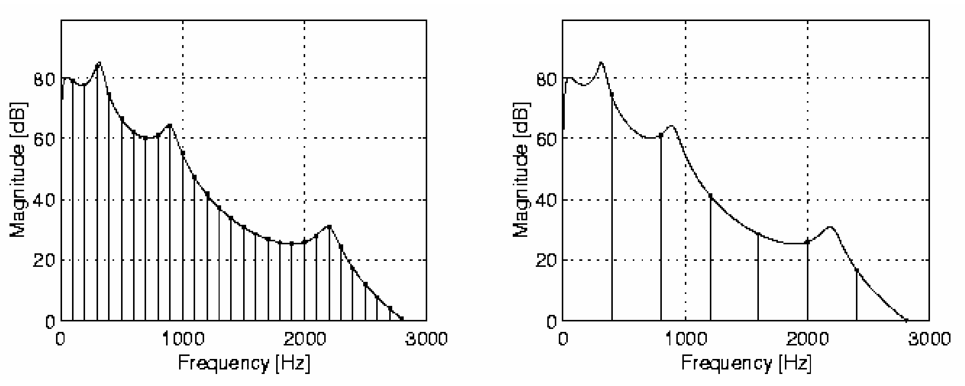


Figure 3.9 Idealized spectrum *P(f)* for the phoneme [u] with f_0=100 Hz (a) and f_0=400 Hz (b)

3.1.2 Vowels and Formants

The most important class of voiced speech sounds is vowels.

Vowels are produced by exciting an essentially fixed vocal tract shape with quasi-periodic pulses of air, caused by the vibration of the vocal folds (glottal excitation). The vocal tract has to be open enough for the pulsed air to flow without meeting any obstacle. The vocal tract shape modifies the glottal excitation and thus the vocal timbre in a manner that the appropriate vowel is perceived. If the vocal tract narrows or even temporarily closes, the airflow gives birth to a noise, so that a consonant is produced.

The spectral qualities of different vowels are mainly caused by the glottal excitation (speaker dependent) and the vocal tract transfer function (depending on the type of vowel, the language, and the speaker). As introduced in section 3.1 the vocal tract transfer function can be characterized by formants, which are resonant peaks in the spectrum.

According to Sundberg [Sun87] the higher the formant frequency, the more its frequency depends on nonarticulatory factors, such as vocal tract length and the vocal tract dimension within and around the larynx tube. Good singers position and move the length and the shape of the vocal tract in a very precise way. These variations make the formants have different amplitudes and frequencies, which we perceive as different voiced sounds.

The length of the vocal tract defined as the distance from the glottis to the lips can be modified up to a certain extent. The larynx can be raised or lowered increasing or decreasing respectively the length. Also the posture of the lips can cause fluctuations to the length of the vocal tract, if we protrude the lips we lengthen it and if we smile we reduce it. The longer the vocal tract, the lower the formant frequencies are. The shape of the vocal tract, on the other hand, is modified by what is referred as articulators. These articulators are the tools we can use to change the area function of the vocal tract and they can move in different ways and each of the movements has a different effect. A movement of any of the articulators generally affects the frequencies of all formants. Sundberg experiments have shown that the first three formants define the vowel type and differ less between different speakers, for example, between a male and a female, than the higher formants. The fourth and the fifth formant are relevant for perceiving the voice timbre, which is the personal component of a voice sound.

The effects of different articulators (e.g. the tongue) and speaker dependent physiological characteristics on the different formants can be summarized as:

First formant: sensitive to varying the jaw opening.
Second formant: sensitive to changes of the tongue shape.
Third formant: sensitive to the position of the tongue, particularly to the size of the cavity that is just behind the front teeth.
Fourth formant: depends on the vocal tract dimensions within and around the larynx tube. Thus, the speaker has no explicit control of this formant frequency.
Fourth and fifth formant: depend on the length of the vocal tract and thus are speaker dependent.

A more detailed description of the dependency of formants on articulators or on physiological characteristics can be found in [Sun87, LB88].

The representation of the spectral shape of a singing voice depends highly on the fundamental frequency f_0. If the glottal excitation is low pitched the idealized spectrum *P(f)* can be easily observed in the spectral envelope of the trajectories. But if the voiced source has a very high fundamental frequency f_0 and the harmonic frequencies of the source spectrum *S(f)* are thus very wide spaced, then *P(f)* and the spectral envelope of the trajectories differ significantly. Figure 3.10 below illustrate such problem.

The trajectories show only *P(f)* at special frequencies f_i. In the case of an ideal analysis, f_i can be written as

$$f_i = i \cdot f_0 \qquad \text{i=1,2,3, …} \tag{3.3}$$

and thus the magnitudes *M(f$_i$)* of the trajectories are equidistant samples of the spectrum *P(f)*. It can occur that the voiced sound is too high pitched so that almost no similarity to the continuous spectrum *P(f)* can be found in the discrete spectral envelope of the sinusoidal trajectories. All these considerations show that visual formant detection from the spectral shape will only be possible when dealing with low pitched voices (approximately under 100Hz).

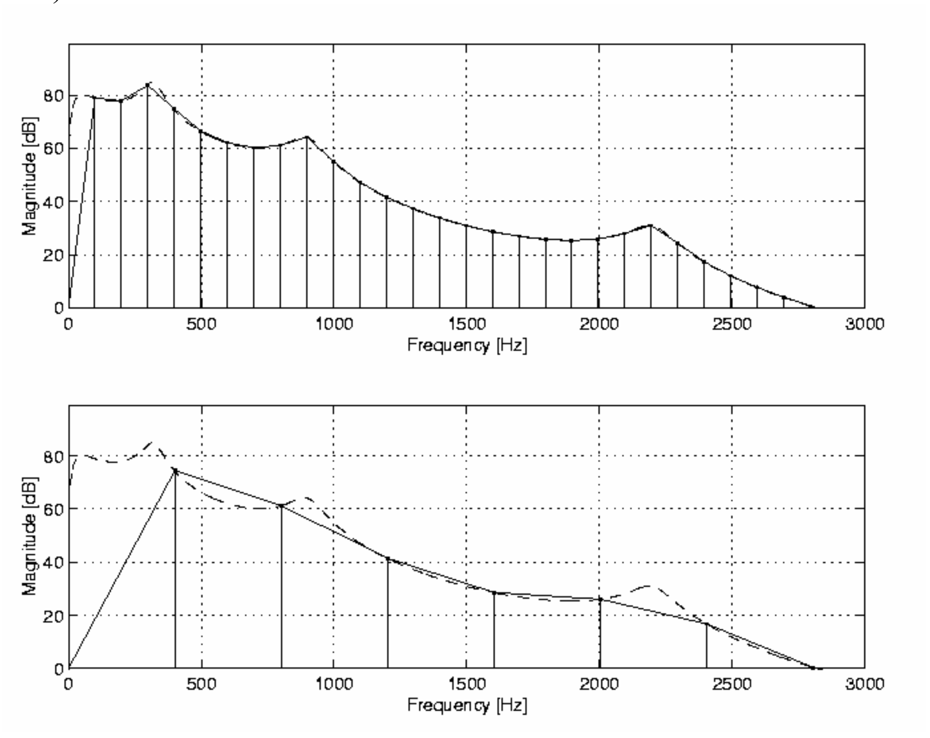


Figure 3.10 Idealized spectrum *P(f)* of the phoneme [u] with f_0= 100Hz (a) and f_0=400 Hz (b) including the spectral envelope of the deterministic trajectories

3.1.3 Speech and Singing Voice Particularities

Although speech and singing voice sounds have many properties in common because they originate from the same production physiology, there are some important differences that make them different. In singing, the intelligibility of the phonemic message is often secondary to the intonation and musical qualities of the voice. Vowels are often sustained much longer in singing than in speech and independent control of pitch and loudness over a large range is required. A trained singer can sing an entire sequence of vowel sounds at the same pitch. These major differences are highlighted in the following brief list, which is not meant to lay claim to completeness but rather to point out some important differences between speech and singing voice processing:

Voiced/unvoiced ratio: The ratio between voiced sounds, unvoiced sounds, and silence is about 60%, 25\%, and 15\% respectively in the case of normal speech. In singing, the percentage of phonation time can increase up to 95\% in the case of opera music. This makes voiced sounds special important in singing because they carry most of the musical information.

Vibrato: Two types of vibrato exist in singing, which differ in their production mechanism. The classical vibrato in opera music corresponds to periodic modulation of the phonation frequency. The regularity of this modulation is considered as a sign of the singer's vocal skill. It is characterized by two parameters, namely the rate and the extent of the modulation of the phonation frequency. In popular music, the vibrato is generated by variations in the subglottal pressure, implying an amplitude modulation of the voice source [Sun87]. In speech, no vibrato exists. The Vibrato adds certain naturalness to the singing voice and it is a very specific characteristic of the singing voice of a singer.

Dynamic: The dynamic range as well as the average loudness is greater in singing than in speech. The spectral characteristics of a voiced sound change with the loudness [Sun87].

Singer's formant: This is a phenomenon that can be observed especially in the singing of male opera singers [Sun87]. The singer's formant is generated by clustering of the third, fourth, and fifth formant. The frequency separation of these formants is smaller in the sung than in the spoken vowel, resulting in one peak at approximately 2-3 kHz with a great magnitude value in the spectrum. A trained singer has the ability to move the formant frequencies in a manner in which already existing formants in speech occurring only in higher frequency regions are tuned down. As a result, the magnitude of the down-tuned formants increases towards lower frequencies due to the voice source spectrum.

Modification of vowels: Singing at a very high pitch includes that the fundamental frequency f_0 is greater than the one of the first formant. In order not to loose this characteristic resonance and thus important acoustic energy, trained singers move about the first formant in frequency to the phonation frequency. The gain of loudness increases as soon as the frequency of the first formant joins the fundamental f_0. Although the perceived vowel is slightly modified, an important ability to use a greater dynamically range is gained and thus it is a gain of musical expression.

Fundamental frequency: In speech, the fundamental frequency variations express an emotional state of the speaker or add intelligibility to the spoken words. This is called prosody and distinguishes, for example, a question from a normal statement. The frequency range of f_0 is very small compared to singing. Considering a trained singer, the minimum range is about two octaves and for excellent singers like Maria Callas it can be up to three octaves.

3.2 The Excitation plus Residual (EpR) Model

Our Voice Model is based on an extension of the well known source/filter approach [Chil94] we call EpR (Excitation plus Resonances). The EpR filter can be decomposed in two cascade filters. The first of them models the differentiated glottal pulse frequency response and the second the vocal tract (resonance filter).

3.2.1 The EpR Source filter

The EpR source is modeled as a frequency domain curve and one source resonance. The curve is defined by a gain and an exponential decay as follows:

$$Source_{dB} = Gain_{dB} + SlopeDepth_{dB}\left(e^{Slope\cdot f} - 1\right) \tag{3.4}$$

It is obtained from an approximation to the harmonic spectral shape (*HSS*) determined by the harmonics identified in the SMS analysis

$$HSS\left(f\right) = envelope_{i=0..n-1}\left[f_i, 20\log\left(a_i\right)\right] \tag{3.5}$$

where i is the index of the harmonic, n is the number of harmonics, f_i and a_i are the frequency and amplitude of the i^{th} harmonic. On top of the curve, we add a second resonance in order to model the low frequency content of the spectrum below the first formant.

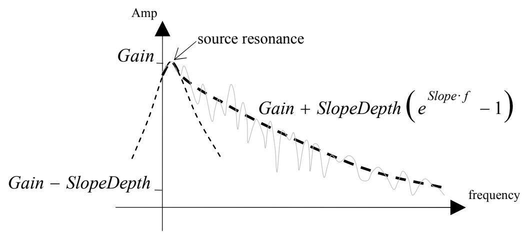


Figure 3.11 The EpR source resonance

The source resonance is modeled as a symmetric second order filter (based on the Klatt formant synthesizer [Kla80]) with center frequency F, bandwidth Bw and linear amplitude Amp. The transfer function of the resonance $R(f)$ can be expressed as follows

$$\begin{aligned} H(z) &= \frac{A}{1 - Bz^{-1} - Cz^{-2}} \\ R\left(f\right) &= Amp\frac{H\left(e^{j2\pi\left(0.5+\frac{f-F}{fs}\right)}\right)}{H\left(e^{j\pi}\right)} \end{aligned} \tag{3.6}$$

where

$$\begin{aligned} fs &= \text{Sampling rate} \\ C &= -e^{-\frac{2\pi Bw}{fs}} \\ B &= 2\cos(\pi)e^{-\frac{\pi Bw}{fs}} \\ A &= 1 - B - C \end{aligned}$$

The amplitude parameter (*Amp*) is relative to the source curve. Notice that in equation (3.6) the resonance amplitude shape is always symmetrical respect to its center frequency.

3.2.2 The EpR vocal tract filter

The vocal tract is modeled by a vector of resonances plus a differential spectral shape envelope. It can be understood as an approximation to the vocal tract filter. These filter resonances are modeled in the same way as the source resonance, see equation (3.6), where the lower frequency resonances are somewhat equivalent to the vocal tract formants.

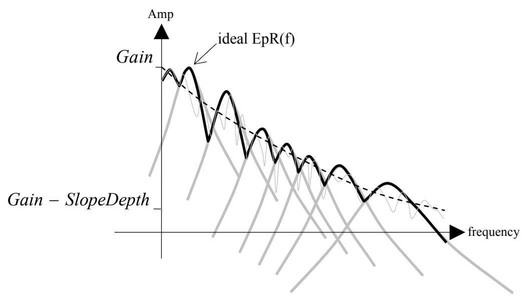


Figure3.12 The EpR filter resonances

The differential spectral shape envelope actually stores the differences (in dB) between the ideal EpR model and the real harmonic spectral shape (*HSS*) of a singer's performance. We calculate it as a 30 Hz equidistant step envelope.

$$DSS\left(f\right) = envelope_{i=0..}\left[30i, HSS_{dB}(30i) - iEpR_{dB}(30i)\right] \tag{3.7}$$

Thus, the original singer's spectrum can be obtained if no transformations are applied to the EpR model.

3.2.3 The EpR phase alignment

The phase alignment of the harmonics at the beginning of each period is obtained from the EpR spectral phase envelope. A time shift is applied just before the synthesis, in order to get the actual phase envelope at the synthesis time (usually it will not match the beginning of the period). This phase alignment is then added to the voiced harmonic excitation spectrum phase envelope.

The EpR spectral phase model assumes that each filter resonance (not the source resonance) produces a linear shift of π on the flat phase envelope with a bandwidth depending on the estimated resonance bandwidth. Although this value has been obtained empirically, the symmetric second order resonance model itself has a phase shift under the resonance peak. The value of this phase shift depends on the center frequency and the bandwidth of the resonance. This phase model is especially important in order to get more intelligible sounds and more natural low pitch male voices.

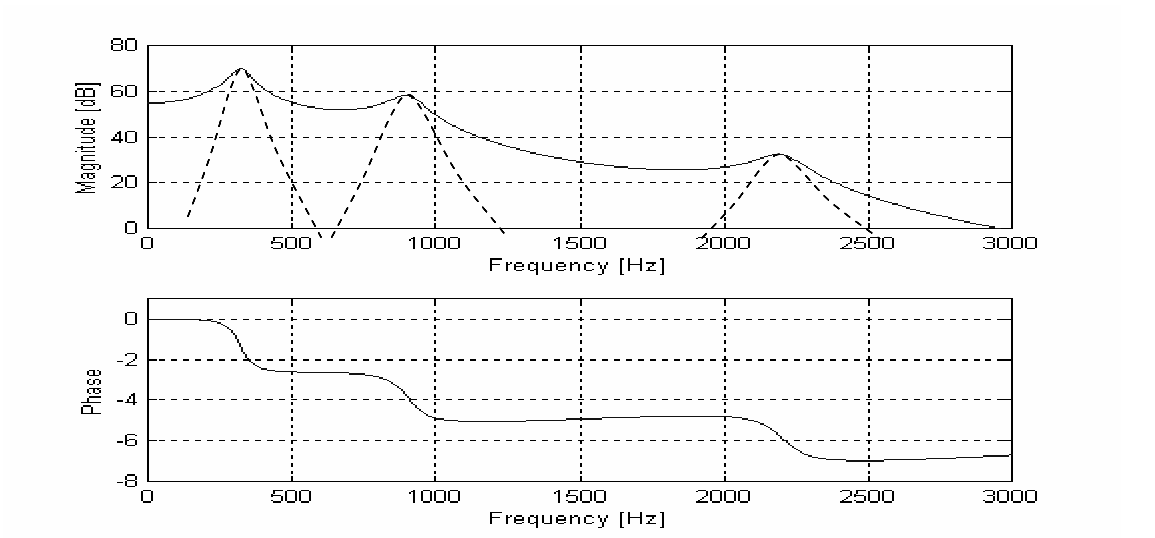


Figure 3.13 The phase alignment is approximated as a linear segment, with a phase shift for each resonance

3.3 Singing Voice Synthesis

The synthesis of voice has been approached form many different directions and though it is a slack categorization, we can split synthesis models into two groups: spectral models, which are based on the perception of the sound, and physical models, which are based on the production of the sound.

Both spectral and physical models have their own benefits and drawbacks [Cook98]. Physical models such as acoustic tube models uses the control parameters humans use to control their own vocal system and can generate time-varying behaviors by the model itself. On the other hand some parameters might not have intuitive significance and might interact in non-obvious ways. Spectral models such as those based on FM, FOFs, phase and channel vocoders or sinusoidal tracks have precise analysis / synthesis methods and work close to the human perception but their parameterization is not that suitable for study, manipulation, or composition since they don't match any auditory system structure.

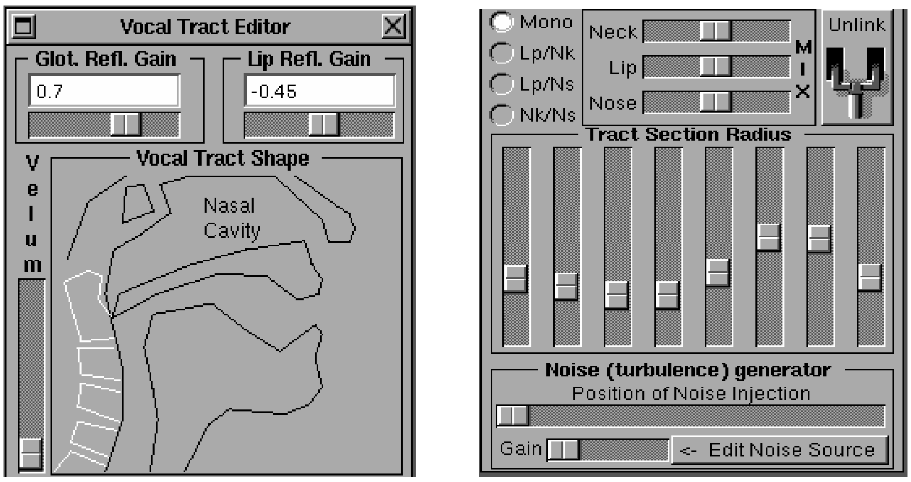
Models such as the one used in formant synthesizers are considered to be pseudo-physical models because even though these are mainly spectral models they make use of the source / filter decomposition. The singing synthesizer we present would be placed in this model group.

3.3.1 Applications

It was in 1961 when Mathews's Daisy first synthetic singing voice was heard in the Bell Laboratories [Puck01]. Since then, the computer has become a more virtuosic singer and nowadays quite a few singing voice systems use different techniques [Rod02,Cook96] to create your own synthetic voice lines. Not all of them, but maybe the most popular ones are listed below.

Chant
Chant is a singing synthesizer developed at IRCAM by Xavier Rodet and Yves Potard. A review of the system can be found in [Rod85]. Chant uses an excitation resonance model. For each resonance, a basic response simulated with Formant Wave Functions (FOF) [Rod84], is associated. Chant produces the resulting sound by adding the FOF corresponding to each formant for a given pseudo-periodic source.
Chant web page: http://www.ircam.fr/produits/technologies/chant-e.html

SPASM
SPASM is a physical model based singing synthesizer that uses a waveguide articulatory vocal tract model. The synthesizer was developed by Perry Cook at Stanford University and is described in [Cook91, Cook92].
SPASM web page: http://www.cs.princeton.edu/~prc/SingingSynth.html



Glot. Refl. Gain 0.7
Lip Refl. Gain -0.45

Figure 3.14 Perry Cook's SPASM graphical interface [h2]

Lyricos
Lyricos is a singing synthesizer based on Analysis-by-Synthesis, Overlap-Add sinusoidal model (ABSOLA) [GS92]. A general overview of the system can be found at Macon's work [MJOCG97].
The Lyricos project is closed but it has a descendent improved system called Flinger based on Macon's singing synthesizer and on the Festival Speech Synthesis System, a general multi-lingual speech synthesis developed at the Centre for Speech Technology Research of the University of Edinburgh.
Lyricos web page: http://cslu.cse.ogi.edu/tts/demos/index.html#sing
Flinger web page http://cslu.cse.ogi.edu/tts/flinger/

VocalWriter
Vocal Writer is a shareware singing synthesizer software for Macintosh. The interface looks like a typical MIDI sequencer interface with some functions added to edit the lyrics and some vocal controls such as brightness, vibrato, breath and others.

Vocal Writer web page: http://www.mccormicksnet.com/vocalwrt.htm

Harmony's Virtual Singer
Harmony-s virtual singer is an opera-like singing synthesizer from Myriad Software. The main differences from other synthesizer are the wide amount of languages in which the system can sing, and the RealSinger capability, which allows defining a Virtual Singer voice out of recordings of the user's own voice.
http://www.myriad-online.com/vs.htm

Vocaloid
Vocaloid has been released by YAMAHA and it is based on the research carried out in the MTG's voice processing group under Daisy's project. Some of their bases are described in next section.
Vocaloid web page: http://www.global.yamaha.com/news/20030304b.html

3.3.2 Our system

Our system is a source-filter type singing synthesizer application based on the concatenation of samples and built on top of the SPP technique and the EpR model. The system generates a performance of an artificial singer out of the musical score and the phonetic transcription of a song. In the figure 3.15 we can see the general diagram of our singing voice synthesizer. The inputs of the system are the lyrics (the phonetic transcription), the melody to be sung and optionally some expression parameters. On the other hand, the system is also feed by a singer database. This database holds the voice characteristics and is created from analysis of recordings of a real singer

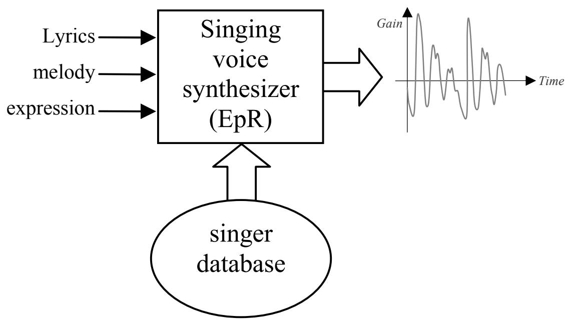


Figure 3.15 Our singing voice synthesizer general diagram

Since the system is a sample-based synthesizer, it synthesizes the virtual performance by concatenating the elemental synthesis units that are stored in the database. Although the database stores timbres, steady states, phonetic articulations, attacks, releases, vibratos, and note to note transitions, two main categories classify these elemental units: steady states and articulations. If these units (steady states and articulations) are concatenated among them as they are, spectral shape and phase discontinuities will appear. To avoid it, the discontinuities are spread out along a set of transition frames that surround the joint frame.

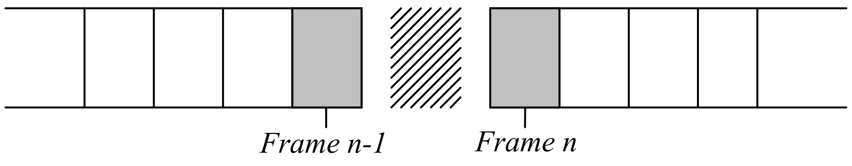


Figure 3.16 Segment concatenation

The steady state segments are synthesized using both transposition and equalization SPP transformations. The main input of the steady state synthesis engine is the SPP steady state template that is compound of the SPP analysis of a steady state sung by the database singer (harmonic peaks, STFT spectrums, SPP region marks). These spectrums are transposed to the desired pitch and equalized to the desired timbre. The desired stationary timbre is defined by the *timbre template EpR* plus some resonance and excitation variations taken from the EpR estimation of a long steady state (*EpR stationary variation template*). The *EpR stationary variation template* is only used for pitch and resonance variations. These processes are represented in figure 3.17.

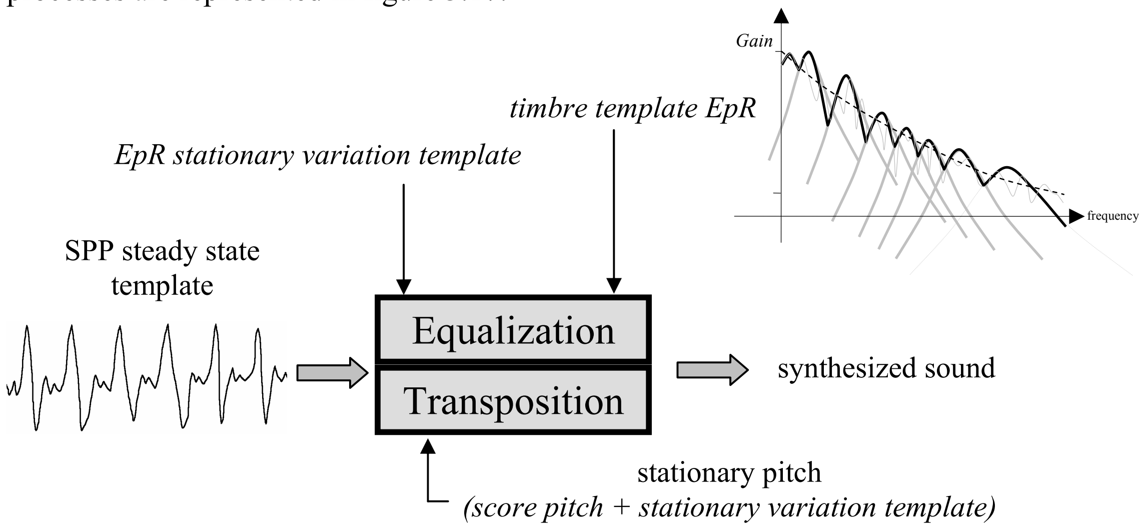

Gain
frequency
timbre template EpR
EpR stationary variation template
SPP steady state template
Equalization
Transposition
synthesized sound
stationary pitch
(score pitch + stationary variation template)

Figure 3.17 SPP steady state synthesis engine

The articulation segments are synthesized using both transposition and equalization SPP transformations as well. The main input of the articulation synthesis engine is the *SPP Articulation* (SPP analysis of a certain phonetic articulation sung by the database singer). The *SPP articulation* spectrums are processed trough the transposition and equalization modules. The target timbre in the equalization process is obtained from the 3rd order spline interpolation of the harmonics peaks found in the *SPP articulation* frame. Therefore, the transposition and equalization together can be considered like a transposition with timbre preservation process. Both equalization and transposition modules are only applied to the voiced regions. The unvoiced frames are synthesized without any transformation. In general, the articulation duration specified in the score will not match the articulation duration in the database and this will force us to apply some time-scaling technique. In the current implementation we have adapted the time-stretching algorithm described in

[Bon00] to the specific requirements of monophonic harmonic solo signals. Figure 3.18 illustrates the articulation synthesis engine.

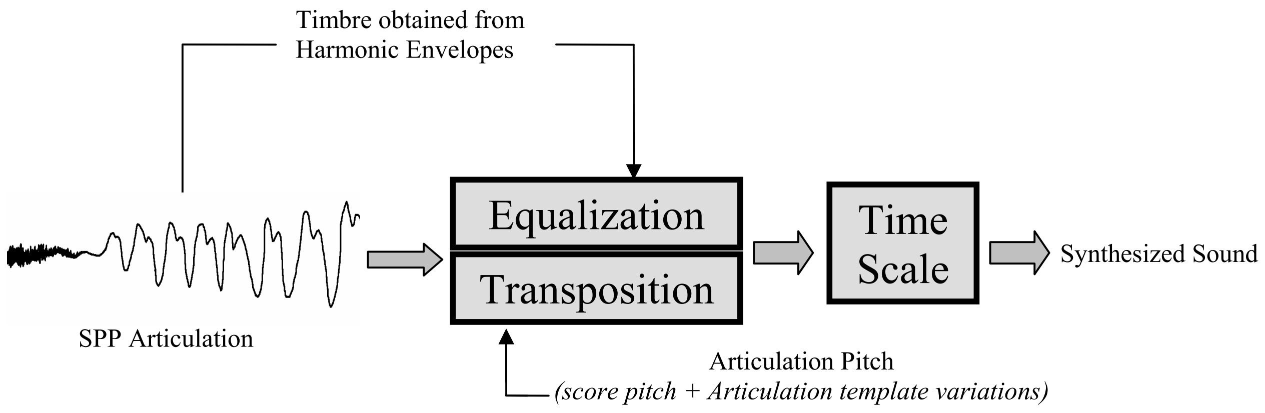


Figure 3.18 SPP articulation synthesis engine

Chapter 4

Research Contributions

Since all publications have been written in collaboration with some of my colleagues from the Music Technology Group and YAMAHA Japan, I will try to point out clearly which has been my contribution to the final work for each of them.

1 - Loscos, A., Resina, E.
'SMSPerformer: A real-time synthesis interface for SMS'
Proceedings of COST G6 Conference on Digital Audio Effects
Barcelona 1998

Abstract: SmsPerformer is a graphical interface for the real-time SMS synthesis engine. The application works from analyzed sounds and it has been designed to be used both as a composition and a performance tool. The program includes programmable time-varying transformations, MIDI control for the synthesis parameters, and performance loading and saving options.

My contribution to the work presented in this paper was mainly the implementation of the initial prototype. This implied designing a skeleton of the GUI, resolving the sound I/O issues, and porting the SMS transformation and synthesis engine to the SMSPerformer requirements. All the composition oriented features and the final design of the interface were implemented in close collaboration with the prestigious guitarist and composer Eduard Resina.

2 - Loscos, A., Cano, P., Bonada, J.
'Low-Delay Singing Voice Alignment to Text'
Proceedings of International Computer Music Conference
Beijing 1999

Abstract: In this paper we present some ideas and preliminary results on how to move phoneme recognition techniques from speech to the singing voice to solve the low-delay alignment problem. The work focus mainly on searching the most appropriate Hidden Markov Model (HMM) architecture and suitable input features for the singing voice, and reducing the delay of the phonetic aligner without reducing its accuracy.

My contribution to the work presented in this paper was to design an HMM architecture suitable for the requirements of the automatic singing voice impersonator. All the research and implementation was carried out in close collaboration with the co-writers.

3 - Cano, P., Loscos, A., Bonada, J.
'Score-Performance Matching using HMMs'
Proceedings of International Computer Music Conference
Beijing 1999

Abstract: In this paper we will describe an implementation of a score-performance matching, capable of score following, based on a stochastic approach using Hidden Markov Models.

My contribution to the work presented in this paper was to conceive a way to design an HMM note model in which the tuning was somehow included. All the research presented in this paper was carried out in close collaboration with the co-writers.

4 - Cano, P., Loscos, A., Bonada, J., de Boer, M., Serra, X
'Voice Morphing System for Impersonating in Karaoke Applications'
Proceedings of International Computer Music Conference
Berlin 2000

Abstract: In this paper we present a real-time system for morphing two voices in the context of a karaoke application. As the user sings a pre-established song, his pitch, timbre, vibrato and articulation can be modified to resemble those of a pre-recorded and pre-analyzed recording of the same melody sang by another person. The underlying analysis/synthesis technique is based on SMS, to which many changes have been done to better adapt it to the singing voice and the real-time constrains of the system. Also a recognition and alignment module has been added for the needed synchronization of the user's voice with the target's voice before the morph is done. There is room for improvements in every single module of the system, but the techniques presented have proved to be valid and capable of musically useful results.

The aim of this paper was to present the Elvis prototype to the scientific community. Therefore the paper was presenting a general view of all the research and implementation involved in real-time singing voice impersonator. My contribution was especially relevant in the improvement of the SMS analysis-morph-synthesis procedures, and in the design of the alignment process.

5 - de Boer, M., Bonada, J., Cano, P., Loscos, A., Serra, X.
'Singing Voice Impersonator Application for PC'
Proceedings of International Computer Music Conference
Berlin 2000

Abstract: This paper presents the implementation aspects of a real-time system for morphing two voices in the context of a karaoke application. It describes the software design and implementation under a PC platform and it discusses platform specific issues to attain the minimum system delay.

The aim of this paper was to give a more detailed explanation of the implementation of the Elvis prototype. My contribution in this work was collaborating in the design and testing of the prototype.

6 - Bonada, J., Loscos, A., Cano, P., Serra, X.
'Spectral Approach to the Modeling of the Singing Voice'
Proceedings of 111th AES Convention
New York 2001

Abstract: In this paper we will present an adaptation of the SMS (Spectral Modeling Synthesis) model for the case of the singing voice. SMS is a synthesis by analysis technique based on the decomposition of the sound into sinusoidal and residual components from which high-level spectral features can be extracted. We will detail how the original SMS model has been expanded due to the requirements of an impersonating applications and a voice synthesizer. The impersonating application can be described as a real-time system for morphing two voices in the context of a karaoke application. The singing synthesis application we have developed generates a performance of an artificial singer out of the musical score and the phonetic transcription of a song. These two applications have been implemented as software to run on the PC platform and can be used to illustrate the results of all the modifications done to the initial SMS spectral model for the singing voice case.

The aim of this paper was to present our works on singing voice to the AES scientific community. My contribution in the work presented in this paper was, besides the research and implementation carried out in the impersonator, all my inputs to the EpR voice model design.

7 - Bonada, J., Celma, O., Loscos, A., Ortolà, J., Serra, X.
'Singing Voice Synthesis Combining Excitation plus Resonance and Sinusoidal plus Residual Models'
Proceedings of International Computer Music Conference
Havana 2001

Abstract: This paper presents an approach to the modeling of the singing voice with a particular emphasis on the naturalness of the resulting synthetic voice. The underlying analysis/synthesis technique is based on the Spectral Modeling Synthesis (SMS) and a newly developed Excitation plus Resonance (EpR) model. With this approach a complete singing voice synthesizer is developed that generates a vocal melody out of the score and the phonetic transcription of a song.

My contribution to the work presented in this paper was my collaboration on the conception of the EpR model, the specification of the expressiveness module, and the design and implementation of the singer database.

8 - Amatriain, X., Bonada, J., Loscos, A., Serra, X
'Spectral Modeling for Higher-level Sound Transformation'

Proceedings of MOSART Workshop on Current Research Directions in Computer Music
Barcelona 2001

Abstract: When designing audio effects for music processing, we are always aiming at providing higher-level representations that may somehow fill in the gap between the signal processing world and the end-user. Spectral models in general, and the Sinusoidal plus Residual model in particular, can sometimes offer ways to implement such schemes.

My contribution to the work presented in this paper is in the description of the FX and transformations catalog, as well as in the brief description of the real-time singing voice conversion.

9 - Amatriain, X., Bonada, J., Loscos, A., Serra, X.
'Spectral Processing'
Udo Zölzer Ed., DAFX: Digital Audio Effects, p.554 John Wiley & Sons Publishers. 2002

Description: Digital Audio Effects (DAFX) is the name chosen for the European Research Project COST G6. DAFX investigates the use of digital signal processing, its application to sounds, and its musical use designed to put effects on a sound. The aim of the project and this book is to present the main fields of digital audio effects. It systematically introduces the reader to digital signal processing concepts as well as software implementations using MATLAB. Highly acclaimed contributors analyze the latest findings and developments in filters, delays, modulators, and time-frequency processing of sound. Features include chapters on time-domain, non-linear, time-segment, time-frequency, source-filter, spectral, bit stream signal processing; spatial effects, time and frequency warping and control of DAFX. Also include MATLAB implementations throughout the book illustrate essential DSP algorithms for sound processing, and accompanying website with sound examples available. The approach of applying digital signal processing to sound will appeal to sound engineers as well as to researchers and engineers in the field of signal processing.

My contribution to the chapter of the DAFX book was the section in which the basics of the real-time singing voice conversion system are presented, the supervision together with all the co-writers of the entire chapter, and the MATLAB implementation of all the SMS analysis, synthesis, and all transformations described in the chapter. Obviously this is not included in Appendix A.

10 - Bonada, J., Loscos, A., Mayor, O., Kenmochi, H.
'Sample-based singing voice synthesizer using spectral models and source-filter decomposition'
Proceedings of 3rd International Workshop on Models and Analysis of Vocal Emissions for Biomedical Applications
Firenze 2003

Abstract: This paper is a review of the work contained in the insides of a sample-based virtual singing synthesizer. Starting with a narrative of the evolution of the techniques involved in it, the paper focuses mainly on the description of its current components and processes and its most relevant features: from the singer databases creation to the final synthesis concatenation step.

My involvement in the work presented in this paper is my contributions to the design and implementation of the singing voice synthesizer.

11 - Bonada, J., Loscos, A., 2003.
'Sample-based singing voice synthesizer by spectral concatenation'
Proceedings of Stockholm Music Acoustics Conference
Stockholm 2003

Abstract: The singing synthesis system we present generates a performance of an artificial singer out of the musical score and the phonetic transcription of a song using a frame-based frequency domain technique. This performance mimics the real singing of a singer that has been previously recorded, analyzed and stored in a database, in which we store his voice characteristics (phonetics) and his low-level expressivity (attacks, releases, note transitions and vibratos). To synthesize such performance the systems concatenates a set of elemental synthesis units (phonetic articulations and stationeries). These units are obtained by transposing and time-scaling the database samples. The concatenation of these transformed samples is performed by spreading out the spectral shape and phase discontinuities of the boundaries along a set of transition frames that surround the joint frames. The expression of the singing is applied through a Voice Model built up on top of a Spectral Peak Processing (SPP) technique.

My involvement in the work presented in this paper is my contribution to the EpR model, to the SPP technique, and above all, to the elemental unit phase and spectral shape concatenation.

12 - Amatriain, X., Bonada, J., Loscos, A., Arcos, J., Verfaille, V.
'Content-based Transformations'
Journal of New Music Research Vol.32 .1
2003

Abstract: Content processing is a vast and growing field that integrates different approaches borrowed from the signal processing, information retrieval and machine learning disciplines. In this article we deal with a particular type of content processing: the so-called content-based transformations. We will not focus on any particular application but rather try to give an overview of different techniques and conceptual implications. We first describe the transformation process itself, including the main model schemes that are commonly used, which lead to the establishment of the formal basis for a definition of content-based transformations. Then we take a quick look at a general spectral based analysis/synthesis approach to process audio signals and how to extract features that can be used in the content-based transformation context. Using this analysis/synthesis approach we give some examples on how content-based *transformations can be applied to modify the basic perceptual axis of a sound and how we can even combine different basic effects in order to perform more meaningful transformations. We finish by going a step further in the abstraction ladder and present transformations that are related to musical (and thus symbolic) properties rather than to those of the sound or the signal itself.*

My contribution to the work presented in this journal article is the brief explanation of the real-time singing voice conversion system and my inputs to the text related with the analysis-synthesis processes, and basic and content based transformations. The article has not been included in Appendix A because of its considerable length.

Chapter 5

Thesis Research Project

In this work the main research in which I have been involved inside the MTG has been explained. This research goes in the direction of finding techniques and models for the singing voice processing.

Concerning the techniques, the work has focused on spectral based ones. Both SMS and SPP have proven to be useful for analyzing, transforming and re-synthesizing pseudo-harmonic sounds in a meaningful way but present some drawbacks.

For the SMS, the main drawback is in the difficult discrimination between sinusoids and residual. This problem comes from the voice signal nature itself and not from the technique. The dual source character of the voice does not seem to match with the binary SMS categorization. Such problem can be noticed, for example, when synthesizing in Daisy, where the articulations, in which most of the times some sinusoidal components get partially masked by the residual, do not sound as good as the steady state parts.

The SPP avoids the sinusoidal plus residual decomposition but still drags some problems that have to be improved and solved. These problems come mainly from the reallocation of the harmonic peaks and thus, the decontextualization of the SPP regions. For this reason efforts will to have to be invested in studying the inter harmonics relationships and solving the phase alignment preservation problem, which is a problem that comes from the phase unwrapping problem and that is present in the SMS as well.

However, both SMS and SPP techniques are low level spectral techniques that are lacked of a higher level layer from which they could take profit of the peculiarities of the voice. A voice model is the one to bring such higher level view providing singing voice specific knowledge, such as formant information, to the techniques.

The voice EpR model presented in this work has its origins in the classical voice models developed for speech processing and it goes in the direction of creating singing voice specific processing tools in which technique and model combine. With such combination we have already been able to implement a singing voice synthesizer and apply new transformations to the voice. Anyway, there is room for much improvement, especially in the EpR phase model, in the voice excitation curve model, and the modeling and estimation of the vocal tract resonances.

The research that will be carried out from now on will explore how to combine spectral techniques and singing voice models to come out with a system with which all kind of voice transformations can be accomplished resulting in a natural synthesized sound. The transformations to which we refer go from voice conversion to voice sample based synthesis and include transposition, voice character change such as roughness, breathiness, or whisper, timbre manipulation, and morph.

Appendix A

Publications

SmsPerformer:
A Real-Time Synthesis Interface for SMS

Alex Loscos, Eduard Resina
Audiovisual Institute, Pompeu Fabra University
Rambla 31, 08002 Barcelona, Spain
{aloscos, eduard}@iua.upf.es http://www.iua.upf.es

Abstract

SmsPerformer is a graphical interface for the real-time SMS synthesis engine. The application works from analyzed sounds and it has been designed to be used both as a composition and a performance tool. The program includes programmable time-varying transformations, MIDI control for the synthesis parameters, and performance loading and saving options.

1 Introduction

The SMS synthesis is based on the combination of additive and subtractive synthesis [1]. The SMS analysis output is (1) a collection a frequency and amplitude values that represent the partials of the sound, (2) either filter coefficients and a gain value or spectral magnitudes and phases representing the residual sound and (3) a set of envelopes that represent high level attributes of the sound. From this representation an efficient synthesis can be implemented that offers many transformation possibilities.

The SmsPerformer application is a continuation of the software implementation of SMS started by a C code program with a command line interface developed by X. Serra and followed by a Visual C++ code with graphical interface implemented by J. Bonada named SmsTools [2]. SmsPerformer brings new possibilities to the SMS synthesis with real-time transformations and a graphical interface.

With the power of current general purpose processors it has become feasible to have software real-time implementations of additive synthesis and other musical research teams are developing similar programs [3][4][5].

2 Real-time SMS

Recent changes and optimizations to the SMS software have led to the current real-time implementation.

One of the changes concerns disc access. Loading the analyzed sound file into cache memory rather than reading it from disc speeds up frame fetching.

Interpolation between frames was one of the most stringent processes. Without frame interpolation, just taking the nearest, solves possible clicks in the output device buffer when morphing or time stretching. These problems are produced if the CPU spends more time in computing, synthesizing a given frame and writing it into the buffer, than the actual frame duration.

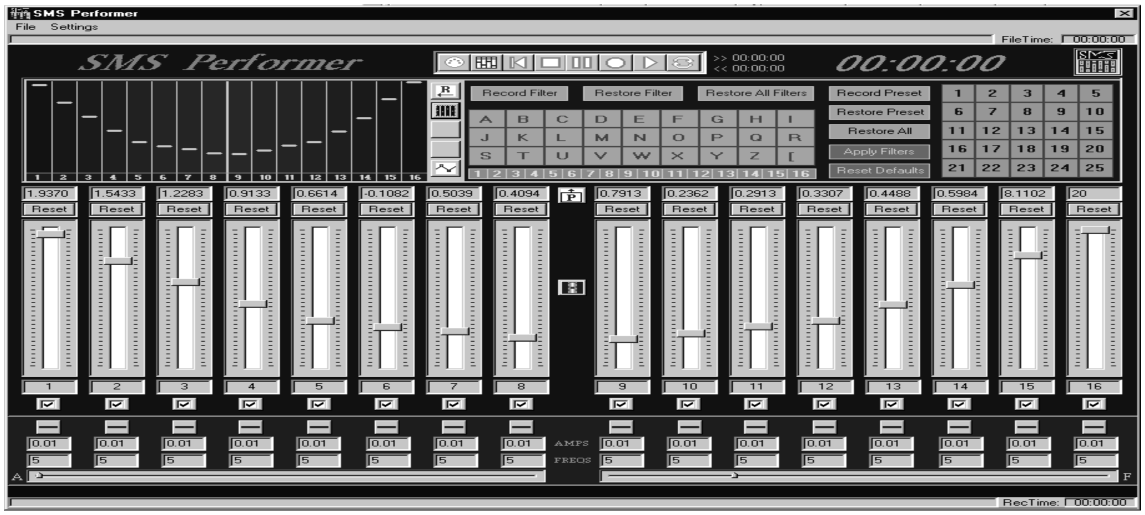
The number of partials to synthesize is another stringent requirement for the system to run in real-time as is the synthesis of the residual. In the analysis we can choose the residual of a frame to be represented as a magnitude spectral envelope, a complete complex spectrum, or a waveform. When kept as a magnitude spectral envelope, or a spectrum, the residual is deconvolved and added to the partials in the frequency domain. Once the spectrum of the current frame is filled, the IFFT is used to synthesize the waveform. When keeping the residual as a waveform, this is added in the time domain to the already synthesized partials, which is faster, but less flexible, than when keeping the residual in the frequency domain.

There is another synthesis process that presents a trade-off between computation time and flexibility. This the high-level attributes to be transformed and added back to the low level SMS representation [6]. More powerful and intuitive musical transformations can be achieved by controlling the high-level attributes at the expense of more operations to be carried in the synthesis.

3 Real-time under Windows

SmsPerformer has been programmed under Windows NT with Visual C++ 5.0 and using the SMS class library.

Because of the interactivity requirement we had to use two threads, one for the main synthesis process and the other for the graphical interface. The main window process creates the synthesis thread and sends new synthesis and transformation parameters to the thread each time a slider is scrolled.



1.9370 1.5433 1.2283 0.9133 0.6614 -0.1082 0.5039 0.4094 0.7913 0.2362 0.2913 0.3307 0.4488 0.5984 8.1102 20

Figure 1. SmsPerformer main window.

When threading a sound application, setting priorities is not very reliable. The system uses the base priority level of all executable threads to determine which thread gets the next slice of CPU time. When setting the synthesis thread to a higher priority, we found the slider-scroll update rate was too slow, so we had both threads running with the same priority level.

SmsPerformer can run the synthesis and all available transformations in parallel and in real-time in a Pentium Processor 200 MHz, 32Mb RAM, under Windows 98 or NT, with 40 sinusoids, waveform residual, 85% of the system resources free. In terms of latency, using ISA (Sound Blaster) and PCI (Aztec and Event) sound cards we obtained transformation response delays under 40 milliseconds. Different framework conditions have different limitations.

3.1 Audio playback

SmsPerformer has been implemented using both Microsoft Win32 application programming interface low-level API and DirectX DirectSound [7][8][9].

The API option presents some tricks once you have sent the sound data to the sound card (filling the lpData field of the WAVEHDR structure and using the waveOutPrepareHeader and waveOutWrite functions) and you have to unprepare the WAVEHDR making sure all sound data has already been written in the Output Audio Device buffer to reuse it (using the waveOutUnprepareHeader function).

The waveOutGetPosition function retrieves the current playback position of the given waveform-audio output device but it does it very inaccurately. This forced us to work with an error margin. On the other hand the WHDR_DONE flag of the dwFlags field of the WAVEHDR structure is set by the device driver to indicate that it is finished with the buffer and is returning it to the application, but this setting has an important delay. So in both cases we need to increase the number of wave headers structures (SmsPerformer uses short sound data buffers).

DirectSound provides low-latency mixing, hardware acceleration, and direct access to the sound device. Its circular view enables infinite streaming buffers; as the front of the buffer is being consumed, the rear of the buffer can be refilled [10][11].

The write cursor indicates the position at which it is safe to write new data to the buffer. The write cursor always leads the play cursor, typically by about 15 milliseconds worth of audio data (shown in *Figure 2)*.

SmsPerformer implementation uses a streaming secondary circular buffer of 16 frames, located in

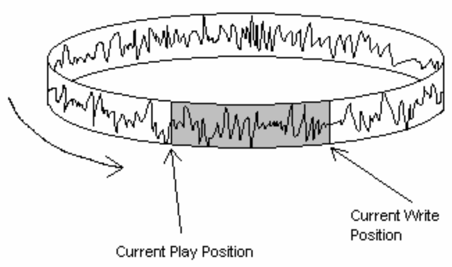


Figure 2. Secondary buffer with current play and write

system memory. DirectSound not only has a lower playback latency but also needs less sound buffers.

4 SmsPerformer features

SmsPerformer has many ways to control the synthesis-transformation of the sound: the main window sliders, a score file, a MIDI controller, or a set of graphical envelopes.

4.1 Main window sliders

The most immediate way to use the program is scrolling the main window sliders. You can configure which synthesis-transformation parameter controls each slider and set the maximum and minimum values of the parameter slider in a dialog window.

You can also modify the sliders value by drawing with the mouse the configuration you want the sliders to have. This can done in the up-left panel of the main window shown in *Figure 1*.

The sliders values can be modulated by a sinusoidal signal and also by a uniform random noise. When sinusoidal modulation is chosen you can control amplitude and frequency of the modulation with the sliders placed above. When noise modulation is chosen you can only control the amplitude of the modulation.

4.2 External MIDI controls

The application accepts MIDI messages for changing the transformation parameters from an external MIDI controller. In this way you can get a better physical feeling (like playing an instrument) than using the main window sliders. It is also possible to change more than one parameter at a time. For this purpose we have used a Peavey MIDI Controller PC1600 (shown in *Figure 3)*.

The MIDI controller sends channel pitch bend messages to the application. The status bit is used to specify the slider that has to receive the message and it is followed by data bytes that specify the position of the slider between the maximum and the minimum defined. The message is like

First byte	Second byte	Third byte
0xEn	LSB	MSB

where n is the number of slider (channel) to control, 0 for slider 1 and F for slider 16, and then data is specified with a 14-bit number, using two 7-bit bytes, least significant byte first [12].

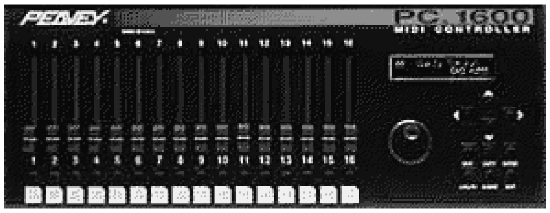


Figure 3. Peavey MIDI Controller PC1600.

A dialog window helps you specify the MIDI device, the number of data bytes of the messages, and the update message rate. This message rate indicates how often the sliders values are updated (pitch bend MIDI messages between time lapses are dismissed).

4.3 MIDI files

You can save performances as MIDI files and play already saved performances by loading them. The MIDI file saves the value, the slider (channel) and the time of a value change. When playing a MIDI file you can act on it modifying some parameter values at the time you hear the saved performance.

4.4 Score files

SmsPerformer can also read sms score files: synthesis files (*.syn) and hybridization files (*.hyb). These files are text files in which you can specify all sms synthesis and transformation parameters [2].

SmsPerformer can hybridize two sounds and modify hybridization parameters while the sound is being played. To do so, once the two sounds have been analyzed and saved as sms files, you need a hybridization file in which the two sms files have been specified. This is an example of hybridization file:

```
InputSmsFile violin.sms
InputSmsHybridFile trumpet.sms
Hybridize 1
```

4.5 Filters and presets

There is a bank of configurable filters and presets. Filters refer to which sliders are active and which are not. Presets refer to the value of each slider. You can also disactivate a slider by clicking the checkbox above it.

There are a number of preset options but you can create your own filters and presets and save them. You can see the filter and preset windows in the right upper side of the main application window (shown in *Figure 1*).

4.6 Envelope

You can have a performance by defining an envelope. This envelope is defined in a x-time axis and y-preset axis as shown in *Figure 4*.

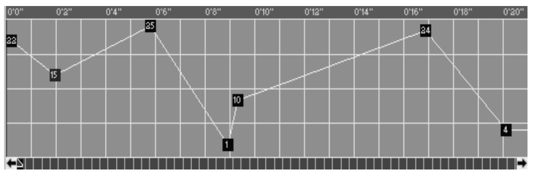


Figure 4. Envelope window.

You can define the presets you need and place them in the time axis. The values of the sliders are interpolated from one point to the next. You can modify both time and preset values of an envelope point while the sound is playing.

4.7 Play modes

The play modes in which you can work when performing with a given envelope, or with saved performances, are Normal, Loop and No-End.

Normal play mode plays the sound only once. If the transformations defined, or saved, make reference to a time outside the actual sound duration, it does not use them.

Loop play mode loops the sound until you click stop. This mode also loops transformations saved or defined whatever time duration they have.

No-End play mode loops the sound but not the transformations. So the sound is modified until the last saved or defined transformation time and then keeps looping maintaining the last parameters values. Clicking stop breaks the loop.

5 Conclusions

SmsPerformer is the first application that makes use of the real-time possibilities of the SMS software. Despite the problems of the Windows operating system to support real-time audio applications, SmsPerformer has shown to be useful for music applications. It is the first attempt of a performance tool and as such there is room for many improvements.

6 Acknowledgements

We would like to acknowledge the contribution to this research of the other members of the Music Technology Group of the Audiovisual Institute.

References

[1] X. Serra. “Musical Sound Modeling with Sinusoids plus Noise”, in G. D. Poli, A. Picialli, S. T. Pope, and C. Roads, editors. *Musical Signal Processing*. Swets & Zeitlinger Publishers. 1996.

[2] Music Technology Group. *SMS Web site*. URL: http://www.iua.upf.es/~sms.

[3] M. Wright, D. Wessel and A. Freed. *"New Musical Control Structures from Standard Gestural Controllers"* Proceedings of the ICMC. 1997. [Available at http://cnmat. CNMAT.Berkeley.EDU/ICMC97/papers-html/ Tablet.html]

[4] IRCAM Sound Analysis/Synthesis Group. *Web site*. URL: http://www.ircam.fr/activites/ recherche/ana-syn-e.html

[5] Bradford Enhanced Synthesis Technology Group *Web Site*. URL: http://www.comp. brad.ac.uk/research/music/simulator.html

[6] X. Serra and J. Bonada. “Sound Transformations based on the SMS High Level Attributes”. *Proceedings of the Digital Audio Effects Workshop*. 1998.

[7] P. L. Childs. *Audio Latency Analysis for Windows-based Systems*. URL:http://telephony. cornell.edu/Latency.html, 1997.

[8] A. Freed, A. Chaudhary, and B. Davila. “Operating Systems Latency Measurement and Analysis for Sound Synthesis and Processing Applications”. *Proceedings of the ICMC,* 1997. [Available at http://cnmat.CNMAT.Berkeley. EDU/ ICMC97/papers-html/Latency.html]

[8] E. Brandt and R. Dannenberg. “Low-Latency music software using off-the-shelf operating systems.” *Proceedings of the ICMC,* 1998.

[9] R. Dannenberg and M. Goto. “Latency in Computer Audio Systems” *Proceedings of the ICMC,* 1997. [Available at http://raven. dartmouth.edu/~icma/array/spring97/articles. html]

[10] B. Bargen and P. Donnelly. *Inside DirectX.* Microsoft Press, *1998.*

[11] Microsoft. *DirectX Web Site*. URL: http://www.microsoft.com/directx/default.asp.

[12] P. Messik. *Maximum Midi.* Manning Publications, 1998.

Low-Delay Singing Voice Alignment to Text

Alex Loscos, Pedro Cano, Jordi Bonada
Audiovisual Institute, Pompeu Fabra University
Rambla 31, 08002 Barcelona, Spain
{aloscos, pcano, jboni }@iua.upf.es http://www.iua.upf.es

Abstract

In this paper we present some ideas and preliminary results on how to move phoneme recognition techniques from speech to the singing voice to solve the low-delay alignment problem. The work focus mainly on searching the most appropriate Hidden Markov Model (HMM) architecture and suitable input features for the singing voice, and reducing the delay of the phonetic aligner without reducing its accuracy.

1 Introduction

An aligner is a system that automatically time aligns speech signals with the corresponding text. This application emerges from the necessity of building large time-aligned and phonetically labeled speech databases for Automatic Speech Recognition (ASR) systems. The most extended and successful way to do this alignment is by creating a phonetic transcription of the word sequence comprising the text and aligning the phone sequence with the speech using a Hidden Markov Model (HMM) speech recognizer [1].

The phoneme alignment can be considered speech recognition without a large portion of the search problem. Since we know the string of spoken words the possible paths are restricted to just one string of phonemes. This leaves time as the only degree of freedom and the only thing of interest then is to place the start and end points of each phoneme to be aligned. For the case of aligning singing voice to the text of a song, more data is available out of the musical information: the time at which the phoneme is supposed to be sung, its approximate duration, and its associated pitch.

We have implemented a system that can align the singing voice signal to the lyrics in real time. Thus, as the singer performs, the signal can be processed and different specific audio effects applied depending on which phoneme of the lyrics is currently being sung. This pursues the idea of content based processing.

2 Singing voice to text aligner

In this section, we consider the main differences between speech and singing voice, and present our proposal for the singing voice to text aligner by searching the most appropriate HMM architecture and suitable input features for the singing voice. Finally we show how to build the composite Finite State Network (FSN) of the song.

2.1 Speech and Singing Voice

Although speech and singing voice sounds have many properties in common because they originate from the same production physiology, there are some differences to bear in mind.

-**Voiced/unvoiced ratio**: The ratio between voiced, unvoiced sounds, and silence is about (60%, 25%, 15%) in the case of normal speech. In singing, the percentage of phonation time can increase up to 95% in the case of opera music.

-**Dynamic**: The dynamic range as well as the average loudness is greater in singing than in speech. The spectral characteristics of a voiced sound change with the loudness [2].

-**Fundamental frequency**: In speech, the fundamental frequency variations express an emotional state of the speaker or add intelligibility to the spoken words. This frequency range of f_0 is very small compared to singing where it can be up to three octaves.

-**Vibrato**: Two types of vibrato exist in singing. The classical vibrato in opera music corresponds to periodic modulation of the phonation frequency, and in popular music the vibrato implies an amplitude modulation of the voice source [3]. In speech, no vibrato exists.

-**Formants**: Because in singing the intelligibility of the phonemic message is often secondary to the intonation and musical expression qualities of the voice, in cases like high pitch singing, wide excursion vibratos, hoarse and aggressive attacks or very loud singing, there is an alteration of the formants position, and therefore the perceived vowel is slightly modified.

2.2 HMM Architecture

As the task of the alignment can be considered as simplified speech recognition, it is natural to adopt a successful paradigm of the ASR, namely HMM, for the alignment. Our approach attempts to use this model for the singing voice case and tune its parameters to make the model singing voice case specific. This tuning has to take into account the following considerations:

(a) No large singing voice database is available to train the model.
(b) The final system will have to align with the minimum possible delay.
(c) The alignment will have phoneme resolution.

The aligner will be a phoneme-based system (c). In this type of systems, contextual effects cause large variations in the way that different sounds are produced. Although training different phoneme HMMs for different phoneme contexts (i.e. triphonemes) would present better phonetic discrimination, this is not recommended in the case (a) no large database is available.

HMMs can have different types of distribution functions: discrete, continuous, and semi continuous. Discrete distribution HMMs match better with small train database and are more efficient computationally [4]. Because of this and considerations (a) and (b), in this first approach, the nature of the elements of the output distribution matrix will be discrete.

The most popular way in which speech is modeled is as a left-to-right HMM with 3 states. We also fit 3 states to most of the phonemes (except for the plosives) as an approach to mimic the attack, steady state and release stages of a note. The plosives are modeled with 2 states to take into consideration somehow their intrinsic briefness, and the silence is modeled with 1 state as it is in speech.

2.3 Front-end Parameterization

The function of this stage is to extract the features that will be used as the observations of the HMMs. To do so the input signal is divided into blocks and from each block the features are extracted. For the singing voice we keep the speech assumption that the signal can be regarded as stationary over an interval of a few milliseconds. Various possible choices of vectors together with their impact on recognition performance are discussed in [5]. Our choice of features to be extracted from the sound in the front-end is the following:

Mel Cepstrum:	12 coefficients
Delta Mel Cepstrum:	12 coefficients
Energy:	1 coefficient
Delta Energy:	1 coefficient
Voiceness:	2 coefficients

With:

Window Displacement:	5.8 ms
Window Size:	20 ms
Window Type:	Hamming
Sampling Rate:	22050 Hz

To compute the Mel frequency cepstral coefficients (MFCC) the Fourier spectrum is smoothed integrating the spectral coefficients within triangular frequency bins arranged on the non-linear Mel-scale. The system uses 24 of these triangular frequency bins (from 40 to 5000 Hz). In order to make statistics of the estimated speech power spectrum approximately gaussian, log compression is applied to the filter-bank output. The final processing stage is to apply the Discrete Cosine Transform to the log filter-bank coefficients.

The voiceness vector consists of a Pitch Error measure and the Zero Crossing rate. The Pitch Error component is a byproduct from the fundamental frequency analysis, which is based on [6]. The zero crossing rate is calculated by dividing the number of consecutive samples with different signs by the number of samples of the frame.

The acoustic modeling assumes that each acoustic vector is uncorrelated with its neighbors. This is a rather poor assumption since the physical constraints of the human vocal apparatus ensure that there is continuity between successive spectral estimates. However, considering differentials to the basic static coefficients greatly reduces the problem. This differential ponders up to two frames of the future and two frames of the past.

2.4 Composite FSN

The alignment process starts with the generation of a phonetic transcription out of the lyrics text. This phonetic transcription is used to build the composite song FSN concatenating the models of the phonemes transcribed.

The phonetic transcription previous to the alignment process has to be flexible and general enough to account for all the possible realizations of the singer. It is very important to bear in mind the non-linguistic units silence and aspiration as they appearance cannot be predicted. Different singers place silences and aspirations in different places. This is why while building the FSN, between each pair of phoneme models, we insert both silence and aspiration models. In the transition probability matrix of the FSN, the jump probability a_{ij} from each speech phonetic unit to the next silence, aspiration or speech phonetic unit will be the same as shown in figure 1.

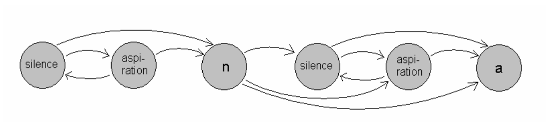


Figure 1: Concatenation of silences and aspirations in the FSN

The aspiration is problematic since in singing its dynamic is more significant. This causes that the aspiration can be easily confused with a fricative.

Moreover, different singers not only sing different but also, as in speech, pronounce different. To take into account these different pronunciations we modify the FSN to add parallel paths as shown in figure 2.

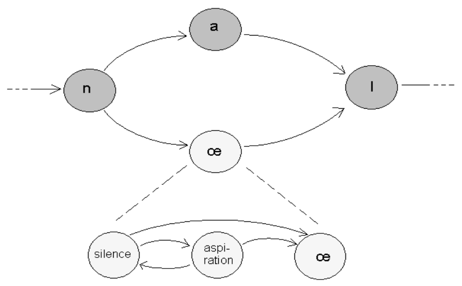


Figure 2: Representation of a phonetic equivalence in the FSN

The alignment resultant from the Viterbi decoding will follow the most likely path, so it will decide if it is more probable that it was phoneme [a] or phoneme [œ] the one sang.

3 Low delay alignment

In this section we modify the Viterbi algorithm to allow low-delay. To compensate the loss of robustness caused by this, some strategies on discarding phony candidates are introduced to preserve a good accuracy.

3.1 Low-delay Viterbi decoding

The usual decoding choice in the text to speech alignment problem is the Viterbi algorithm [7]. This algorithm gives as result the most probable path through the models, giving the points in time for every transition from one phoneme model to the following.

Most applications perform the backtracking at the end of the utterance. In the case of a limited decoding delay, backtracking has to be adapted in order to determine the best path at each frame iteration. If we consider a decoding delay of Δm frames, we will have to follow the backtracking pointers of the selected best path to determine the associated phone index in the FSN Δm frames before. Strategies for low delay backtracking are discussed in [8] for the analogous case of recognition.

In general, the best path at frame m will be different from the best path at the end of the utterance. As a general rule, the reduction in the delay causes an important degradation in performance. However, since we want to be able to offer real time audio effects, we will work with the most extreme case, deciding for each input frame the current singer position in the lyrics with a decoding delay of $\Delta m=0$. To avoid a large amount of jumps from one path to a complete different one, we introduce some strategies.

3.2 Strategies for discarding candidates

During the low-delay alignment we have several hypothesis on our location in the song with similar probability. We use heuristic rules as well as musical information from the score to discard candidates.

An example of a knowledge based candidates discarding is that once we have decided we are in a certain fricative of the target, since the fricatives are recognized very reliably, the only candidates we consider are the fricative and the phonemes that comes next in the phonetic transcription of the target.

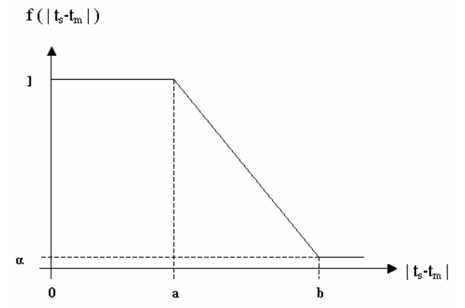


Figure 3: Function of the factor applied to the Viterbi probability

We have also implemented routines that use the information that we have apart from the lyrics. Since we are aligning to a song, we know that the phoneme corresponding to a note in the score is supposed to have certain duration. Moreover, the user, supposedly, sings following the tempo so we take advantage of this fact to better choose a phoneme from the target by modifying the output Viterbi probabilities by the function shown in figure 3.

In this figure 3 t_s is the time at which the phoneme happens in the singer performance, t_m is the time at which this phoneme happens in the song score, parameters a and b are tempo and duration dependent, and α has a very low value, nearly null. This function can be defined differently for the case in which the singer comes from silence and attacks the beginning of a word, and for the case the singer has already started a word, due to the very different behaviors of these two situations.

4 Results

The aligner has been tested over a set of songs and it has proved to be quite accurate and robust for all kind of singers. In order to check the performance of the system, we have implemented a graphical interface where the results of the alignments can be displayed as shown in figure 4.

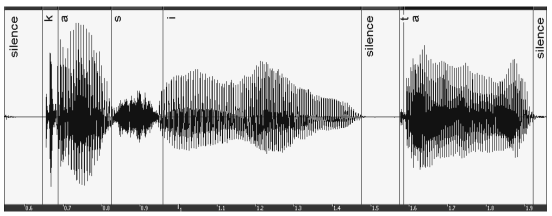


Figure 4: View of the real-time alignment results in the graphical interface

The Time Delay (TD) of the system has been computed from the formulation done in [8], in which only the intrinsic delay of the alignment algorithm is taken into account. Therefore, the following delays are considered: 10 ms due to the Window Size (WS), and 11.6 ms due the Window Displacement (WD) and the Delta Frames (DF). No delay is due to second derivatives, as we are not using acceleration feature computations (AF=0), neither any delay is introduced in the Viterbi decoding step (DD=0). This is:

$$\mathrm{TD} > \frac{\mathrm{WS}}{2},\ \mathrm{WD}(\mathrm{DF},\ \mathrm{AF},\ \mathrm{DD})$$

This makes a delay of 21 ms, which has to be added to the hardware delay to get the total delay of the system.

5 Conclusions

Certainly the system can be improved, especially in certain phone transitions. We believe taking into consideration the pitch information could bring about some improvements. In the system, the pitch information has been discarded so that singers could be aligned regardless in how in tune they sing. However, if we can rely on the singer's pitch, this information can be very useful to improve the accuracy of the phone boundaries. We can even think of a hybrid system where two parallel alignments, phonetic and musical [9], would merge to complement each other.

We believe that using context dependent phoneme models and using non-discrete symbol probability distributions would bring better results. This is why part of our efforts have to focus on building a large singing voice database, which at this point of time is 22 minutes large.

6 Acknowledgements

We would like to acknowledge the contribution to this research of the other members of the Music Technology Group of the Audiovisual Institute.

7 References

[1] A. Waibel and K. F. Lee. *Readings in Speech Recognition*. Morgan Kaufmann, 1990.

[2] J. Oliveiro, M.A. Clements, M.W. Macon ,L. Jensen-Link and E.B.George. "Concatenation based midi-to-singing voice synthesis" *AES Preprint 4591*, 103rd Meeting of the AES, September 1997.

[3] J. Sundberg. *The Science of Singing Voice*. Illinois Universitary Press, 1987.

[4] S. Young. "Large Vocabulary Speech Recognition: a Review". *Technical Report*, Cambridge University Engineering Department, 1996.

[5] R. Haeb-Umbach, D. Geller, and H. Ney. "Improvements in connected digit recognition using linear discriminant analysis and mixture densities". *Proceedings of the ICASSP*, 1993.

[6] P. Cano. "Fundamental Frequency Estimation in the SMS Analysis". *Proceedings of the Digital Audio Effects Workshop*, 1998.

[7] L. Rabiner and B.H. Juang. *Fundamentals of Speech Recognition*. Prentice Hall, 1993.

[8] J.A.R. Fonollosa, E. Batlle, J.B. Mariño. "Low Delay Phone Recognition". EURASIP IX European Signal Processing Conference. EUSIPCO, September 1998.

[9] P. Cano, A. Loscos, and J. Bonada "Score-performance matching using HMMs". *Proceedings of the ICMC*, 1999.

Score-Performance Matching using HMMs

Pedro Cano, Alex Loscos, Jordi Bonada
Audiovisual Institute, Pompeu Fabra University
Rambla 31, 08002 Barcelona, Spain
{ pcano, aloscos, jboni }@iua.upf.es http://www.iua.upf.es

Abstract

In this paper we will describe an implementation of a score-performance matching, capable of score following, based on a stochastic approach using Hidden Markov Models.

1 Introduction

Linking notes in a musical performance to the corresponding notes in a score is called score-performance matching. Proper matching algorithms are crucial for real-time composition and automatic accompaniment systems, where the computer has to find out where the musician is with respect to a known score, in order to make an appropriate musical response (like playing an accompaniment). In the context of musical performance research, matching algorithms are necessary to be able to measure timing: it has to be known which performance note relates to which score note to be able to extract expressive timing patterns, calculate tempo, pitch deviation patterns... Also audio effects can be applied depending on our location in the score.

One category of algorithms focuses on real-time matching and they are often called score-following. Early work in score matching was performed by Dannenberg and Vercoe, both primarily interested in making real-time matchers. Dannenberg describes an algorithm solely based on pitch information, intended to robustly follow a monophonic instrument. Later these algorithms were extended to deal with polyphonic music and multiple instruments [1]. Most algorithms primarily match pitch, possibly in combination with time information. The method here presented focuses on monophonic music and can be seen as a continuation of Puckette's work [2] moving from knowledge-based to stochastic models.

2 Stochastic modeling

The input features needed to make a decision when performing a match, namely fundamental frequency, notes duration, etc, are inherently uncertain. The uncertainty rises from the fact that instrument players or singers do not perform an ideal realization of what it is written in the score and the fundamental frequency algorithms are not absolutely reliable. Others algorithms can perform reasonably well with specific instruments like flute but they fail when the problem above stated is especially troublesome. This occurs very clearly with the singing voice in which the output does not resemble at all a sequence of discrete tempered pitches attained at well-defined times.

Stochastic modeling is a flexible general method to situations like the above described. It consists of employing a specific probabilistic model for the uncertainty or incompleteness of the information. A music performance is a nonstationary process since its statistical parameters vary over time. Hidden Markov Models (HMM) are already used for statistical modeling of nonstationary stochastic processes such as speech.

3 Hidden Markov Models

A HMM is most easily understood as a generator of vector sequences. It is a finite state machine which changes state once every time unit and each time t that a state j is entered, a n acoustic speech vector y_t is generated with probability density $b_j(y_t)$. Furthermore, the transition from state *i* to state *j* is also probabilistic and governed by the discrete probability a_{ij}. In the figure below we show an example of this process where the model moves through the state sequence *X*=1,1,2,2,2,2,2,3,3,3 in order to generate the 10 observation vectors of *k-index model*.

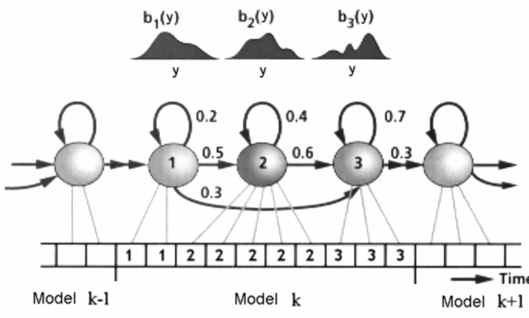


Figure 1: Markov Process example

The joint probability of a vector sequence *Y* and state sequence *X* given some model M is calculated simply as the product of the transition probabilities and the output probabilities. The joint probability of an acoustic vector sequence *Y* and some state sequence *X = x(1), x(2), x(3),…,x(T)* is:

$$P(Y,X|M) > a_{x(0)x(1)} \overset{T}{\underset{t>1}{\sqrt{}}}\; b_{x(t)}(y_t)a_{x(t)x(t,\,1)}$$

In practice, of course, only the observation sequence *Y* is known and the underlying state sequence *X* is hidden. This is why it is called *Hidden Markov Model*.
In our problem, the model M is the sequence of notes specified in the score script, Y the feature sequence extracted from the audio signal and X, the state sequence, which is actually what we aim for in this paper.

4 Note models based in HMM

In this section we present the features and the characteristics of the HMMs used to model the notes.

4.1 Front-End Parameterization

The function of front-end parameterization stage is to divide the input signal into blocks and to extract from each block relevant features. The six features that will be used for the observation sequence are:

Energy,

$$Energy\ > 20\ \log_{10}\left(\sum_{k>0}^{N.\,1} |X)k*|\right)$$

Delta Energy,

$$\mathrm{E}Energy\ (n) > Energy\ (n\ ,\ 1)\ .\ \ Energy\ (n\ .\ 1)$$

Zero Crossing,

$$Z_s(n) > \frac{1}{N} \sum_{m>n.\,M,\,1}^{n} \frac{|\mathrm{sgn}|x(m)\sim.\ \mathrm{sgn}|x(m\ .\ 1)\dashv}{2} w(n\ .\ m)$$

Delta Fundamental Frequency,

$$\mathrm{E}F_0)n*> \begin{cases} \log\left(\dfrac{F_0)n,\ 1*}{F_0)n.\ 1*}\right) & \text{if}\, F_0)n.\ 1*\!\dashv\! F_0)n,\ 1*\equiv 0 \\ \eta & \textit{elsewhere} \end{cases}$$

Fundamental Frequency, and *Fundamental Frequency Error*, which is a measure of "goodness" of the F_0 estimated.

Obtaining fundamental frequencies is a popular subject of study. The particular algorithm we use is an adaptation of the Two-Way Mismatch [3]. In the procedure, the estimated F_0 is chosen as to minimize discrepancies between measured partial frequencies and harmonic frequencies generated by trial values of F_0. For each trial F_0 mismatches between the harmonics generated and the measured partial frequencies are averaged over a fixed subset of the available partials. The predicted to measured error is defined as:

$$Err_{p\downarrow m} > \sum_{n>1}^{N} E_w)\mathrm{E}f_n, f_n, a_n, A_{max}*$$
$$> \sum_{n>1}^{N} \mathrm{E}f_n \dashv f_n *^{\cdot\, p}\ ,\ \left(\frac{a_n}{A_{max}}\right) \propto \dot{q}\mathrm{E}f_n \dashv (f_n)^{\cdot\, p}\ .\ r\perp$$

where $\mathrm{E}f_n$ is the difference between a predicted and its closest measured peak, f_n and are the frequency and magnitude of the predicted peaks, and A_{max} is maximum peak magnitude. The measured to predicted error is defined as:

$$Err_{m\downarrow p} > \sum_{k>1}^{K} E_w)\mathrm{E}f_k, f_k, a_k, A_{max}*$$
$$> \sum_{k>1}^{K} \mathrm{E}f_k \dashv f_k *^{\,p}\ ,\ \left(\frac{a_k}{A_{max}}\right) \propto \dot{q}\mathrm{E}f_k \dashv (f_k)^{\cdot\, p}\ .\ r\perp$$

where $\mathrm{E}f_k$ is the difference between a measured and its closest predicted peak, f_k and a_k are the frequency and magnitude of the measured peaks, and A_{max} is maximum peak magnitude. The total error is:

$$Err_{total}\ > Err_{p\downarrow m}\,/\,N\ ,\ \ \sigma Err_{m\downarrow p}\,/\,K$$

This pitch estimation method gives as then a temporal evolution of the F_0. This envelope is then used to calculate some of the system's input features.

4.2 Note Model Architecture

We have three left-to-right HMM models: a note, a no-note and a silence model. We model notes with 3 states so as to mimic the note behavior of attack, steady state and release. The silence is modeled with only 1 state, since there is no temporal structure to exploit. The no-note, modeled with 3 states, aims to account for all the unpitched sounds that appear in the performance. This can include noisy spurious sounds or in the case of singing voice aspirations, fricatives, plosives…

The choice of output probability function is crucial since it must model the intrinsic variability of the score realizations. Some HMM systems use discrete output probably functions in conjunction with a vector quantizer. Each incoming feature vector is replaced by the index of the closest vector in a precomputed codebook and the output probability functions are just look-up tables containing the probabilities of each possible VQ index. This approach is computationally very efficient but the quantization introduces noises, which limit the precision that can be obtained. Hence, it seems a better choice to use parametric continuous density output distribution, which model the feature vectors directly, for instance with multivariate mixture Gaussian

$$b_j(y_t) = \sum_{m=1}^{M} c_{jm} N(y_t; \nu_{jm}, \mathrm{T}_{jm})$$

Where c_{jm} is the weight of mixture component m in state j and $N(j; \nu, T)$ denotes a multivariate Gaussian of mean ν and covariance T.

This method results in a system with too many parameters to train. Thus, in this work, we use the same large Gaussian codebook for all the states and only estimate different mixture weights for each state.

We build a codebook with vectors that are composed of all the above features but the F_0, which will be treated differently. To construct a codebook l, the N_l corresponding observations, $y_t \sim l$ are clustered into M_l subsets, being M_l the number of mixture components of the codebook. To do this clustering we use LBG algorithm. Then

$$\nu_{lm} = \frac{1}{N_l} \sum_{\substack{m=1 \\ y_t \sim l}}^{M} y_t$$

$$\mathrm{T}_{lm} = \frac{1}{N_l} \sum_{\substack{m=1 \\ y_t \sim l}}^{M} (y_t - \nu_{lm})(y_t - \nu_{lm})'$$

The mixture coefficient c_{jm}, for the case N_{jm} vectors are assigned to the mth mixture of the codebook, results in

$$c_{lm} = \frac{N_{lm}}{N_l}$$

The output probability of the observation F_0 is defined with two discrete symbols, one symbol when F_0 is not defined and another one when F_0 has been found. The discrete symbol in the states of note models is decomposed in a continuous density function whenever F_0 has been found as shown in figure 2.

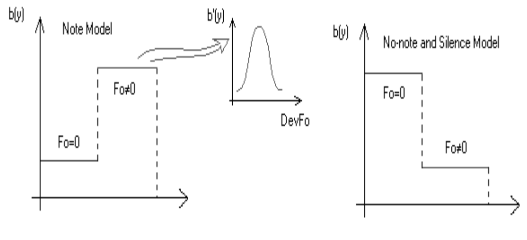


Figure 2 F_0 observation probability function

The continuous density function is modeled as a mixture of gaussians whose input is:

$$F_0 Dev = \left|\log(F_{Perfect\,0}) - \log(F_0)\right|$$

The total output probability is the product of the five-feature vector's and the fundamental frequency's.

5 Model Training

For the training of models we need labeled training set of musical phrases, where each sentence consists of the music waveform and its transcription into notes. According to the transcription we concatenate the HMMs of the musical units to build an extended finite-state network (FSN). It is important to realize that, even though, apart from silences and unvoiced sounds, most models are notes, each HMM note model differs from each other because they keep information of the fundamental frequency associated and also its duration. This information is needed to calculate some input features.

Once a composite sentence FSN is created for each sentence in the training set, the training problem becomes one of estimating the unit model parameters that maximize the likelihood of the models for all the given training data. The maximum likelihood parameters can be solved for using either the forward-backward procedure or the segmental k-means training algorithm. For our system we have chosen the Segmental K-Means [4] and we have implemented if as follows:

1. **Initialization:** Linearly segment each training utterance into units and HMM states.

2. Estimation: The transition probabilities are estimated by merely counting the number of times the transition is used and dividing it by the number of times the source state for the transition is used. This requires maintaining counters to track each transition and each output symbol during training**.**

The mixture weights for the five-feature vector probability function are estimated for each state i as:

$$c_{im} = \frac{N_{im}}{N_i}$$

For the F_0, the output probability function is estimated:

$$\hat{b}_j = \frac{no.\text{ of times in } \mathrm{s_j} \text{ and observed symbol } \mathrm{v_k}}{no.\text{ of times in } \mathrm{s_j}}$$

For the discrete symbol that expands to a continuous density function, we estimate this function the same way as the five-feature one but now it is only the states belonging to note models that share the codebook. This codebook has to be re-estimated every iteration.

3. Segmentation: The updated set of unit models (based on the estimation of step 2) is used to re-segment each training utterance into units and states (via Viterbi decoding).

4. Iteration: Steps 2 and 3 are iterated until convergence.

Since we wanted the system to be accurate in the definition of borders, we manually supervised the resulting segmentation and adjusted some note borders. By doing so, we got a growing parallel database where the notes are labeled and segmented fixed borders. These segmented sentences are used for the training then in a different way than the not segmented ones. Instead of linearly segmenting each training utterance into units and HMM states, for the note-segmented utterances, we only linearly segment the HMM states inside the unit borders. When we iterate, only these inside HMM states borders are allowed to reallocate, the note borders are left fixed. Doing this resulted in a more accurate alignment.

6 Viterbi Decoding

For alignment the trained models are concatenated and run against the feature vectors extracted from the sound using Viterbi decoding. As a result, the most probable path through the models is found, giving the points in time for every transition from one model to the following.

The Viterbi algorithm is an efficient algorithm to find the state sequence that most likely produced the observations. Let *$\gamma_j(t)$* represent the maximum likelihood of observing speech vectors y_1 to y_t and being in state *j* at time *t*. This partial likelihood can be computed using the following recursion

$$\gamma_j(t) > \max_i \Big| \; \gamma_j(t\,.\;1) \neg a_{ij} \sim b_j(y_t)$$

where

$$\gamma_1(1) > 1$$
$$\gamma_j(1) > a_{1j} b_j(y_1)$$

for *$1<j<N$*. The maximum likelihood P'(Y|M) is then given by

$$\gamma_N(T) > \max_i \Big| \; \gamma_j(T) a_{iN} \sim$$

By keeping track of the state j giving the maximum value in the above recursion formula, it is possible, at the end of the input sequence, to retrieve the states visited by the best path, thus obtaining the time-alignment of input frames with models states.

It is possible to modify the algorithm to work on-line. To do so, the backtracking is adapted to determine the best path at each frame iteration instead of waiting until the end of the utterance. This adjustment implies a lost of robustness, some hints to overcome it can be found in [5].

The implementation for explicit note duration modeling by modifying the Viterbi algorithm [6] allows us to include the note duration information. The proposed modified Viterbi algorithm keeps track of the duration *D(t)* of each note *n* at time *t* and introduces a duration penalty P of making a transition from state *i* at time *t* to state j at time *t+1* given by,

$$P > \begin{cases} 0 \quad \textit{if } \mathrm{E}D(t) = 0 \; \textit{and } i > j \\ l(\mathrm{E}D(t\,,\;1))\,.\;l(\mathrm{E}D(t)) \quad \textit{if } \mathrm{E}D(t) \times 0 \; \textit{and } i > j \\ l(\mathrm{E}D(t)) \quad \textit{if } \mathrm{E}D(t) = 0 \; \textit{and } i \equiv j \\ l(0) \quad \textit{if } \mathrm{E}D(t) \times 0 \; \textit{and } i \equiv j \end{cases}$$

where E*D(t)*= *D(t)* - D_n is the difference between the duration of the model and the note duration, D_n, specified in the score, and *l(u)=log p(u), p(.)* is the probability density of E*D* and it is modeled by mixture gaussian densities.

7 Conclusions

The instrument we have mainly worked with is the singing voice. This choice is due not only to the fact that, from all instruments, the larger training database available for us is the singing voice one, but also because we believe this instrument is the most critic, and once solved the score-matching problem for the singing voice case, we will have solved it for any other harmonic instrument. There is still experimentation to be done to tune the different parameters but results are promising.

Improvements can be achieved by considering context. We can train different note models depending on the notes that precede and follow them. In the case of the singing voice, using the lyrics [5] can add robustness to the alignment. This kind of information can also be used to improve accuracy.

References

[1] P. Desain, H. Honing and H. Heijink. "*Robust Score-Performance Matching: Taking Advantage of Structural Information*" ICMC Proceedings, 1997.

[2] M. Puckette. "*Score following using the sung voice*" ICMC Proceedings, 1995.

[3] P. Cano . "*Fundamental Frequency Estimation in the SMS Analysis*" DAFX Proceedings 1998.

[4] L. Rabiner and B.H. Juang *Fundamentals of Speech Recognition*. Prent ice Hall, 1993

[5] A. Loscos, P. Cano, J. Bonada. "*Singing Voice Alignment to text*" ICMC Proceedings, 1999.

[6] D. Burshtein, "*Robust Parametric Modeling of Durations in HMMs*" ICASSP Proceedings 1995

Voice Morphing System for Impersonating in Karaoke Applications

Pedro Cano, Alex Loscos, Jordi Bonada, Maarten de Boer, Xavier Serra
Audiovisual Institute, Pompeu Fabra University
Rambla 31, 08002 Barcelona, Spain
{pcano, aloscos, jboni, mdeboer, xserra}@iua.upf.es
http://www.iua.upf.es

[*Published in the Proceedings of the ICMC 2000*]

Abstract

In this paper we present a real-time system for morphing two voices in the context of a karaoke application. As the user sings a pre-established song, his pitch, timbre, vibrato and articulation can be modified to resemble those of a pre-recorded and pre-analyzed recording of the same melody sang by another person. The underlying analysis/synthesis technique is based on SMS, to which many changes have been done to better adapt it to the singing voice and the real-time constrains of the system. Also a recognition and alignment module has been added for the needed synchronization of the user's voice with the target's voice before the morph is done. There is room for improvements in every single module of the system, but the techniques presented have proved to be valid and capable of musically useful results.

1. Introduction

With different names, and using different signal processing techniques, the idea of audio morphing is well known in the Computer Music community (Serra, 1994; Tellman, Haken, Holloway, 1995; Osaka, 1995; Slaney, Covell, Lassiter. 1996; Settel, Lippe, 1996). The main goal of the developed audio morphing methods is the smooth transformation from one sound to another, thus, the combination of two sounds to create a new sound with an intermediate timbre. Most of these methods are based on the interpolation of sounds parameterizations resulting from analysis/synthesis techniques, such as the Short-time Fourier Transform (STFT), Linear Predictive Coding (LPC) or Sinusoidal Models.

In this paper we present a very particular case of audio morphing. What we want is to be able to morph, in real-time, a user singing a melody with the voice of another singer. It results in an "impersonating" system with which the user can morph his/her voice attributes, such as pitch, timbre, vibrato and articulation, with the ones from a prerecorded target singer. The user is able to control the degree of morphing, thus being able to choose the level of "impersonation" that he/she wants to accomplish. In our particular implementation we are using as the target voice a recording of the complete song to be morphed. A more useful system would use a database of excerpts of the target voice, thus choosing the appropriate target segment at each particular time in the morphing process.

The obvious use of our technique is in Karaoke applications. In such a situation it is very common for the user to want to impersonate the singer that originally sang the song. Our system is capable to do that automatically.

In order to incorporate to the user's voice the corresponding characteristics of the "target" voice, the system has to first recognize what the user is singing (phonemes and notes), finding the same sounds in the target voice (i.e. synchronizing the sounds), then interpolate the selected voice attributes, and finally generate the output morphed voice. All this has to be accomplished in real-time.

Next we present the overall system functionality, then we discuss the basic techniques used and finally we comment on the results obtained. In another paper (Cano, Loscos, Bonada, M. de Boer, Serra, 2000) the actual software implementation is discussed.

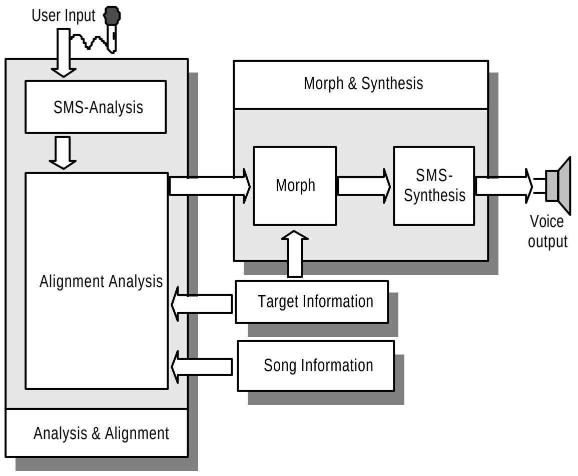


Figure 1. System block diagram.

2. The Voice Morphing System

Figure 1 shows the general block diagram of the voice impersonator system. The underlying analysis/synthesis technique is SMS (Serra, 1997) to which many changes have been done to better adapt it to the singing voice and to the real-time constrains of the application. Also a recognition and alignment module was added for synchronizing the user's voice with the target voice before the morphing is done.

Before we can morph a particular song we have to supply information about the song to be morphed and the song recording itself (Target Information and Song Information). The system requires the phonetic transcription of the lyrics, the melody as MIDI data, and the actual recording to be used as the target audio data. Thus, a good impersonator of the singer that originally sang the song has to be recorded. This recording has to be analyzed with SMS, segmented into "morphing units", and each unit labeled with the appropriate note and phonetic information of the song. This preparation stage is done semi-automatically, using a non-real time application developed for this task.

The first module of the running system includes the real-time analysis and the recognition/alignment steps. Each analysis frame, with the appropriate parameterization, is associated with the phoneme of a specific moment of the song and thus with a target frame. The recognition/alignment algorithm is based on traditional speech recognition technology, that is, Hidden Markov Models (HMM) that were adapted to the singing voice (Loscos, Cano, Bonada, 1999).

Once a user frame is matched with a target frame, we morph them interpolating data from both frames and we synthesize the output sound. Only voiced phonemes are morphed and the user has control over which and by how much each parameter is interpolated. The frames belonging to unvoiced phonemes are left untouched thus always having the user's consonants.

3. Voice analysis/synthesis using SMS

The traditional SMS analysis output is a collection of frequency and amplitude values that represent the partials of the sound (sinusoidal component), and either filter coefficients with a gain value or spectral magnitudes and phases representing the residual sound (non sinusoidal component) (Serra, 1997). Several modifications have been done to the main SMS procedures to adapt them to the requirements of the impersonator system.

A major improvement to SMS has been the real-time implementation of the whole analysis/synthesis process, with a processing latency of less than 30 milliseconds and tuned to the particular case of the singing voice. This has required many optimizations in the analysis part, especially in the fundamental frequency detection algorithm (Cano, 1998). These improvements were mainly done in the pitch candidate's search process, in the peak selection process, in the fundamental frequency tracking process, and in the implementation of a voiced-unvoiced gate (Cano, Loscos, 1999).

Another important set of improvements to SMS relate to the incorporation of a higher-level analysis step that extracts the parameters that are most meaningful to be morphed (Serra, Bonada, 1998). Attributes that are important to be able to interpolate between the user's voice and the target's voice in a karaoke application include spectral shape, fundamental frequency, amplitude and residual signal. Others, such as pitch micro variations, vibrato, spectral tilt, or harmonicity, are also relevant for various steps in the morphing process or to perform other sound transformation that are done in parallel to the morphing. For example, transforming some of these attributes we can achieve voice effects such as Tom Waits hoarseness (Childers, 1994).

4. Phonetic recognition/alignment

This part of the system is responsible for recognizing the phoneme that is being uttered by the user and also its musical context so that a similar segment can be chosen from the target information.

There is a huge amount of research in the field of speech recognition. The recognition systems work reasonably well when tested in the well-controlled environment of the laboratory. However, phoneme recognition rates decay miserably when the conditions are adverse. In our case, we need a speaker independent system capable of working in a bar with a lot of noise, loud music being played and not very-high quality microphones. Moreover the system deals with singing voice, which has never been worked on and for which there are no available databases. It has to work also with very low delay, we cannot wait for a phoneme to be finished before we recognize it and we have to assign a phoneme to each frame.

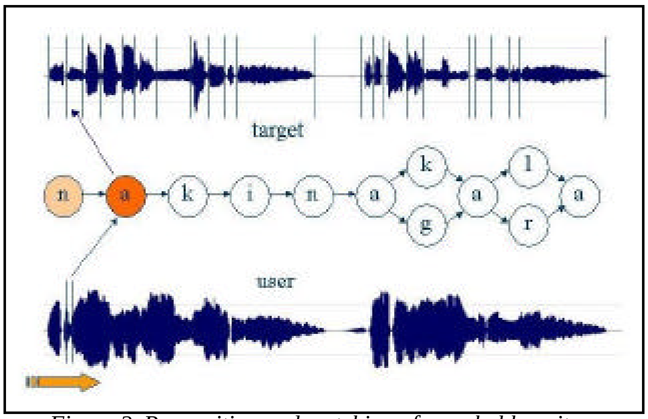


Figure 2. Recognition and matching of morphable units.

This would be a rather impossible/impractical problem if it was not for the fact that we know the words beforehand, the lyrics of the song. This reduces a big portion of the search problem: all the possible paths are restricted to just one string of phonemes, with several possible pronunciations. Then the problem reduces to locating the phoneme in the lyrics and placing the start and end points.

Besides knowing the lyrics, music information is also available. The user is singing along with the music and hopefully according to a tempo and melody already specified in the score of the song. We thus also know the time at which a phoneme is supposed to be sung, its approximate duration, its associated pitch, etc. All this information is used to improve the performance of the recognizer and also to allow resynchronization, for example in the case that the singer skips a part of the song.

We have incorporated a speech recognizer based on phoneme-base discrete HMM's that handles musical information and that is able to work with very low delay. The details of the recognition system can be found in another paper of our group (Loscos, Cano, Bonada, 1999).

The recognizer is also used in the preparation of the target audio data, to fragment the recording into morphable units (phonemes) and to label them with the phonetic transcription and the musical context. This is done out of real-time for a better performance.

5. Morphing

Depending on the phoneme the user is singing, a unit from the target is selected. Each frame from the user is morphed with a different frame from the target, advancing sequentially in time. Then the user has the choice to interpolate the different parameters extracted at the analysis stage, such as amplitude, fundamental frequency, spectral shape, residual signal, etc. In general the amplitude will not be interpolated, thus always using the amplitude from the user and the unvoiced phonemes will also not be morphed, thus always using the consonants from the user. This will give the user the feeling of being in control.

In most cases the durations of the user and target phonemes to be morphed will be different. If a given user's phoneme is shorter than the one from the target the system will simply skip the remaining part of the target phoneme and go directly to the articulation portion. In the case when the user sings a longer phoneme than the one present in the target data the system enters in the loop mode. Each voiced phoneme of the target has a loop point frame, marked in the preprocessing, non-real time stage. The system uses this frame to loop-synthesis in case the user sings beyond that point in the phoneme. Once we reach this frame in the target, the rest of the frames of the user will be interpolated with that same frame until the user ends the phoneme. This process is shown in Figure 3.

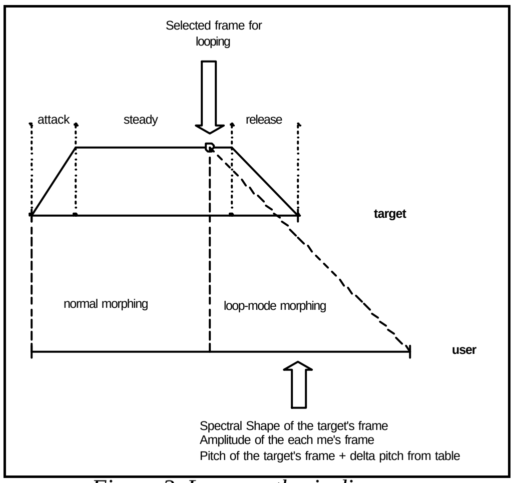

Selected frame for looping
attack
steady
release
target
normal morphing
loop-mode morphing
user
Spectral Shape of the target's frame
Amplitude of the each me's frame
Pitch of the target's frame + delta pitch from table

Figure 3. Loop synthesis diagram.

The frame used as a loop frame requires a good spectral shape and, if possible, a pitch very close to the note that corresponds to that phoneme. Since we keep a constant spectral shape, we have to do something to make the synthesis sound natural. The way we do it is by using some "natural" templates obtained from the analysis of a longer phoneme that are then used to generate more target frames to morph with out of the loop frame. One feature that adds naturalness is pitch variations of a steady state note sung by the same target. These delta pitches are kept in a look up table whose first access is random and then we just read consecutive values. We keep two tables, one with variations of steady pitch and another one with vibrato to generate target frames.

Once all the chosen parameters have been interpolated in a given frame they are added back to the basic SMS frame of the user. The synthesis is done with the standard synthesis procedures of SMS.

6. Experiments

The singing voice impersonator has been implemented on a PC platform (Cano, Loscos, Bonada, de Boer, Serra, 2000). To check the feasibility of the real-time technology presented, that is the SMS based morph engine and the recognizer, the target data used was a complete song, as shown in Figure 2, instead of a database of target excerpts. Thus the search for the most appropriate morphing frame pairs becomes a simple process.

Another simplification is that the system only morphs the voiced parts; the unvoiced consonants of the user are directly bypassed to the output. This is done because the morph engine deals better with voiced sounds and the results show that this restriction does not limit the quality of the impersonation. However, some audible artefacts may appear. One emerges from the fact the human voice organ produces all type of voice-unvoiced sounds and the pitch-unpitch boundaries are, in most cases, uncertain. This makes the system sometimes fails in the boundaries of unvoiced-voiced transitions. The other problem appears when the interpolation factor for the spectral shape parameter is set to be around 50%. Since the shapes are linearly interpolated, the morphed spectral shape is too smoothed and looses the timbre character of the original voices. This is currently being solved by working on a more complex model for the spectral shape that takes into account the formants to do the interpolation.

The HMM phonetic models were trained with a limited singing voice database. It is a fact that the recognition step works better when the user singer has been used to train the database. We believe that taking into account context and using non-discrete symbol probability distributions would bring better results but they require bigger databases.

The system as a whole produces quite high quality sound. The delay between the sound input and the final sound output in the running system is less than 30 milliseconds. This delay is just good enough to make the user have the feeling of being in control of the output sound.

7. Conclusions

In this paper we have presented a singing voice morphing system. Obviously, there is room for improvements in every single module of the system, but the techniques presented have proved to be valid and capable of musically useful results.

The final purpose of the system was to make an average singer sing any song like any professional singer. In fact, we would like the system to morph the user's voice with qualities of several singers, for instance a mixture timbre of *Sinatra* and *Tom Jones* and the horseness of Tom Waits. However, at this point, and due to the limitations of our system, we need a clear recording of *Tom Jones*, or whomever the user wants to impersonate, singing the song. It is not easy to have this kind of popular professional singer to record songs for us and so in this project, we used professional impersonators' recordings. However it is clear that is by no means efficient, not only because of technical issues like memory requirements, but also due to the cost of having professional singers recording every song of the system. To allow the user to sing any song with the voice and expression of whomever he wants without having a professional singer singing each song for each possible timbre, we will need a model for every desired target voice and also every type of singing style. In order to achieve this, techniques to perform the match of the phonemes, considering style and musical context, must be incorporated into the system. One approach for the case of the saxophone has been studied (Arcos, Lopez de Mantaras, Serra, 1997).

8. Acknowledgments

We would like to acknowledge the support from Yamaha Corporation and the contribution to this research of the other members of the Music Technology Group of the Audiovisual Institute.

References

Arcos, J. LL., R. Lopez de Mántaras, X. Serra. 1997. "Saxex: a Case-Based Reasoning System for Generating Expressive Musical Performances". *Proceedings of the ICMC 1997.*

Cano, P. 1998. "Fundamental Frequency Estimation in the SMS Analysis". *Proceedings of the Digital Audio Effects Workshop*, 1998.

Childers, D.G. 1994. "Measuring and Modeling Vocal Source-Tract Interaction". *IEEE Transactions on Biomedical Engineering 1994.*

Cano, P., A. Loscos. 1999. *Singing Voice Morphing System based on SMS. UPC*, 1999.

Cano, P., A. Loscos, J. Bonada, M. de Boer, X. Serra. 2000. "Singing Voice Impersonator Application for PC". *Proceedings of the ICMC 2000.*

Loscos, A., P. Cano, J. Bonada. 1999. "Low-Delay Singing Voice Alignment to text". *Proceedings of the ICMC* 1999.

Osaka, N. 1995. "Timbre Interpolation of sounds using a sinusoidal model", *Proceedings of the ICMC* 1995.

Serra, X. 1994. "Sound hybridization techniques based on a deterministic plus stochastic decomposition model", *Proceedings of the ICMC 1994.*

Serra, X. 1997. "Musical Sound Modeling with Sinusoids plus Noise". G. D. Poli and others (eds.), *Musical Signal Processing*, Swets & Zeitlinger Publishers, 1997.

Serra, X., J. Bonada. 1998. "Sound Transformations on the SMS High Level Attributes". *Proceedings of 98 Digital Audio Effects Workshop*, Barcelona 1998.

Settel, Z., C. Lippe. 1996. "Real-Time Audio Morphing", 7th International Symposium on Electronic Art, 1996.

Slaney, M., M. Covell, B. Lassiter. 1996. "Automatic audio morphing", *Proc. IEEE Int. Conf. Acosut. Speech Signal Process.* 2, 1001-1004 (1996).

Tellman, E., L. Haken, B. Holloway. 1995. "Timbre Morphing of Sounds with Unequal Number of Features", *J. Audio Eng. Soc.*, 43:9 1995.

Singing Voice Impersonator Application for PC

Maarten de Boer, Jordi Bonada, Pedro Cano, Alex Loscos, Xavier Serra
Audiovisual Institute, Pompeu Fabra University
Rambla 31, 08002 Barcelona, Spain
{mdeboer, jboni, pcano, aloscos, xserra}@iua.upf.es
http://www.iua.upf.es

[*Published in the Proceedings of the ICMC 2000*]

Abstract

This paper presents the implementation aspects of a real-time system for morphing two voices in the context of a karaoke application. It describes the software design and implementation under a PC platform and it discusses platform specific issues to attain the minimum system delay.

1. Introduction

Singing voice conversion can be defined as the conversion of the voice quality from one singer to another. The impersonator application we present in this paper makes use of a conversion that gives a specific individuality to the synthesized voice. The goal is to come up with a karaoke-type application for PC in which the user can sing like his/her favorite singers.

In order to incorporate to the user's voice the corresponding characteristics of the "target" voice, the "target" voice is recorded and analyzed beforehand. Then the system performs a real-time morphing between the user's voice and the pre-stored analysis of the target voice. The user is able to control the degree of morphing, thus being able to choose the level of "impersonation" that he/she wants to accomplish.

The application relies on two main algorithms that define and constrict the architecture: the Spectral Modeling Synthesis (SMS) (Serra, 1997) and a Speech Recognizer. This paper presents the practical implementation aspects of such an application. The theoretical background can be found in other papers (Cano, Loscos, 1999; Loscos, Cano, Bonada, 1999; Cano, Loscos, Bonada, de Boer, Serra, 2000).

2. Implementation

The application has been implemented mostly in ANSI C++, in order to be easily portable to different operating systems. For some time, development has been focusing on the Windows platform only, but now it compiles under Linux as well, thus adding a second development platform. One of the purposes was to compare latencies between Windows and Linux, using the same hardware, in non-real-time and real-time implementations. A flow diagram with the most important classes, structures, and procedures is shown in figure 1.

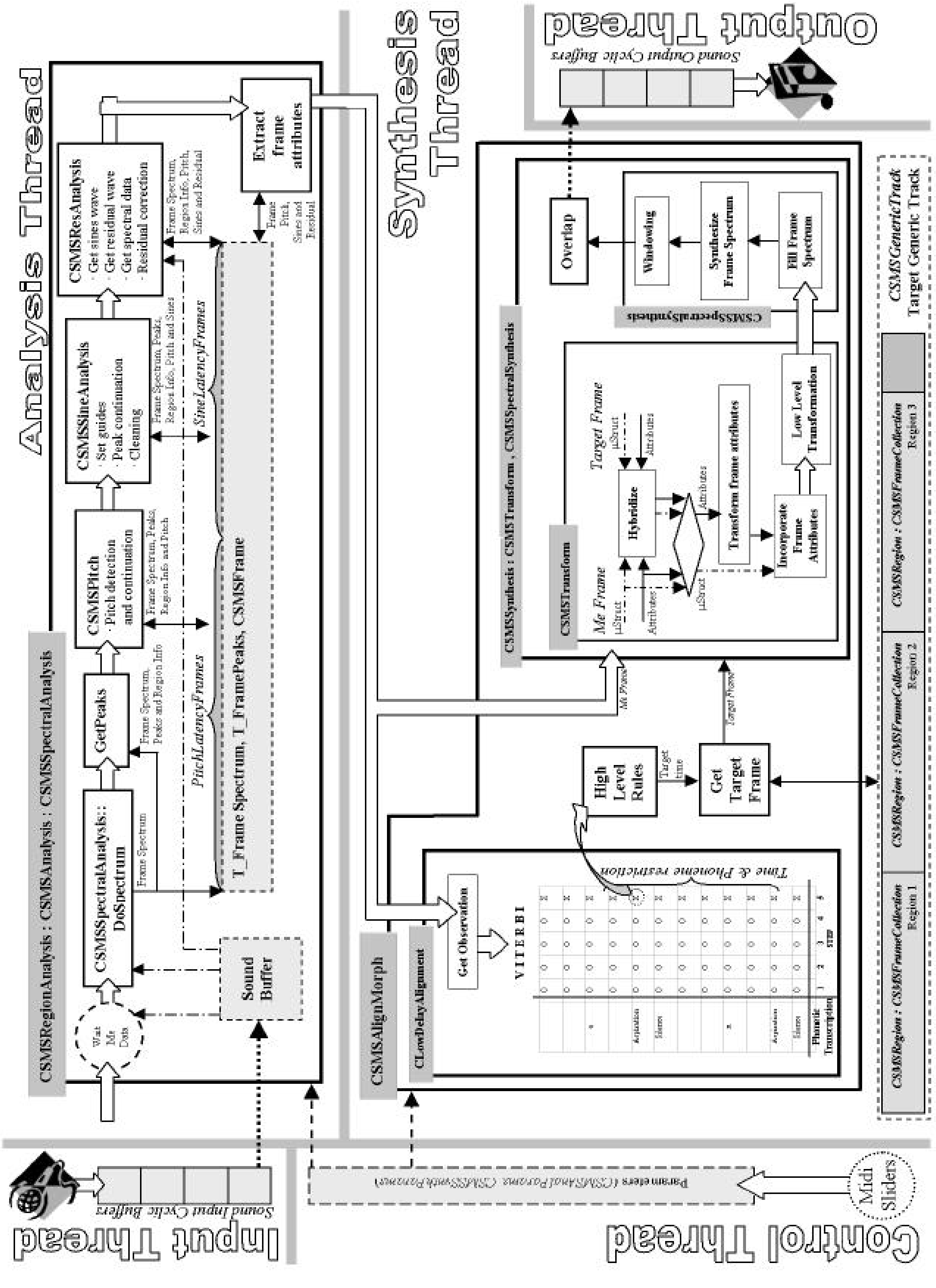
On both platforms the application uses multi-threading. The approach shown in figure 1 turned out to be the most efficient, in terms of latency, as well as the least sensitive for underruns. The Sound Thread makes use of a tight loop in which the audio is read and written from the soundcard. This ensures that when the processing does not keep up with the sound input, something that occasionally happens, sound throughput continues. Every read buffer of input samples is passed to the Analysis Thread, and when a new frame has been analyzed, the Synthesis Thread is signaled, which writes into the output buffer, that has been prepared for playback. The GUI is running in a third thread with low priority, and can modify the synthesis and analysis parameters used by the Analysis Thread and Synthesis Thread. Thread priority for the Sound Thread, Analysis Thread and Synthesis Thread are set to maximum, and memory pages are locked.

3. Graphical User Interface

The graphical interface of the application has been thought as a test platform from which all relevant parameters could be modified in real time. As shown in figure 2, the application has got six main sections. The analysis section controls some of the analysis parameters such as the lowest and highest pitch, the analysis window size or the residual model used.

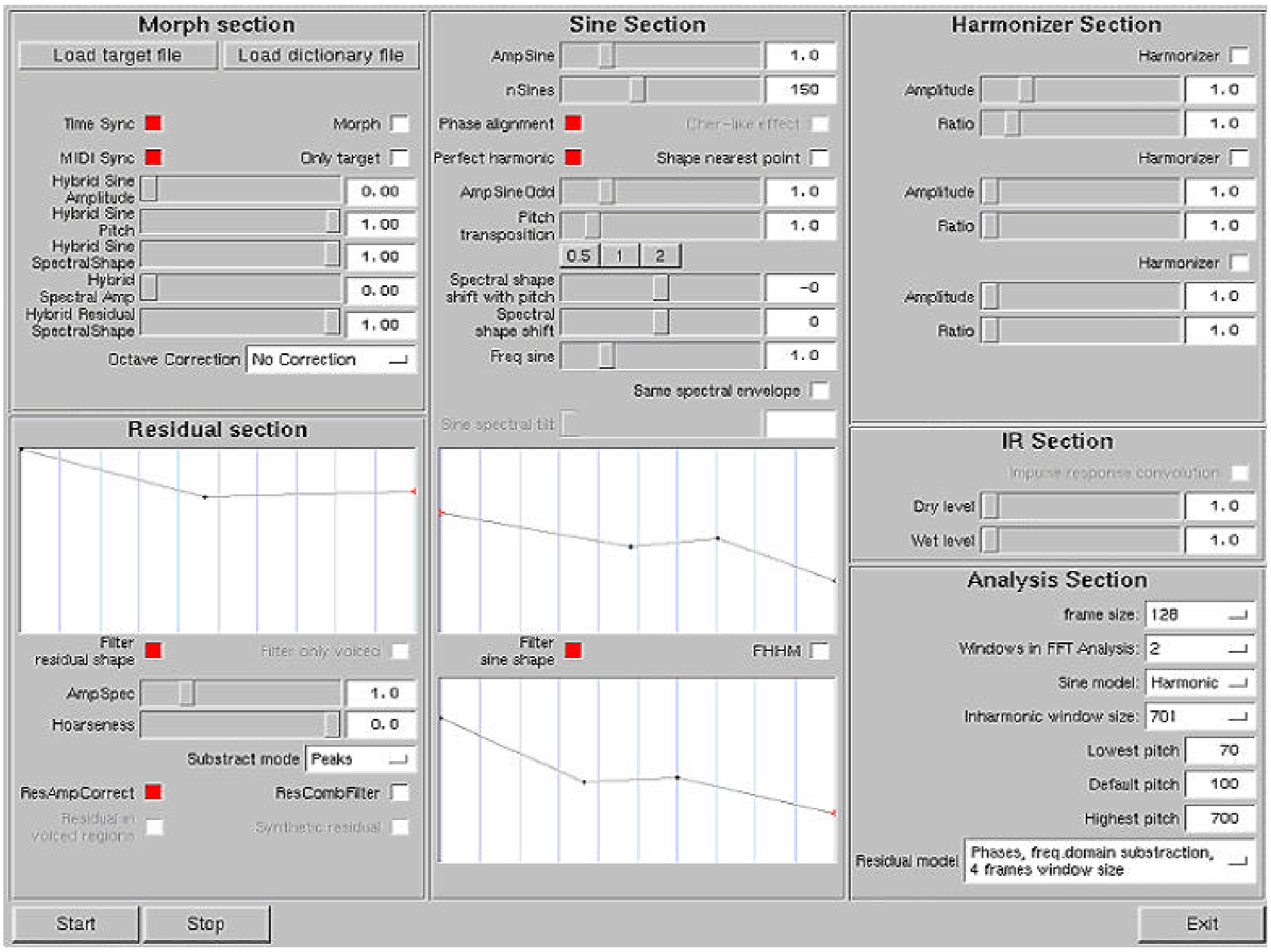
In the morph section five sliders control the interpolation values of the attributes involved the main part in the morph. These are the pitch, the amplitude of the sinusoidal part, the amplitude of the residual part, the spectral shape of the sinusoidal part, and the spectral shape of the residual part.

The sine and residual sections control the most important synthesis parameters of SMS and also include some graphic filtering capabilities. The two sections left, the harmonizer section and the IR section, were included to test some of the effects we were working on.



Input Thread
Sound Input Cyclic Buffers
Analysis Thread
CSMSRegionAnalysis : CSMSAnalysis : CSMSSpectralAnalysis
Wait Me Data
Sound Buffer
CSMSSpectralAnalysis::DoSpectrum
Frame Spectrum
GetPeaks
Frame Spectrum, Peaks and Region Info
CSMSPitch
· Pitch detection and continuation
Frame Spectrum, Peaks, Region Info and Pitch
CSMSSineAnalysis
· Set guides
· Peak continuation
· Cleaning
Frame Spectrum, Peaks, Region Info, Pitch and Sines
CSMSResAnalysis
· Get sines wave
· Get residual wave
· Get spectral data
· Residual correction
Frame Spectrum, Region Info, Pitch, Sines and Residual
Extract frame attributes
Frame Pitch, Sines and Residual
PitchLatencyFrames
SineLatencyFrames
T_Frame Spectrum, T_FramePeaks, CSMSFrame
Synthesis Thread
CSMSAlignMorph
CLowDelayAlignment
Get Observation
VITERBI
Phonetic Transcription
STEP
Time & Phoneme restriction
High Level Rules
Target time
Get Target Frame
Me Frame
Target Frame
CSMSSynthesis : CSMSTransform , CSMSSpectralSynthesis
CSMSTransform
Hybridize
µStruct
Attributes
Transform frame attributes
Incorporate Frame Attributes
Low Level Transformation
CSMSSpectralSynthesis
Fill Frame Spectrum
Synthesize Frame Spectrum
Windowing
Overlap
Output Thread
Sound Output Cyclic Buffers
CSMSGenericTrack
Target Generic Track
CSMSRegion : CSMSFrameCollection
Region 1
Region 2
Region 3
Control Thread
Midi Sliders
Parameters (CSMSAnalParams, CSMSSynthParams)

Figure 1. System architecture diagram.


Morph section
Load target file | Load dictionary file
Time Sync 🟥 | Morph ☐
MIDI Sync 🟥 | Only target ☐
Hybrid Sine Amplitude 0.00
Hybrid Sine Pitch 1.00
Hybrid Sine SpectralShape 1.00
Hybrid Spectral Amp 0.00
Hybrid Residual SpectralShape 1.00
Octave Correction: No Correction

Sine Section
AmpSine 1.0
nSines 150
Phase alignment 🟥 | Cher-like effect ☐
Perfect harmonic 🟥 | Shape nearest point ☐
AmpSineOdd 1.0
Pitch transposition 1.0
0.5 | 1 | 2
Spectral shape shift with pitch -0
Spectral shape shift 0
Freq sine 1.0
Same spectral envelope ☐
Sine spectral tilt
Filter sine shape 🟥 | FHHM ☐

Harmonizer Section
Harmonizer ☐
Amplitude 1.0
Ratio 1.0
Harmonizer ☐
Amplitude 1.0
Ratio 1.0
Harmonizer ☐
Amplitude 1.0
Ratio 1.0

Residual section
Filter residual shape 🟥 | Filter only voiced ☐
AmpSpec 1.0
Hoarseness 0.0
Substract mode: Peaks
ResAmpCorrect 🟥 | ResCombFilter ☐
Residual in voiced regions ☐ | Synthetic residual ☐

IR Section
Impulse response convolution ☐
Dry level 1.0
Wet level 1.0

Analysis Section
frame size: 128
Windows in FFT Analysis: 2
Sine model: Harmonic
Inharmonic window size: 701
Lowest pitch 70
Default pitch 100
Highest pitch 700
Residual model: Phases, freq.domain substraction, 4 frames window size

Start | Stop | Exit

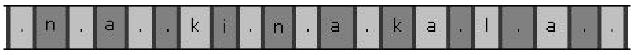
Figure 2. GUI screenshot.

4. Scoring of pitch and timing accuracy

We included in our application one of the features that many karaoke systems have; a score indicator to show how well the user is singing at each instant. To do so the system measures the pitch and timing of the user, comparing it with some reference values. The pitch and timing chosen as reference are not from the target (the target singer does not necessarily sing perfect) but the ones included in the song's MIDI representation.

In order to do the comparison, we have to know which is the note that the user is singing. The morph-alignment module tells us which region of the target the user is at. From there we should be able to find out the corresponding note information, by giving every region a note reference. Instead of doing this by hand, this is done automatically be adding the phonetic label found in each of the regions of the target to the MIDI file containing the correct melody.

Representation of the regions with the corresponding phonetic transcription (the dot means silence):



Note information from midi file:

id	begin (sec)	duration (sec)	pitch (Hz)
0	0.20	0.23	174.61
1	0.43	0.22	196.00
2	0.65	0.22	207.65
3	0.88	0.22	261.63
4	1.11	1.04	349.23

Add phonetic lyrics to the midi file:

id	begin (sec)	duration (sec)	pitch (Hz)	transcription
0	0.20	0.23	174.61	na
1	0.43	0.22	196.00	ki
2	0.65	0.22	207.65	na
3	0.88	0.22	261.63	ka
4	1.11	1.04	349.23	la

The automated mapping matches regions and id's in the following way:

.
n0
.
a0
.
.
k1
i1
.
n2
.
a2
.
k3
a3
.
l4
.
a4
.
.

From the analysis/synthesis loop, we obtain the user's fundamental frequency and time, and the target region used for the morph. This target region contains a reference to a note ("note id") with the pitch and timing information. A history of these note id's is kept, so when a certain number of the same note id's has been reached, we presume that this is the note the user is singing (current note id). We use the time the current note id changes, as the begin time of the user note, and compare it with the reference note, and can compare the pitch continuously. The scoring is then calculated from the pitch difference in semitones:

$$Dif_{semitones} = \left| \left(12\log\left(freq_{user} \big/ freq_{reference} \right) + 6 \right) \% 12 \right|$$

When the timing error is within a 0.1 second, maximum score is assigned. If the error is bigger than that, score goes from 1 (0.1 sec) to 0 (0.5 sec). A lowpass filter is applied on these scores to filter out sudden changes that are likely to be caused by erroneous analysis. Apart from this current score, also an overall score is kept.

We tried to improve the visual presentation of the scoring. The presentation with sliders is neither visually attractive nor inspiring for the singer. Because of that we implemented an animation-style scoring indicator, with seven different levels of scoring, each with a small variation to make it less static. The character in the animation is an animation-style Elvis-alike figure that takes the role of the singer as shown in figure 3. The character is sad/desperate when the user is singing badly and happy when the singer is doing well.

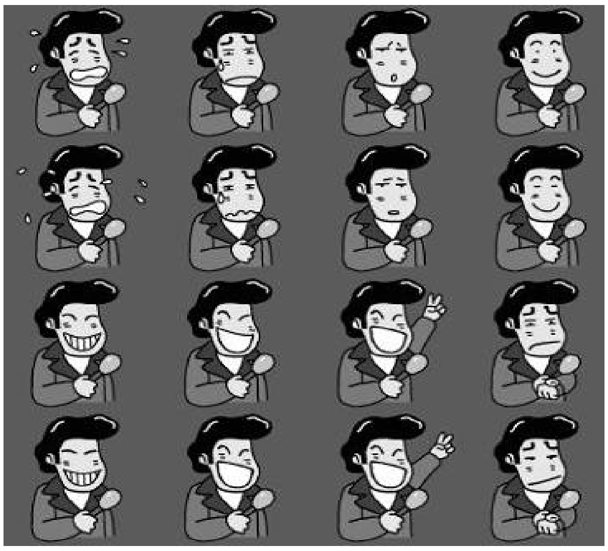


Figure 3. Animated-scoring indicator of the user's performance.

5. Conclusions

The running system has been tested with several songs and many users. Even though there are still many improvements to be made, both in the algorithms and in the implementation, the application has proved to be musically useful.

We have measured the hardware latency (with a SB AWE64 PCI), recording a stereo signal where one channel was the original sound, and the other channel was the sound through adc-dac read/write with small buffers (128 samples, 22100 kHz, 16 bit, mono). The difference was 6 ms, with constant throughput. To this, we have to add the latency that is inherent to the FFT's and pitch detection, which around 20 ms. Thus the delay between the sound input and the final sound output in the running system is less than 30 milliseconds. This delay is just good enough to make the user have the feeling of being in control of the output sound.

Some underrun problems occur by unpredictable performance problems, when the analysis or synthesis takes longer than expected. This might be because of memory allocation. Since the occurrence of these kinds of underruns is very rare, it is not a real problem for the current set up.

Currently, the scoring is only related to the pitch accuracy. We experienced that the timing accuracy is more difficult to translate to a significant scoring, at least to a continuously changing one. It is in some extend visualized with an extra image with the animation character looking at his watch.

Acknowledgments

We would like to acknowledge the support from Yamaha Corporation and the contribution to this research of the other members of the Music Technology Group of the Audiovisual Institute.

References

Cano, P., A. Loscos. 1999. *Singing Voice Morphing System based on SMS, UPC*, 1999

Cano, P., A. Loscos, J. Bonada, M. de Boer, X. Serra. 2000. "Singing Voice Impersonator Application for PC", *Proceedings of the ICMC* 2000.

Loscos, A., P. Cano, J. Bonada. 1999. "Low-Delay Singing Voice Alignment to text". *Proceedings of the ICMC* 1999.

Serra, X. 1997. "Musical Sound Modeling with Sinusoids plus Noise". G. D. Poli and others (eds.), *Musical Signal Processing*, Swets & Zeitlinger Publishers, 1997.

Audio Engineering Society

Convention Paper

Presented at the 111th Convention
2001 September 21–24 New York, NY, USA

Spectral Approach to the Modeling of the Singing Voice

Jordi Bonada, Alex Loscos, Pedro Cano, Xavier Serra
Audiovisual Institute, Pompeu Fabra University
Barcelona, Spain
{jordi.bonada, alex.loscos, pedro.cano, xavier.serra}@iua.upf.es
http://www.iua.upf.es/mtg

Hideki Kenmochi
Advanced System Development Center, YAMAHA Corporation
Hamamatsu, Japan
kenmochi@beat.yamaha.co.jp

ABSTRACT

In this paper we present two different approaches to the modeling of the singing voice. Each of these approaches has been thought to fit in the specific requirements of two applications. These are an automatic voice impersonator for karaoke systems and a singing voice synthesizer.

1. INTRODUCTION

Singing voice synthesis has been an active research field for almost fifty years [Cook, 1996]. Most of the systems developed until now do not provide enough quality or do not meet the practical requirements to have found real-world commercial applications. Anyhow, it seems that one of the main issues behind singing voice synthesis is to offer not only quality but flexible and musically meaningful control over the vocal sound. In that sense, we may think of applications where impossible singing voices can be synthesized or where existing voices can be enhanced.

In a broad sense, and according to whether the focus is put on the system or its output, synthesis models used in singing voice synthesis can be classified into two main groups: spectral models and physical models. Spectral models are based on perceptual mechanisms of the listener while physical models focus on modeling the production mechanisms of the original system. Any of these two models might be regarded as suitable depending on the specific requirements of the application or may even be combined for taking advantages of both approaches.

The main benefit of using Physical models is that the parameters used in the model are closely related to the ones a singer uses to control his/her own vocal system. As such, some knowledge of the real-world mechanism can be brought on the design. The model itself can provide intuitive parameters if it is constructed so that it sufficiently matches the physical system. Conversely, such a system usually has a large number of parameters and the mapping of those quite intuitive controls of the production mechanism to the final output of the model, and so to the listener's perceived quality, is not a trivial task.

Alternatively, spectral models are closely related to some aspects of the human perceptual mechanism. Changes in the parameters of a spectral model can be more easily mapped to a change of sensation in the listener. Yet parameter spaces yielded by these systems are not necessarily the most natural ones for manipulation.

On this context, in this article we introduce two different applications related to singing voice synthesis. First, in section 2, we introduce the basis of the Spectral Modeling Synthesis technique, which has inspired many of the models characteristics. We then introduce a singing voice impersonator application that is able to morph the user's voice with a professional singer's version in real-time. We finish by outlining, in section 4, the basic approach towards a singing voice synthesizer that is currently being developed by our team.

2. SPECTRAL MODELING SYNTHESIS

The Spectral Modeling Synthesis (SMS) is a synthesis by analysis technique based on modeling the sounds as stable sinusoids (partials) plus noise (residual component). This sinusoidal plus residual model can be seen as a generalization of the *STFT* and the S*inusoidal* representations as we can decide what part of the sound to model as sinusoidal and what part to leave as STFT. [Serra 1996; Serra and Smith 1990].

The input sound *s(t)* is modeled by,

$$s(t) = \sum_{r=1}^{R} A_r(t)\cos\left[q_r(t)\right] + e(t) \qquad (1)$$

where $A_r(t)$ and $q_r(t)$ are the instantaneous amplitude and phase of the r^{th} sinusoid, respectively, and *e(t)* is time-varying noise component.

This estimation of the sinusoidal component is generally done by first computing the STFT of the sound, then detecting the spectral peaks (and measuring the magnitude, frequency and phase of each one), and organizing them as time-varying sinusoidal tracks. By using the fundamental frequency information in the peak continuation algorithm, we can identify the harmonic partials.

The sinusoidal plus residual model assumes that the sinusoids are stable partials of the sound with a slowly changing amplitude and frequency. With this restriction, we are able to add major constraints to the detection of sinusoids in the spectrum and omit the detection of the phase of each peak. The instantaneous phase that appears in the equation is taken to be the integral of the instantaneous frequency $w_r(t)$, and therefore satisfies

$$q_r(t) = \int_o^t w_r(t)\,dt \qquad (2)$$

where *w(t)* is the frequency in radians, and *r* is the sinusoid number. When the sinusoids are used to model only the stable partials of the sound, we refer to this part of the sound as the deterministic component.

The residual component is obtained by first generating the sinusoidal component with additive synthesis, and then subtracting it from the original waveform. This is possible because the instantaneous phases of the original sound are matched and therefore the shape of the time domain waveform preserved. A spectral analysis of this time domain residual is done by first windowing it, window which is independent of the one used to find sinusoids, and thus we are free to choose a different time-frequency compromise. Finally, the FFT is computed.

Within this model we can either leave the residual signal, *e(t)*, to be the difference between the original sound and the sinusoidal component, resulting into an identity system, or we can assume that *e(t)* is a stochastic signal. In this case, the residual can be described as filtered white noise,

$$e(t) = \int_0^t h(t,t)\,u(t)\,dt \qquad (3)$$

where *u(t)* is white noise and *h(t,t)* is the response of a time varying filter to an impulse at time *t*. That is, the residual is modeled by the time-domain convolution of white noise with a time-varying frequency-shaping filter.

The calculation of the residual component can be optimized by subtracting the sinusoids directly in frequency domain (see figure 1). This can be done subtracting the spectrum resulting of the convolution of each sinusoid with the transform of the same window used in the residual spectral analysis.

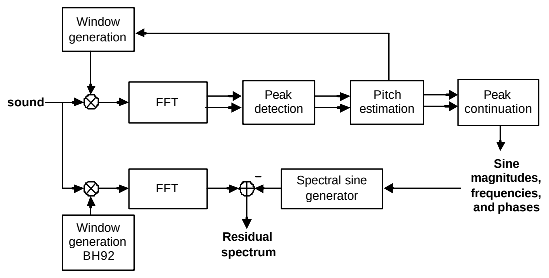


Fig 1 *Block diagram of the SMS analysis*

From the output of the analysis techniques presented we can obtain several features of the input sound and its frequency-domain representation. These features are then used in the transformation block in order to modify the characteristics of the sound in a meaningful way. Transformations as the ones used in the applications here presented (such as morphing, pitch shifting, spectral shape modification...) can be performed using this approach. All these transformations can be done in the frequency domain. Afterwards, the output sound can be synthesized.

The sinusoidal component is generated using some type of additive synthesis approach and the residual, if present, is synthesized using a subtractive synthesis approach using an IFFT approach, efficient implementations may be provided. Figure 2 shows a block diagram of the final part of the synthesis process.

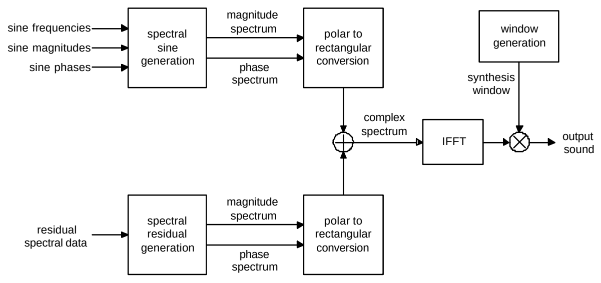


Fig 2 *Block diagram of the SMS synthesis*

Several modifications have been done to the basic SMS procedures to adapt them to the requirements of the applications outlined in this article. The major changes include the real-time implementation of the whole analysis/synthesis process with a processing latency of less than 30 milliseconds and the tuning of all parameters to the particular case of the singing voice. The later ones include the extraction of the most appropiate higher-level parameters for the case of the singing voice.

3 THE SINGING VOICE IMPERSONATOR

Morphing is a technique with which, out of two or more elements, we can generate new ones with hybrid properties. With different names, and using different signal processing techniques, the idea of audio morphing is well known in the Computer Music community [Serra, 1994; Tellman, Haken, Holloway, 1995; Osaka, 1995; Slaney, Covell, Lassiter. 1996; Settel, Lippe, 1996]. In most of these techniques, morph is based on the interpolation of sound parameterizations resulting from analysis/synthesis techniques, such as the Short-time Fourier Transform (STFT), Linear Predictive Coding (LPC) or Sinusoidal Models.

3.1 The application

The application we present here is a very particular case of audio morphing. What we want is to be able to morph, in real-time, two singing voice signals in order to control the resulting synthetic voice by mixing the characteristics of the two sources. In such a context, a karaoke-type application was developed in which the user can sing like his/her favorite singers [Cano, Loscos, Bonada, Boer, Serra, 2000; Boer, Bonada, Cano, Loscos, Serra, 2000]. The result is an automatic impersonating system that allows the user to morph his/her voice attributes (such as pitch, timbre, vibrato and articulations) with the ones from a prerecorded singer, which from now on we will refer to as *target*.

In this particular implementation, the target's performance of the complete song to be morphed is recorded and analyzed beforehand. In order to incorporate the corresponding characteristics of the target's voice to the user's voice, the system first recognizes what the user is singing (phonemes and notes), looks for the same sounds in the target performance (i.e. synchronizing the sounds), interpolates the selected voice attributes, and synthesizes the output morphed voice. All this is accomplished in real-time.

Figure 3 shows the general block diagram of the voice impersonator system. The system relies on two main techniques that define and constrict the architecture: the SMS framework and a Hidden Markov Model based Automatic Speech Recognizer (ASR). The SMS implementation is responsible of providing a suitable parameterization of the singing voice in order to perform the morph in a flexible and musically meaningful way. On the other hand, the ASR is responsible for aligning the singing voice of the user with that of the target.

Before we can morph a particular song, we have to supply information about the song to be morphed and the song recording itself (Target Information and Song Information). The system requires the phonetic transcription of the lyrics, the melody as MIDI data, and the actual recording to be used as the target audio data. Thus, a good impersonator of the singer that originally sang the song has to be recorded. This recording has to be analyzed with

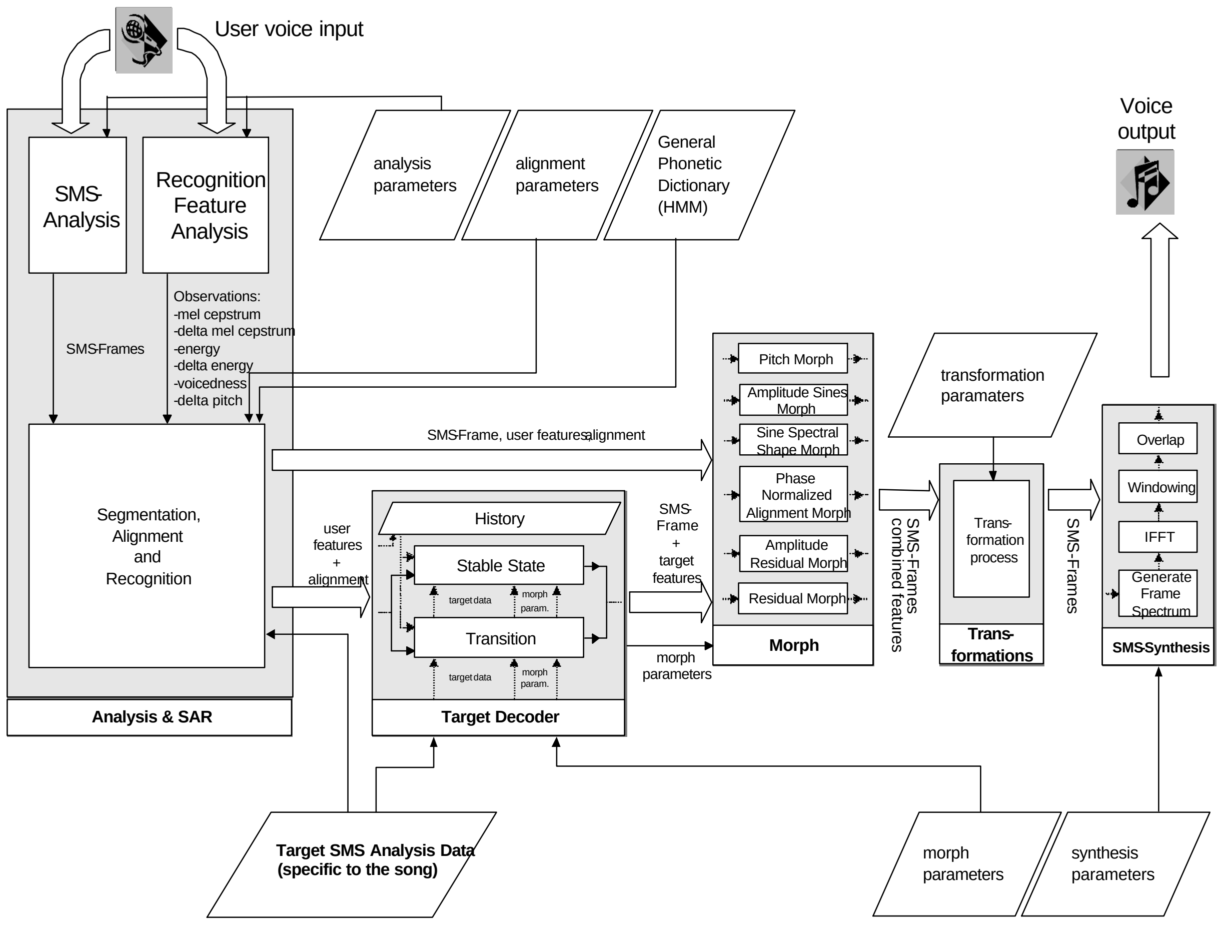


Fig 3 *Block diagram of the singing voice impersonator*

SMS, segmented into morphing units (phonemes), and each unit has to be labeled with the appropriate note and phonetic information of the song. This preparation stage is done semi-automatically, using a non-real time application developed for this purpose.

Once we have all the required inputs set we can start processing the user's voice. The first module of the running system includes the real-time analysis and the recognition/alignment steps. Each analysis frame, with the appropriate parameterization, is associated with the phoneme of a specific moment of the song and thus with a target frame. Once a user frame is matched to a target frame, we morph them interpolating data from both frames and we synthesize the output sound. Only voiced phonemes are morphed and the user has control over which and by how much each parameter is interpolated. The frames belonging to unvoiced phonemes are left untouched, thus always having the user's unvoiced consonants in the output.

Therefore, depending on the phoneme the user is singing, a unit from the target is selected and then each frame from the user is morphed with a different frame from the target, advancing sequentially in time as illustrated in figure 4. The user has the choice to interpolate the different parameters extracted at the analysis stage, such as amplitude, fundamental frequency, spectral shape, residual signal, etc. In general, the amplitude will not be interpolated, always using the amplitude from the user. The unvoiced phonemes will not be morphed either, so we will have always the user's. This will give the user the feeling of being in control of the synthesis.

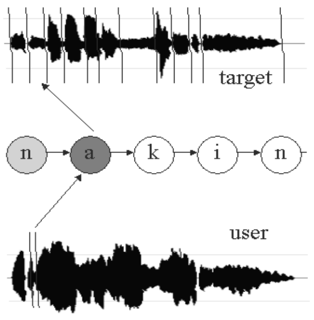


Fig 4 *Recognition and matching of morphable units*

In most cases, the durations of the user and target phonemes to be morphed will be different. If a given user's phoneme is shorter than the one from the target, the system will simply skip the remaining part of the target phoneme and go directly to the articulation portion. In the case when the user sings a longer phoneme than the one present in the target data, the system enters in the loop mode. Each voiced phoneme of the target has a loop point frame, marked in the preprocessing, non-real time stage. The system uses this frame to loop-synthesis in case the user sings beyond that point in the phoneme. Once we reach this frame in the target, the rest of the frames of the user will be interpolated with that same frame until the user ends the phoneme. In these cases, in order to avoid unnaturalness, we apply pitch templates obtained from longer utterances to the last frame of the target. This process is illustrated in figure 5.

Once all the chosen parameters have been interpolated for a given frame, they are added back to the basic SMS synthesis frame. Synthesis is done with the standard synthesis procedures of SMS.

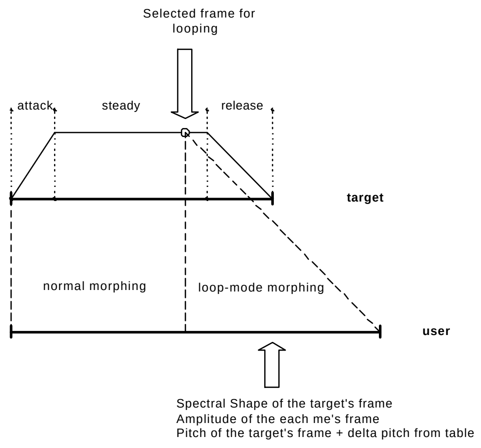

Selected frame for looping
attack
steady
release
target
normal morphing
loop-mode morphing
user
Spectral Shape of the target's frame
Amplitude of the each me's frame
Pitch of the target's frame + delta pitch from table

Fig 5 *Synthesis loop diagram*

3.2 The aligner

To solve the matching problem the system includes an ASR based on phoneme-base discrete HMM's. This ASR has been trained using a 22 minutes long Japanese singing database and the following front-end parameterization:

Mel Cepstrum	12 coefficients
Delta Mel Cepstrum	12 coefficients
Energy	1 coefficient
Delta Energy	1 coefficient
Voiceness	2 coefficients

The voiceness vector consists of a Pitch Error measure and the Zero Crossing rate. The Pitch Error component is a by-product from the fundamental frequency analysis, which is based on [Cano, 98]. The zero crossing rate is calculated by dividing the number of consecutive samples with different signs by the number of samples of the frame. The differentials (deltas) ponder up to two frames of the future and two frames of the past.

The alignment process starts with the generation of a phonetic transcription out of the lyrics text. This phonetic transcription is used to build the composite song Finite State Network (FSN) concatenating the models of the phonemes transcribed.

The phonetic transcription previous to the alignment process has to be flexible and general enough to account for all the possible realizations of the singer. It is very important to bear in mind the non-linguistic units silence and aspiration, as their appearance cannot be predicted. Different singers place silences and aspirations in different places. This is why, while building the FSN, between each pair of phoneme models, we insert both silence and aspiration models. In the transition probability matrix of the FSN, the jump probability a_{ij} from each speech phonetic unit to the next silence, aspiration or speech phonetic unit will be the same, as shown in figure 6.

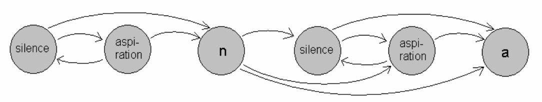


Fig 6 *Concatenation of silences and aspirations in the FSN*

The aspiration is problematic since in singing performance its dynamics are more significant. This causes the aspiration to be easily confused with a fricative. Moreover, different singers not only sing different but also, as in speech, pronounce different. To take into account these different pronunciations we modify the FSN to add parallel paths as shown in figure 7.

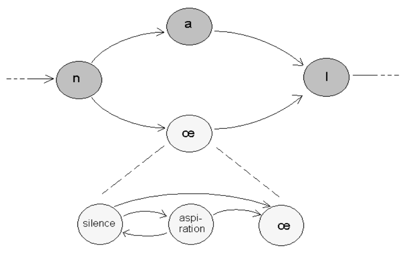


Fig 7 *Representation of a phonetic equivalence in the FSN*

The alignment resulting from the Viterbi decoding will follow the most likely path, so it will decide whether it is more probable that phoneme [a] or phoneme [œ] was sung.

The ASR has been adapted to handle musical information and works with very low delay [Loscos, Cano, Bonada, 1999] since the system cannot wait for a phoneme to be finished before it is recognized. Moreover, a phoneme has to be assigned to each frame. This would be a rather impossible/impractical situation if the lyrics of the song were not known beforehand. This constraint reduces the search problem: all the possible paths are restricted to just one string of phonemes, with several possible pronunciations. The problem is cut down to the question of locating the phoneme in the lyrics and placing the start and end points.

Besides knowing the lyrics, music information is also available. The user is singing along with the music, and hopefully according to a tempo and melody already specified in the score. Thus, we also know the time at which a phoneme is supposed to be sung, its approximate duration, its associated pitch, etc. All this information is used to improve the performance of the recognizer and also to allow resynchronization, for example in the case that the singer skipped a part of the song. The tempo information, for example, is used to modify the output Viterbi probabilities by the function shown in figure 8.

t_s is the time at which the phoneme happens in the singer performance, t_m is the time at which this phoneme happens in the song score, parameters *a* and *b* are tempo and duration dependent, and *alpha* has a very low value, nearly null. This function can be defined differently for the case in which the singer comes from silence and attacks the beginning of a note/word, and for the case where the singer has already started a note/word, due to the very different behaviors of these two situations.

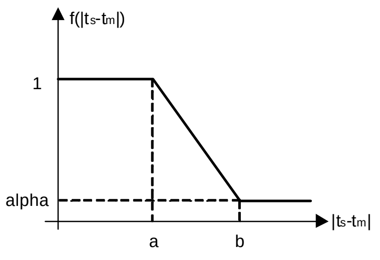


Fig 8 *Function of the factor applied to the Viterbi probability*

3.3 The impersonator singing voice model

The basic SMS analysis results in a simple parameterization appropriate for describing the inner properties of a sound, namely the instantaneous frequency, the amplitude and phase of each partial and the instantaneous spectral characteristics of the residual signal. Still, there are other useful instantaneous attributes that give a higher-level abstraction of the sound characteristics. These attributes are calculated at each analysis frame from the output of the basic sinusoidal plus residual analysis. [Serra, Bonada, 98]

The attributes we use for this application are: the amplitude of sinusoidal component, the amplitude of residual component, the fundamental frequency, the spectral shape of sinusoidal component, and the spectral shape of the residual component. These attributes are extracted from the frame data, leaving a normalized frame. This way, we can morph by interpolating the high-level attributes and later add these back to the synthesis frame. By carrying out the morph at the high-level plane, we have a more intuitive and musical control of the process and we minimize crossed effects between interpolations of different attributes.

The amplitude of the sinusoidal component is calculated as the sum of the amplitudes of all harmonics of the current frame expressed in dB,

$$AS_{total} = 20\log_{10}\left(\sum_{i=1}^{I} a_i\right) \qquad (4)$$

where a_i is the linear amplitude of the i^{th} harmonic and I is the total number of harmonics found in the current frame.

The amplitude of the residual component is calculated as the sum of the absolute values of the residual of the current frame expressed in dB. This amplitude can also be computed by adding the frequency samples of the corresponding magnitude spectrum,

$$\begin{aligned} AR_{total} &= 20\log_{10}\left(\sum_{n=0}^{M-1}\left|x_R(n)\right|\right) \\ &= 20\log_{10}\left(\sum_{k=0}^{N-1}\left|X_R(k)\right|\right) \end{aligned} \qquad (5)$$

where $x_R(n)$ is the residual sound, M is the size of the frame, $x_R(k)$ is the spectrum of the residual sound, and N is the size of the magnitude spectrum.

The fundamental frequency is defined as the frequency that best explains the harmonics of the current frame. This can be computed by taking the weighted average of all the normalized harmonic frequencies,

$$F_0 = \sum_{i=1}^{I} \frac{f_i}{i} \cdot \frac{a_i}{\sum_{i=1}^{I} a_i} \qquad (6)$$

where f_i is the frequency of the i^{th} harmonic. A more complete discussion on the issue of fundamental frequency in the context of SMS can be found in [Cano, 98].

The spectral shape of the sinusoidal component is expressed as an envelope described by the amplitudes and frequencies of the harmonics, or its approximation,

$$SShape = \{(f_1, a_1)(f_2, a_2)\ldots(f_I, a_I)\} \qquad (7)$$

This set of points that define the spectral shape envelope is joined with a third order spline interpolation instead of linear interpolation. Spline interpolation gives a better approximation of the resonant character of the spectrum of the singing voice.

For the sinusoidal component, we also store the phase envelope at the beginning of each period. This envelope is computed applying a time shift depending of the pitch found at each frame. This phase envelope is the one responsible of preserving the phase alignment.

However, whenever the spectral shape is interpolated, and the morph factor is set around 50%, the resulting spectral shape is smoothed and looses much of its timbre characteristics. This problem can be solved if we include anchor points (i.e. resonances) in the spectral shape model and we take them into account in the interpolation step as shown in figure 9.

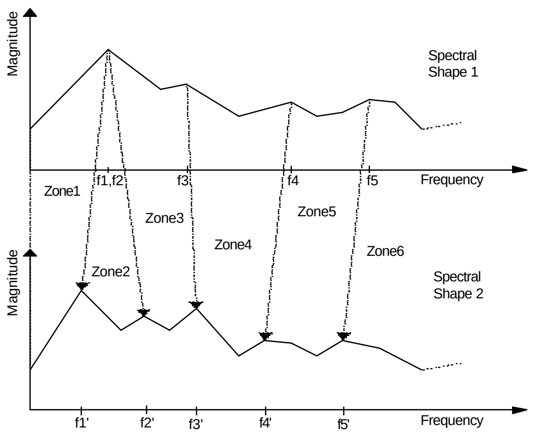


Fig 9 *Anchor points in the sinusoidal spectral shapes*

By using anchor points we can preserve the maximums and the valleys of the spectrum so that when going from one timbre to another, resonances would not appear and disappear but move from one frequency to another.

The spectral shape of the residual component is expressed as an approximation of the magnitude spectrum of the residual sound at the current frame. A simple function is computed as the line segment approximation of the spectrum,

$$Rshape = \left\{ e_1, e_2, \dots, e_q, \dots, e_{N/M} \right\} = \max_k \left[\left| X_R \left(qM + k \right) \right| \right] \quad (8)$$

where $k=-M/2,-M/2+1,\dots,\dots,M/2-1$, and M is the number of frequency samples used for each calculation of a local maximum.

For both the sinusoidal and residual components the magnitudes of the spectral shape envelope are interpolated whereas the phases are always taken either from the user or the target but never interpolated.

3.4 Discussion

In this section we have presented a singing voice morphing system. The final purpose of the system was to make an average singer sing any song like any professional singer. However the system has some weaknesses that arise from its specifications and disallow its practical commercialization.

At this point of time, in order for a user to sing a song like somebody else's voice, we need to get before hand a dry recording of the target singer singing the selected song. Since it is not easy to afford dry recordings of the most probable candidates to be chosen as targets (i.e. Frank Sinatra or Aretha Franklin), in this project we have used recordings of professional impersonators. Anyhow, the requirement pointed out here makes the system by no means efficient.

The inefficiency comes from two main intractable hitches. First, it entails an unreachable cost to have one popular singers or even his impersonator recording in a studio all the songs available in a karaoke database. Second, we need to analyze and store before hand the targets performances and this process generates such a huge amount of data it turns the database into something impossible to handle.

There are many solutions that can be thought to solve these problems. For example, we could think of a system in which the user could only sing Elvis's songs like Elvis, and would not allow possibilities such as singing "Walk like an Egyptian" like John Lennon. This would not only reduce the recording requirements but also would open the possibility of coming to an agreement with some record labels in order to get dry recordings of their most well-known singers. Also all sorts of data-compression techniques could be applied to the system to reduce the amount of data stored in the database.

Anyway, our future plan does not consider any of the mentioned propositions to be the one to concentrate on. We believe the solution implies accomplishing a more flexible system in which we would work with singer models rather than singer performances. The idea is to record the target singer singing all possible phonetics at different registers, with different intensities, in different emotion contexts, etcetera, sampling this way the space of all the possible phonetic and musical contexts. The analysis of these recordings would be the basis of a singer model from which we could later synthesize, out of the score and the lyrics of a song, a possible performance of the singer modeled. That is what brought us to the singing synthesizer application.

4. THE SINGING VOICE SYNTHESIZER

We have developed a source-filter type singing synthesizer application based on the sinusoidal plus residual decomposition of the sound. This system generates a performance of an artificial singer out of the musical score and the phonetic transcription of a song.

Mimicking the performance of a singer using computer synthesis tools is a very ambitious and complex goal and there are many parameters at different levels that affect the naturalness of the resulting synthesized sound. The system we present takes care of it at two main levels: the expressiveness level and the synthesis level. The expressiveness level is the one in charge of reproducing the musical and emotional expressions. The synthesis level is in charge of reproducing the generation of a human singing voice signal. This synthesis level is the one we present next.

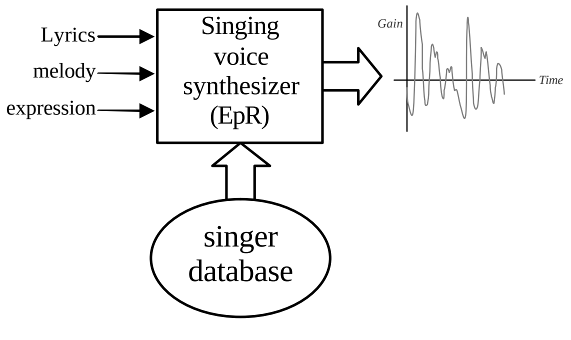


Fig 10 *Frequency domain implementation of the EpR model*

In figure 10 we can see the general diagram of our singing voice synthesizer. The inputs of the system are the lyrics (the phonetic transcription), the melody to be sung and optionally some expression parameters. On the other hand, the system is also feed by a singer database. This database holds the voice characteristics and is created from the analysis of singer's recordings.

4.1 The EpR voice model

Our singing voice synthesizer is based on an extension of the well known source/filter approach [Childers, 94] we call EpR (***E**xcitation **p**lus **R**esonances).* The excitation can be either voiced, unvoiced, or a combination of both. Besides, in the case of a voiced phonation we model a harmonic source plus a residual source. In figure 11 we can see a graphic with the three types of excitation generated in our system and the corresponding filters. For the case of a voiced phonation, the filter applied to each excitation tries to mimic the singing voice spectral shape using a frequency domain filtering function that can be decomposed into two filters in cascade: an exponential decay curve plus several resonances. After filtering, the voiced residual excitation needs to be transposed because it is a filtered SMS residual recording and

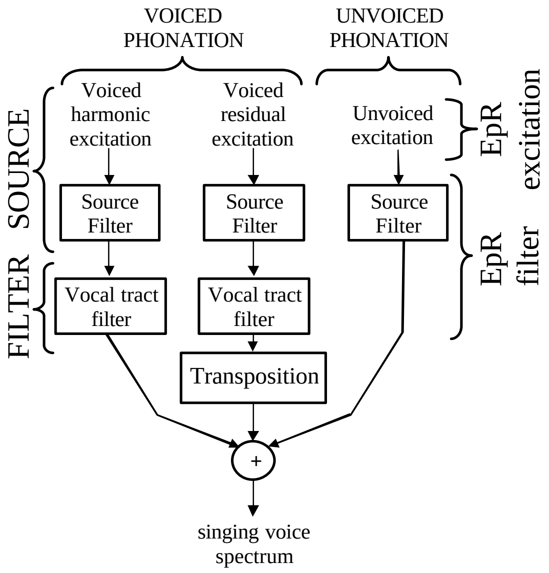


Fig 11 *The EpR voice model*

has traces of the original pitch. Otherwise, in the case of an unvoiced phonation, we apply a filter that just changes the tilt curve and the gain of the STFT of an original recording.

4.1.1 The EpR excitation

Voiced harmonic excitation

The inputs that control the voiced harmonic excitation generation are the desired pitch and gain envelopes. The resulting excitation signal is obtained by generating a delta train in the time domain thus allowing to achieve period resolution and to use some simple excitation templates. This can be useful to generate jitter or different types of vocal disorders. This delta train can be seen as a glottal source wave previous to a convolution with the differentiated glottal pulse.

A fractional delay filter is needed to position the excitation deltas between samples, since we have to go beyond the sampling rate resolution. The filter is implemented using a windowed sinc like function situated at each pulse location with the offset subtracted [Smith, Gosset, 1984]. Finally the windowing and the fft are applied. The result is a spectrum approximately flat that contains the harmonics approximately synchronized in phase. If no excitation template is applied, the spectrum will be perfectly flat and the phase synchronization precise.

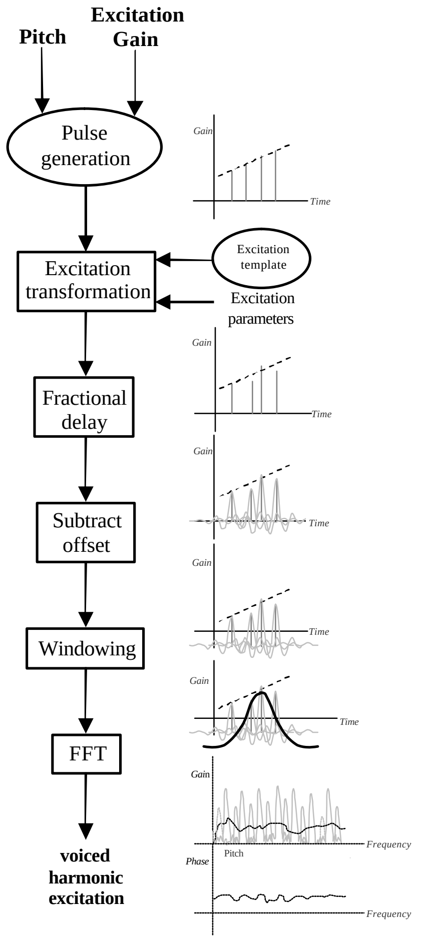


Fig 12 *The EpR voiced harmonic excitation*

Voiced residual excitation

The voiced residual excitation is obtained from the residual of the SMS analysis of a long steady state vowel recorded from a real singer. The SMS residual is inverse-filtered by its short-time average spectral shape envelope to get an approximately flat excitation magnitude spectrum.

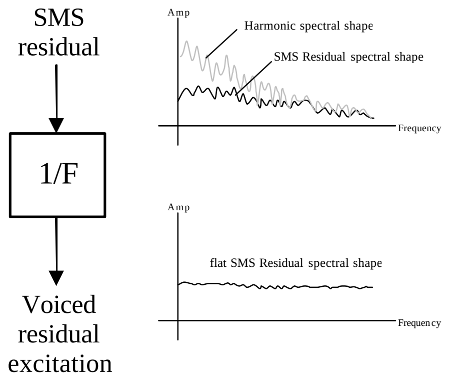

SMS residual
Amp
Harmonic spectral shape
SMS Residual spectral shape
Frequency
1/F
flat SMS Residual spectral shape
Voiced residual excitation

Fig 13 *The voiced residual excitation*

Unvoiced excitation

The excitation in the unvoiced parts is left unmodeled, using directly the original recording of a singer's performance.

4.1.2 The EpR filter

The EpR filter can be decomposed in two cascade filters. The first of them models the differentiated glottal pulse frequency response, and the second the vocal tract (resonance filter).

The EpR source filter

The EpR source is modeled as a frequency domain curve and one source resonance applied to the input frequency domain flat excitation described in the previous section. This source curve is defined by a gain and an exponential decay as follows:

$$Source_{dB} = Gain_{dB} + SlopeDepth_{dB}\left(e^{Slope\cdot f} \quad 1\right) \qquad (9)$$

This curve is obtained from an approximation to the harmonic spectral shape (*HSS*) determined by the harmonics identified in the SMS analysis

$$HSS\left(f\right) = envelope_{i=0..n}\left[f_i, 20\log\left(a_i\right)\right] \qquad (10)$$

where i is the index of the harmonic, n is the number of harmonics, f_i and a_i are the frequency and amplitude of the i^{th} harmonic.

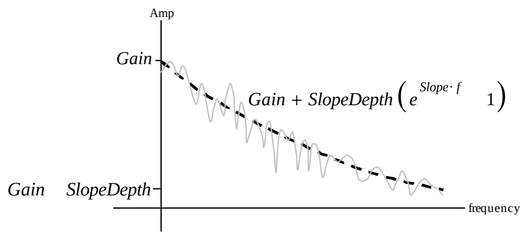


Fig 14 *The EpR source curve*

On top of the source curve, we add a second resonance in order to model the low frequency content of the spectrum below the first formant. This resonance affects the synthesis in a different way than the vocal tract resonances, as will be explained later.

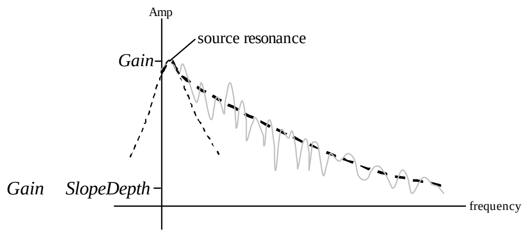


Fig 15 *The EpR source resonance*

The source resonance is modeled as a symmetric second order filter (based on the Klatt formant synthesizer [Klatt, 1980]) with center frequency *F*, bandwidth *Bw* and linear amplitude *Amp*. The transfer function of the resonance *R(f)* can be expressed as follows

$$H(z) = \frac{A}{1 \quad Bz^{\ 1} \quad Cz^{\ 2}}$$

$$R\left(f\right) = Amp\,\frac{H\left(e^{j2p\left(0.5+\frac{f\ \ F}{fs}\right)}\right)}{H\left(e^{jp}\right)} \qquad (11)$$

where

$$fs = \text{Sampling rate}$$

$$C = \ \ e^{\frac{2p\,Bw}{fs}}$$

$$B = 2\cos(p)e^{\frac{p\,Bw}{fs}}$$

$$A = 1 \quad B \quad C$$

The amplitude parameter (*Amp*) is relative to the source curve (a value of 1 means the resonance maximum is just over the source curve).

The EpR vocal tract filter

The vocal tract is modeled by a vector of resonances plus a differential spectral shape envelope. It can be understood as an approximation to the vocal tract filter. These filter resonances are modeled in the same way as the source resonance (see eq. 11), where the lower frequency resonances are somewhat equivalent to the vocal tract formants.

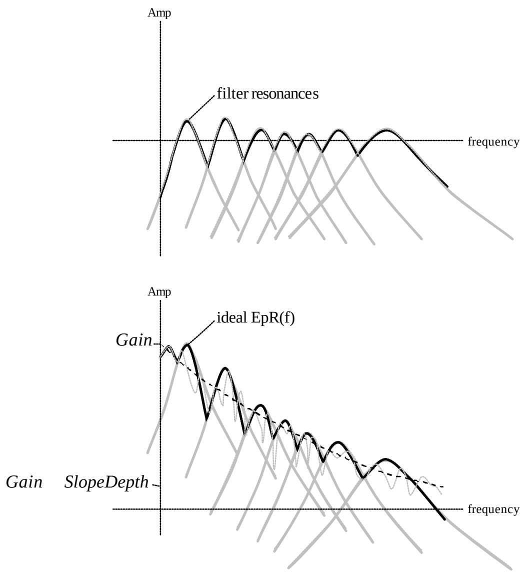


Fig 16 *The EpR filter resonances*

The EpR filters for voiced harmonic and residual excitations are basically the same, but just differ in the gain and slope depth parameters. This approximation has been obtained after comparing the harmonic and residual spectral shape of several SMS analysis of singer recordings. Figure 17 shows these differences.

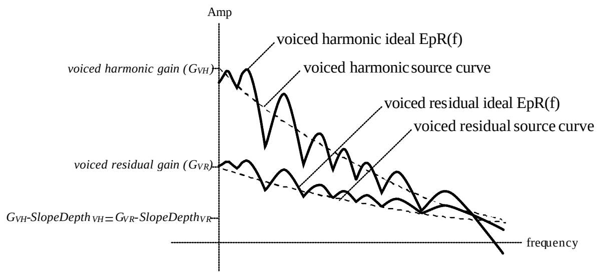


Fig 17 *Differences between harmonic and residual EpR filters*

The differential spectral shape envelope actually stores the differences (in dB) between the ideal EpR model (*iEpR*) and the real harmonic spectral shape (*HSS*) of a singer's performance. We calculate it as a 30 Hz equidistant step envelope.

$$DSS\left(f\right) = envelope_{i=0..}\left[30i, HSS_{dB}(30i) \quad iEpR_{dB}(30i)\right] \quad (12)$$

The EpR phase alignment

The phase alignment of the harmonics at the beginning of each period is obtained from the EpR spectral phase envelope. A time shift is applied just before the synthesis, in order to get the actual phase envelope at the synthesis time (usually it will not match the beginning of the period). This phase alignment is then added to the voiced harmonic excitation spectrum phase envelope. The EpR spectral phase model states that each vocal tract resonance produces a linear shift of p on the flat phase envelope with a bandwidth depending on the estimated resonance bandwidth. This phase model is especially important for the intelligibility and in order to get more natural low pitch male voices.

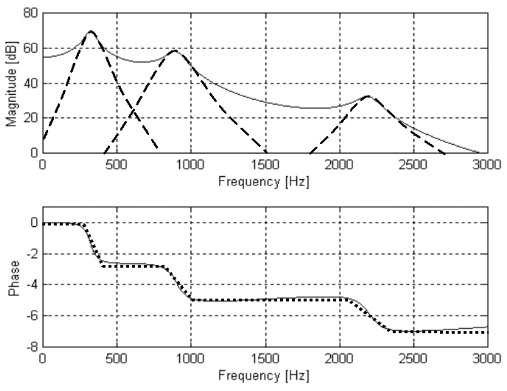


Fig 18 *The phase alignment is approximated as a linear segment, with a phase shift for each resonance*

The EpR filter implementation

The EpR filter is implemented in the frequency domain. The input is the spectrum that results out from the voiced harmonic excitation or from the voiced residual excitation. Both inputs are supposed to be approximately flat spectrums, so we just need to add the EpR resonances, the source curve and the differential spectral shape to the amplitude spectrum. In the case of the voiced harmonic excitation we also need to add the EpR phase alignment to the phase spectrum.

For each frequency bin we have to compute the value of the EpR filter. This implies a considerable computational cost, because we have to calculate the value of all the resonances. However, we can optimize this process by assuming that the value of the sum of all the resonances is equal to the maximum amplitude (dB) of all the filter and excitation resonances (over the source curve). Then we can even do better by only using the two neighbors' resonances for each frequency bin. This is not a low-quality approximation of the original method because the differential spectral shape envelope takes care of all the differences between the model and the real spectrum.

If we want to avoid the time domain voiced excitation, especially because of the computational cost of the fractional delay and the FFT, we can change it to be directly generated in the frequency domain. From the pitch and gain input we can generate a train of deltas in frequency domain (sinusoids) that will be convolved with the transform of the synthesis window and then synthesized with the standard frame based SMS synthesis, using the IFFT and overlap-add method. However the voice quality may suffer some degradation due to the fact that the sinusoids are assumed to have constant amplitude and frequency along the frame duration.

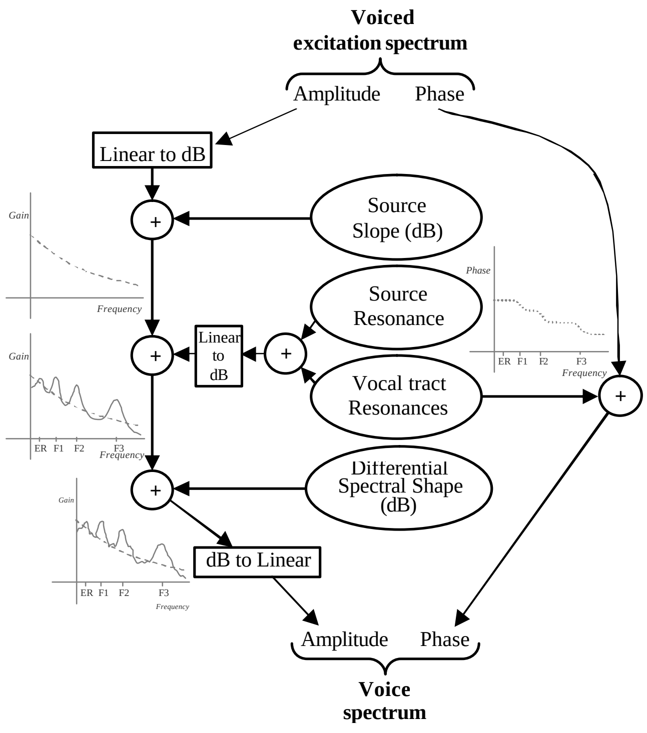


Fig 19 *Frequency domain implementation of the EpR model*

The EpR filter transformation

We can transform the EpR by changing its parameters:

excitation gain, slope and slope depth
frequency, amplitude and bandwidth of the resonances

However, we have to take into account that the differential spectral shape is related to the resonances position. Therefore, if we change the frequency of the resonances we should stretch or compress the differential spectral shape envelope according to the resonances frequency change (using the resonances center frequency as anchor points).

4.2 The singer database

The voice characteristics of a real singer are stored in a database. About one hour of recording is needed in order to build one singer database. All the audio data must be recorded in a dry and noiseless environment. The singer database is divided in two parts: timbre DB and voice DB.

4.2.1 The Timbre DB

The timbre DB stores the voice model (EpR) for each of the voiced phonemes. For each of these, we store several samples at different pitches and at different dynamics. When a phoneme with intermediate pitch or dynamics is required, the EpR parameters of the neighboring samples are interpolated.

4.2.2 The Voice DB

The voice DB represents and stores the time varying characteristics of the voice. It is divided in several categories: steady states, phonetic articulations, note attacks, note-to-note articulations, note releases and vibratos. It is possible to have different templates at different pitches for each of them.

Steady states

There are steady states stored for each of the phonemes. They model the behavior of the stationary part. To do so, they store the time-varying evolution of the EpR parameters (if the phoneme is voiced) and the SMS residual along the steady state.

Phonetic articulations

All the articulations are segmented in two regions. If the regions are different in terms of voiceness, each of the regions is analyzed with different specific parameters. Otherwise, if both regions are voiced or unvoiced the segmentation is used in synthesis to know the onset of the articulation. The behavior of the EpR model is estimated only in the voiced part regions.

Note attacks, note-to-notes and note releases

They model the excitation behavior along their duration. They are phoneme independent since they do not model the EpR changes. They can be organized into different musical evocative classification.

Vibratos

They define the behavior of the source changes along a vibrato. In particular, they track the pitch, gain and source curve changes. Each of them is segmented into attack, body and release. There can be different template labels according to some musical or expression classification.

4.3 Results

A first prototype of the singing voice synthesizer has been implemented. In order to evaluate the prototype, the synthesis obtained from our system was compared with the synthesis obtained from commercial singing synthesizers. Although the synthesis obtained was not comparable with a real singer performance, it resulted more natural than the commercial software synthesis.

However, the system presents important drawbacks. Unnatural artifacts appear in the synthesis, especially in the voiced consonants phonemes. These artifacts emerge from the fact our synthesis engine is built on top of a sinusoidal plus residual decomposition. For these phonemes, it is difficult to determine what should be included in the sinusoidal component and what not. The low register timbres of a male voice suffer from unnaturalness. This problem may come from the simplicity of the EpR phase model. Sharp attacks, specially the ones belonging to plosive phonemes are smeared. This is due to the fact the sinusoidal model can not deal with unstable partials in an accurate manner. Some solutions to this problem are proposed in [Fitz, Haken, Christensen, 2000; Verma, Meng, 2000].

So there is room for improvement in every step of the system but presuming naturalness is the essential feature we take into account whenever we evaluate the quality of a singing synthesizer, results are promising and prove the suitability of our approach.

References

Cano, P. 1998. “Fundamental Frequency Estimation in the SMS Analysis”. *Proceedings of the Digital Audio Effects Workshop (DAFX98)*, 1998.

M. de Boer; J. Bonada; Cano, P.; A. Loscos; X. Serra; 2000. “Singing voice impersonator for PC.” *Proceedings of the 2000 International Computer Music Conference*. San Francisco: Computer Music Association.

Cano, P.; A. Loscos; J. Bonada; M. de Boer; X. Serra; 2000. “Voice Morphing System for Impersonating in Karaoke Applications.” *Proceedings of the 2000 International Computer Music Conference*. San Francisco: Computer Music Association.

Childers, D.G. 1994. “Measuring and Modeling Vocal Source-Tract Interaction”. *IEEE Transactions on Biomedical Engineering 1994*.

Fitz, K.; L. Haken, and P. Christensen. 2000. “A New Algorithm for Bandwidth Association in Bandwidth-Enhanced Additive Sound Modeling”. *Proceedings of the 2000 International Computer Music Conference*. San Francisco: Computer Music Association.

Klatt, D.H. 1980. “Software for a cascade/parallel formant synthesizer” *Journal Acoustics of American Society*, 971-995.

Loscos, A.; Cano, P.; Bonada, J. 1999. “Low Delay Singing Voice Alignment to text” *Proceedings of the ICMC 1999*

Osaka, N. 1995. “Timbre Interpolation of sounds using a sinusoidal model”, *Proceedings of the ICMC 1995*.

Serra, X. and J. Smith. 1990. “Spectral Modeling Synthesis: A Sound Analysis/Synthesis System based on a Deterministic plus Stochastic Decomposition.’’ *Computer Music Journal* 14(4):12--24.

Serra, X. 1994. “Sound Hybridization Techniques based on a Deterministic plus Stochastic Decomposition Model.” *Proceedings of the 1994 International Computer Music Conference*. San Francisco: Computer Music Association.

Serra, X. 1996. “Musical Sound Modeling with Sinusoids plus Noise”, in G. D. Poli, A. Picialli, S. T. Pope, and C. Roads, editors, *Musical Signal Processing*. Swets & Zeitlinger Publishers.

Serra, X.; J. Bonada. 1998. "Sound Transformations Based on the SMS High Level Attributes", *Proceedings of the 98 Digital Audio Effects Workshop*.

Settel, Z., C. Lippe. 1996. “Real-Time Audio Morphing”, *7th International Symposium on Electronic Art*, 1996.

Slaney, M., M. Covell, B. Lassiter. 1996. “Automatic audio morphing”, *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.* 2, 1001-1004 (1996).

Smith, J.O., Gosset, P. 1984. "A flexible sampling-rate conversion method", *Proceedings of the ICASSP*, San Diego, New York, vol. 2, pp 19.4.1-19.4.2. IEEE Press.

Tellman, E., L. Haken, B. Holloway. 1995. “Timbre Morphing of Sounds with Unequal Number of Features”, *J. Audio Eng. Soc.*, 43:9 1995.

Tellman, E.; L. Haken; B. Holloway 1995. “Timbre morphing of sounds with unequal numbers of features”. *Journal of the Audio Engineering Society*, 43(9), 678-89.

Verma, T. S. ; T. H. Y. Meng. 2000. “Extending Spectral Modeling Synthesis With Transient Modeling Synthesis”, *Computer Music Journal* 24:2, pp.47-59.

Singing Voice Synthesis Combining Excitation plus Resonance and Sinusoidal plus Residual Models

Jordi Bonada, Òscar Celma, Àlex Loscos, Jaume Ortolà, Xavier Serra
Music Technology Group, Audiovisual Institute, Pompeu Fabra University
Barcelona, Spain
{jordi.bonada, oscar.celma, alex.loscos, jaume.ortola, xavier.serra}@iua.upf.es
http://www.iua.upf.es/mtg

Yasuo Yoshioka, Hiraku Kayama, Yuji Hisaminato, Hideki Kenmochi
Advanced System Development Center, YAMAHA Corporation
Hamamatsu, Japan
{yoshioka, kayama, hisaminato, kenmochi}@beat.yamaha.co.jp

Abstract

This paper presents an approach to the modeling of the singing voice with a particular emphasis on the naturalness of the resulting synthetic voice. The underlying analysis/synthesis technique is based on the Spectral Modeling Synthesis (SMS) and a newly developed Excitation plus Resonance (EpR) model. With this approach a complete singing voice synthesizer is developed that generates a vocal melody out of the score and the phonetic transcription of a song.

1 Introduction

The human voice is clearly the most flexible and fascinating of the musical instruments. All through the history of music, it has attracted the attention of composers and has captivated audiences. Already organ builders had the dream of imitating as faithfully as possible the sound of human voice, as we can hear in the Vox Humana stop in some organs. With the use of the new digital technologies and our current understanding of the voice, the possibility of synthesizing a natural singing voice has become more feasible.

The work presented here is a continuation of an automatic singing voice impersonator system for karaoke [Cano, Loscos, Bonada, de Boer, Serra, 2000]. That system morphed the voice attributes of a user (such as pitch, timbre, vibrato and articulations) with the ones from a prerecorded singer in real time.

The focus of our current work is to generate a performance of an artificial singer out of the musical score (melody) and the phonetic transcription (lyrics) of a song. To achieve such goal we have defined a quality evaluation criteria which takes naturalness as the fundamental consideration.

2 Introduction to Singing Voice Synthesis

Singing voice synthesis has been an active research field for almost fifty years [Cook, 1996]. Traditionally, the voice has been modeled as a linear system consisting of one or more sound sources and a set of filters which shape the spectrum of these sources. The sound source can be a periodic signal, a noisy signal, or a mixture of both, and the set of filters can be regarded as the vocal tract filters. The resulting spectrum is mainly characterized by resonant peaks called formants. Thus a vocal synthesizer has to allow the control of the resonant peaks of the spectrum and of the source parameters.

With regard to the synthesis models used in singing voice synthesis, they can be classified into two groups: Spectral models, which can be viewed as based on perceptual mechanisms, and Physical models, which can be viewed as based on production mechanisms. Any of these two models might be suitable depending on the specific requirements of the application, or they may even be combined to take advantage of both approaches.

The main benefit of using Physical models is that the parameters of the model are closely related to the ones a singer uses to control his/her own vocal system. As such, some knowledge of the real-world mechanism can be introduced in the design. The model itself can provide intuitive parameters if it is constructed with the intention that it sufficiently matches the physical system. Conversely, such a system usually has a large number of parameters. This turns the mapping of the controls of the production mechanism to the final output, and so to the listener's perceived quality, into something not trivial.

On the other hand, Spectral models are closely related to some aspects of the human perceptual mechanism. Changes in the parameters of a spectral model can be more easily mapped to a change of sensation in the listener. Yet parameter spaces yielded by these systems are not necessarily the most natural ones for manipulation. The methods based on spectral models include Frequency Modulation, FOFs, Vocoder and sinusoidal models. Acoustic tube models are an example of physical models. Linear Predictive Coding (LPC) and formant synthesizers can be considered as spectral models and also as pseudo-physical, not strictly physical because of the source/filter decomposition they use.

Some commercial software products for singing synthesis have been released in recent years. For example, Vocalwriter [VOCALWRITER, 2000] for the English language and SmartTalk 3.0 for the Japanese language have to be mentioned [SMARTTALK, 2000]. However, the systems developed until now are far from providing enough quality to meet the practical requirements of real-world commercial applications.

The goal of a singing voice synthesis indistinguishable from a real human voice is still remote. Thus, there is a lot of room for improvement in this research area, and naturalness is one of the keywords for the work to be done. Moreover, it seems that one of the main issues behind singing voice synthesis is to offer not only quality but flexible and musically meaningful controls over the vocal sound. In that sense, we may think of applications where impossible singing voices can be synthesized or where existing voices can be enhanced.

In the next section we present the EpR voice model that we have developed and in the following ones we describe the complete voice synthesis system built around the model.

3 The EpR voice model

Our singing voice synthesizer is based on an extension of the well known source/filter approach [Childers, 1994] that we call Excitation plus Resonance (EpR). This EpR model is built on top of the sinusoidal plus residual representation obtained by the SMS analysis [Serra, 1990]. Thus the model parameters are extracted from real voice sounds that have been analyzed with SMS and in the synthesis stage the model parameters are converted to SMS parameters, from which the output sound is obtained. In this article we concentrate on the EpR model part of the method developed.

Figure 1 shows a diagram with the three types of excitation used and the corresponding filters. For the case of a voiced phonation, the filter applied to each excitation generates the appropriate spectral shape by using a frequency domain filtering function that is decomposed into two cascaded operations: an exponential decay curve plus several resonances. After filtering, the voiced residual excitation needs to be transposed to the synthesis pitch because it is a filtered SMS residual recording and has traces of the original pitch. Otherwise, in the case of an unvoiced phonation, we apply a filter that just changes the tilt curve and the gain of the STFT (Short-time Fourier Transform) of an original recording.

3.1 The EpR excitation

Voiced harmonic excitation

The inputs that control the voiced harmonic excitation generation are the desired pitch and gain envelopes. The actual excitation signal can either be generated in the time or the frequency domains. The most flexible excitation is obtained by generating a delta train in the time domain, thus allowing to achieve the best period resolution and to use some simple excitation templates. This can also be useful to generate jitter or different types of vocal disorders. This delta train can be seen as a glottal source wave previous to a convolution with the differentiated glottal pulse.

A fractional delay filter is needed to position the excitation deltas between samples, since we have to go beyond the sampling rate resolution. The filter is implemented using a windowed sinc-like function situated at each pulse location with the offset subtracted [Smith, Gosset, 1984]. Then the signal is windowed and the FFT computed. The result is a spectrum approximately flat that contains the harmonics approximately synchronized in phase.

When the harmonic excitation is generated directly is the frequency domain, which is good enough for many voiced sounds, the excitation spectrum will be perfectly flat and the phase synchronization precise.

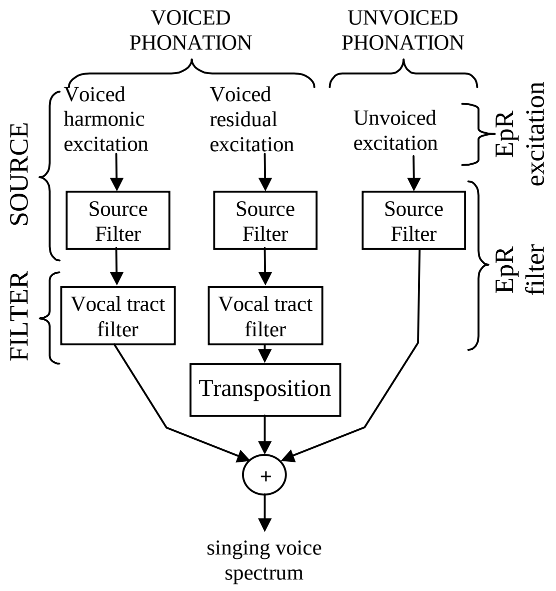


Figure 1. The EpR voice model.

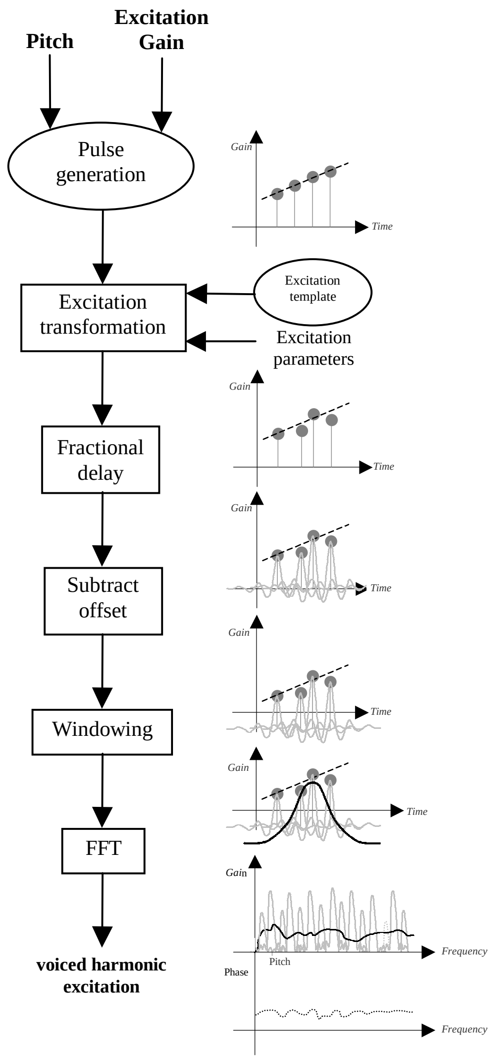


Figure 2. The EpR voiced harmonic excitation.

Voiced residual excitation

The voiced residual excitation is obtained from the residual of the SMS analysis of a long steady state vowel recorded from a real singer. The SMS residual is then inverse-filtered by its short-time average spectral shape envelope to get an approximately flat excitation magnitude spectrum.

Unvoiced excitation

The excitation in the unvoiced parts of the sounds uses directly the original recording of a singer's performance.

3.2 The EpR filter

The EpR filter is the combination of two cascaded filters. The first of them models the differentiated glottal pulse frequency response, and the second the vocal tract (resonance filter).

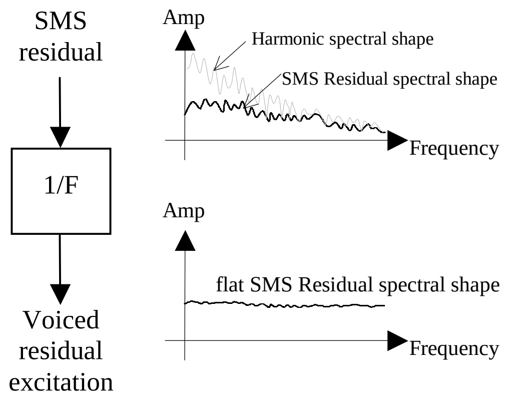

SMS residual
1/F
Voiced residual excitation
Amp
Harmonic spectral shape
SMS Residual spectral shape
Frequency
Amp
flat SMS Residual spectral shape
Frequency

Figure 3. The voiced residual excitation.

The EpR source filter

The EpR source filter is modeled as a frequency domain curve and one source resonance applied to the input frequency domain flat excitation described in the previous section.

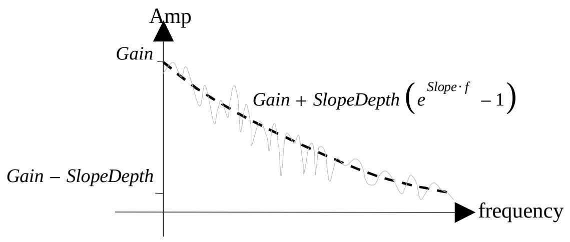


Figure 4. The EpR source curve.

This source curve is defined by a gain and an exponential decay as follows:

$$Source_{dB} = Gain_{dB} + SlopeDepth_{dB}\left(e^{Slope\cdot f}-1\right) \quad (1)$$

This curve is obtained from an approximation to the harmonic spectral shape (*HSS*) determined by the harmonics identified in the SMS analysis

$$HSS\left(f\right) = envelope_{i=0..n}\left[f_i, 20\log\left(a_i\right)\right] \quad (2)$$

where *i* is the index of the harmonic, *n* is the number of harmonics, f_i and a_i are the frequency and amplitude of the i^{th} harmonic.

On top of the source curve, we add a second resonance in order to model the low frequency content of the spectrum below the first formant. This resonance affects the synthesis in a different way than the vocal tract resonances, as will be explained later.

The source resonance is modeled as a symmetric second order filter (based on the Klatt formant synthesizer [Klatt, 1980]) with center frequency *F*, bandwidth *Bw* and linear amplitude *Amp*. The transfer function of the resonance *R(f)* can be expressed as follows:

$$H(z) = \frac{A}{1 - Bz^{-1} - Cz^{-2}}$$
$$R(f) = Amp \frac{H\left(e^{j2\pi\left(0.5+\frac{f-F}{fs}\right)}\right)}{H\left(e^{j\pi}\right)} \quad (3)$$

where

$$fs = \text{Sampling rate}$$
$$C = -e^{-\frac{2\pi Bw}{fs}}$$
$$B = 2\cos(\pi)e^{-\frac{\pi Bw}{fs}}$$
$$A = 1 - B - C$$

The amplitude parameter (*Amp*) is relative to the source curve (a value of 1 means the resonance maximum is just over the source curve).

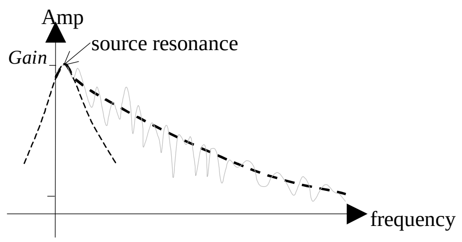


Figure 5. The EpR source resonance.

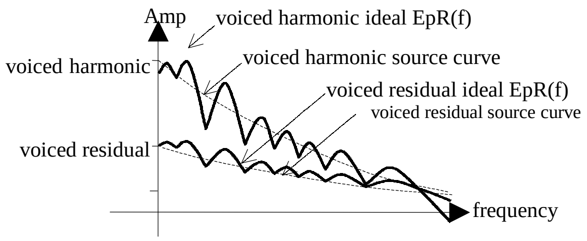

Amp
voiced harmonic ideal EpR(f)
voiced harmonic source curve
voiced harmonic
voiced residual ideal EpR(f)
voiced residual source curve
voiced residual
frequency

Figure 6. The EpR filter resonances.

The EpR vocal tract filter

The vocal tract is modeled by a collection of resonances plus a differential spectral shape envelope. These filter resonances are modeled in the same way as the source resonance, where the lower frequency resonances are somewhat equivalent to the vocal tract formants.

The EpR filters for voiced harmonic and residual excitations are basically the same, they just differ in the gain and slope depth parameters. This approximation has been obtained after comparing the harmonic and residual spectral shape of several SMS analysis of singer recordings. Figure 7 shows these differences.

The differential spectral shape envelope actually stores the differences (in dB) between the ideal EpR model (*iEpR*) and the real harmonic spectral shape (*HSS*) of a singer's performance. We calculate it as a 30 Hz equidistant step envelope.

$$DSS(f) = envelope_{i=0..}\left[30i, HSS_{dB}(30i) - iEpR_{dB}(30i)\right] \quad (4)$$

The EpR phase alignment

The phase alignment of the harmonics at the beginning of each period is obtained from the EpR spectral phase envelope. A time shift is applied just before the synthesis, in order to get the actual phase envelope (usually it will not match the beginning of the period). This phase alignment is then added to the voiced harmonic excitation spectrum phase envelope. The EpR spectral phase model states that each vocal tract resonance produces a linear shift of π on the flat phase envelope with a bandwidth depending on the estimated resonance bandwidth. This phase model is especially important for the intelligibility and in order to get more natural low pitched male voices.

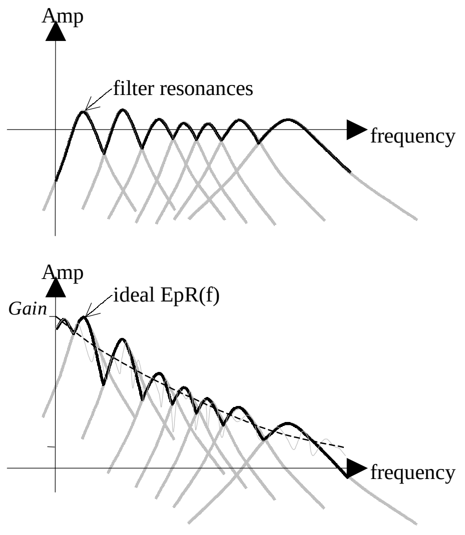


Figure 7. Differences between harmonic and residual EpR filters.

The EpR filter implementation

The EpR filters are implemented in the frequency domain. The input is the spectrum that results from the voiced harmonic excitation or from the voiced residual excitation. Both inputs are approximately flat spectrums, so we just need to add the EpR resonances, the source curve and the differential spectral shape to the amplitude spectrum. In the case of the voiced harmonic excitation we also need to add the EpR phase alignment to the phase spectrum.

For each frequency bin we have to compute the value of the EpR filter. This implies a considerable computational cost, because we have to calculate the value of all the resonances. However, we can optimize this process by assuming that the value of the sum of all the resonances is equal to the maximum amplitude (dB) of the whole filter and excitation resonances (over the source curve) at that bin. Then we can even do better by only using the two neighboring resonances for each frequency bin. This is not a low-quality approximation of the original method because the differential spectral shape envelope, which is always kept, takes care of all the differences between the model and the real spectrum.

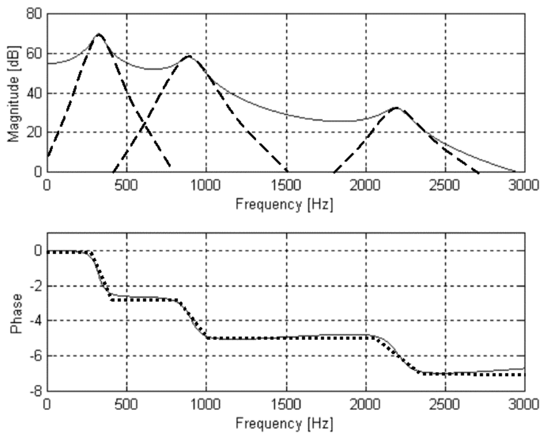


Figure 8. Phase alignment of the EpR resonances.

If we want to avoid the generation of the time domain voiced excitation, especially because of the computational cost of the fractional delay and the FFT, we can generate it in the frequency domain. From the pitch and gain input we can generate a train of deltas in frequency domain (sinusoids) that will be convolved with the transform of the synthesis window and then synthesized with the standard frame based SMS synthesis, using the IFFT and overlap-add method. However the voice quality may suffer some degradation due to the fact that the sinusoids are assumed to have constant amplitude and frequency throughout the frame duration.

The EpR filter transformation

We can transform the EpR by changing its parameters: excitation gain, slope and slope depth frequency, amplitude and bandwidth of the resonances. However, we have to take into account that the differential spectral shape is related to the resonances position. Therefore, if we change the frequency of the resonances we should stretch or compress the differential spectral shape envelope according to the resonances frequency change (using the resonances center frequency as anchor points).

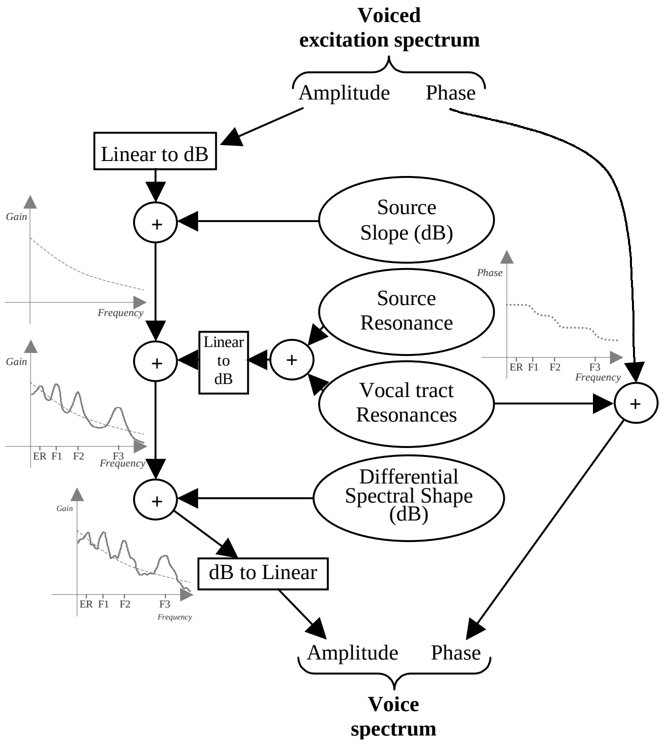


Figure 9. Frequency domain implementation of the EpR model.

4 System Overview

From the synthesis model presented in the previous section we have developed a complete singing voice synthesizer. The inputs to the system are the melody and the lyrics (in English or Japanese language), with its phonetic transcription in SAMPA format [SAMPA, 2000]. The system is able to read standard MIDI files as well as METRIX files (an ascii text file format designed as a sound synthesis control language [Amatriain, 1998]). The output of the system is a wave file with the synthesized singing voice.

The system is composed of two modules. The first one, Expressiveness module, controls the performance of the synthetic voice giving some expressiveness and naturalness to the input melody. The output of the Expressiveness module is a detailed musical score with all the needed features to characterize an expressive performance of a singer, which we call MicroScore. The second module, Synthesis module, is in charge of the actual synthesis process and uses a database containing the voice and timbre characteristics of a real singer. The analyzed voice data of the database is the result of the SMS analysis and the EpR modelization.

5 Musical Controls

To achieve naturalness on the output voice, the system defines a set of musically meaningful controls that are either related to individual notes (note parameters) or to the whole song (general control parameters). With these parameters the system attempts to cover as many situations as possible in a singing performance. In addition to the common musical parameters (pitch, duration and dynamics), the system uses other parameters to control vocal characteristics such as attack, release, vibrato, articulation between notes, etc. The specification of all these controls has been thought of with the end-user in mind, so as to be as easy as possible to control.

The table below shows a list of the note parameters as well as the general controls. These parameters are defined in the MicroScore structure:

Note parameters	**Observations**
Pitch	Midi number (0-127) or G#3
Begin Time	of the note, in milliseconds
Duration	of the note, in milliseconds
Loudness	Normalized value [0, 1]
Lyrics	Syllable associate in a note
Dynamic Envelope	(time, normalized value)
Pitch Envelope	(time, cents)
Attack Type	{normal, sharp, soft, high}
Attack Duration	Normalized value [0, 1]
Release Type	{normal, soft, high}
Release Duration	Normalized value [0, 1]
Transition Type	{legato, staccato, marcato, portamento, glissando}
Transition Duration	Normalized value [0, 1]
Vibrato Type	---
Vibrato Depth	Envelope (time, cents)
Vibrato Rate	Envelope (time, Hz)
Opening of vowels	Envelope
Hoarseness	Envelope
Whisper	Envelope

Control parameters	**Observations**
Singer Type	Change singer of the DB
Gender Type	Change gender (male/female)
Transposition	To transpose input melody

5.1 Expressiveness Controls

In order to obtain a good synthesis, it is essential to take into account the high-level expressive controls used by real performers. In our system, this is done by the expressiveness module, which is in charge of applying certain deviations to the input score with the purpose of making it more natural and expressive.

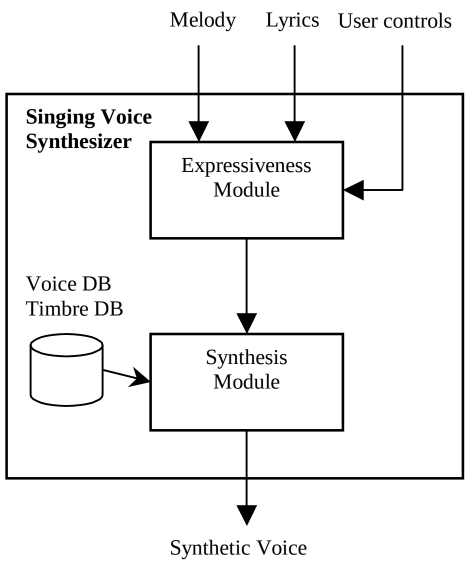


Figure 10. System overview.

Following the research made by Sundberg and Friberg on musical expressiveness [Friberg, 1991], we have implemented a very basic rule-based system for expressive control. These rules were designed as a general tool for any kind of instruments, and need adaptations and proper tuning to be suitable for the singing voice [Berndtsson, 1996]. So far, only a few rules have been successfully tested in our system. We still need more experimentation to include rules specific to voice parameters such as vibrato, breathiness, hoarseness or special kinds of attack.

An important contribution to the expressive control of sound synthesis systems has been made by Manfred Clynes [Clynes, 1987]. In view of the fact that the human ear is specially sensitive to the amplitude contour of a note, Clynes has developed a mechanism to shape individually the amplitude of each note to give more authenticity to the synthesis. In this method, called *Predictive Amplitude Shaping*, there is a global amplitude contour for every note within a musical fragment which is slightly modified for each individual note according to the pitch interval to the next note. This creates a sense of continuity and phrasing. We have implemented and tested this technique in our system, adapting the parametric controls to the human voice, with excellent results.

The pitch contour of the singing voice has to be also carefully generated in order to obtain a faithful synthesis. So we have designed a mathematical model for reproducing the smooth pitch transitions between notes. This model allows us to control the transition duration and the tuning deviations at the end and the beginning of the notes in accordance with the musical context. In the note to note transitions, the synchronization between phonetics and musical rhythm is assured by reaching always the target pitch at the onset of the vowel of each syllable.

6 Singer Database

The creation of the database is one of the critical steps in most speech and singing voice synthesizers. There are no singing voice databases available, thus we recorded our own samples by first defining a set of musical exercises to be performed by a singer (i.e. different type of attacks, releases, note to note transitions, etc). Needless to say, the voice had to be recorded in a dry and noiseless environment to get the best possible SMS analysis.

In terms of what information is needed to be stored in the database, we can either attempt to record as many samples as possible (all possible pitches, attacks, …) or start from fewer samples and obtain the rest through transformations from the recorded/analyzed ones. This second approach gives more flexibility to the system at the possible expense of sound quality. Our approach has been this second one.

In order to choose the most useful English phonetic articulations to be included in the database, we ordered all possible articulations according to its frequency of use in actual songs. A statistical analysis was made from 76000 English songs, with close to three million words. With this information, now we know how many articulations are needed to cover a fixed percentage of all the possible articulations.

Number of articulations	Covered percentage
71	50 %
308	90 %
395	95 %
573	99 %
785	99.9 %
1129	100 %

The database is organized into two parts; timbre and voice DB.

6.1 Timbre DB

The timbre database stores the voice model (EpR) for each of the voiced phonemes. For each of these, we store several samples at different pitches and at different dynamics. When a phoneme with intermediate pitch or dynamics is required, the EpR parameters of the neighboring samples are interpolated to synthesize the phoneme at the desired pitch.

6.2 Voice DB

The voice database includes analysis data from the time varying characteristics of the voice in the form of templates. It is divided into several categories: steady states, phonetic articulations, note attacks, note-to-note articulations, note releases and vibratos. It is also possible to have different templates for the different pitches of the same type of sound.

Steady states

There are steady states stored for each of the phonemes. They model the behavior of the stationary part of a note. To do so, we store the time-varying evolution of the EpR parameters (if the phoneme is voiced) and the SMS residual along the steady state.

Phonetic articulations

All the articulations are segmented in two regions, the end of the first sound and the beginning of the next. If the regions are different in terms of voiceness, each region is analyzed with different specific parameters. Otherwise, if both regions are voiced or unvoiced the segmentation is used in synthesis to control the onset of the articulation. The EpR parameters are estimated only in the voiced part regions.

Vibratos

The vibrato templates characterize the vibrato characteristics by keeping the behavior of the voice excitation parameters. In particular, the fundamental frequency evolution, gain and source curve changes. Each time-varying function is segmented into attack, body and release. There can be different template labels according to some musical or expression classifications.

Note attacks, note transitions and note releases

These templates model the amplitude and fundamental frequency evolution at the note boundaries. They can easily be represented by a model (such as the one proposed for the amplitude by Clynes [Clynes, 1987]) which will give a smoother time evolution of the synthetic vocal sound. These templates, or models, are phoneme independent since they do not model the EpR changes. They can be organized into different expression categories.

7 Conclusions

With the purpose of demonstrating the potential of our system, two small databases with a male singer and a female singer have been created. With the female voice, we have synthesized fragments of two different songs (“We’ve only just begun”, by The Carpenters, and “Natural woman”, by Carol King). With the male voice, we have also synthesized the same song by The Carpenters and some choruses for accompanying the female songs.

The system evaluation was made by a group of musicians that had never heard about the project. Taking the Vocalwriter synthesizer as a comparison for the English synthesis, our demo songs were considered more intelligible and more natural. In contrast, our synthesis showed a lack of timbre uniformity. As the evaluation conclusion, although the synthesis obtained was not comparable with a real singer performance, the system was judged to be of a higher quality than the available commercial systems.

From our evaluation the system still presents important drawbacks. The most important one are that unnatural artifacts appear in the synthesis, specially in the voiced consonants phonemes. Also the low register timbres of a male voice suffer from unnaturalness and sharp attacks, specially the ones belonging to plosive phonemes are smeared. We can think of some solutions to these problems by incorporating some recently proposed enhancements to the sinusoidal modeling of musical sounds [Fitz, Haken, Christensen, 2000; Verma, Meng, 2000].

The creation of a complete voice database is for now a demanding process that has to be supervised by a user. More automated ways have to be developed to facilitate and speed up the database creation process.

Certainly there is room for improvement in every step of the system. Even so, assuming, naturalness as the essential feature for evaluating the quality of a singing synthesizer, results are promising and prove the suitability of our approach.

8 References

Amatriain, X. 1998. "METRIX: A Musical Data Definition Language and Data Structure for a Spectral Modeling Based Synthesizer," *Proceedings of 98 Digital Audio Effects Workshop.*

Berndtsson, G. 1996. "The KTH Rule System for Singing Synthesis," *Computer Music Journal, 20:1, 1996.*

Cano, P.; A. Loscos; J. Bonada; M. de Boer; X. Serra. 2000. "Voice Morphing System for Impersonating in Karaoke Applications," *Proceedings of the 2000 International Computer Music Conference*. San Francisco: Computer Music Association.

Childers, D. G. 1994. "Measuring and Modeling Vocal Source-Tract Interaction," *IEEE Transactions on Biomedical Engineering 1994.*

Clynes, M. 1987. "What can a musician learn about music performance from newly discovered microstructure principles (PM and PAS)?," *Action and Perception in Rhythm and Music. Royal Swedish Academy of Music No. 55, 1987.*

Cook, P. 1996. "Singing Voice Synthesis History, Current Work, and Future Directions," *Computer Music Journal, 20:2 1996.*

Fitz, K.; L. Haken; P. Christensen. 2000. "A New Algorithm for Bandwidth Association in Bandwidth-Enhanced Additive Sound Modeling," *Proceedings of the 2000 International Computer Music Conference*.

Friberg, A. 1991. "Generative Rules for Music Performance: A Formal Description of a Rule System," *Computer Music Journal 15:2, 1991.*

Klatt, D. H. 1980. "Software for a cascade/parallel formant synthesizer," *Journal of the Acoustical Society of America, 971-995, 1980.*

SAMPA, 2000. Speech Assessment Methods Phonetic Alphabet machine-readable phonetic alphabet. http://www.phon.ucl.ac.uk/home/sampa/home.htm

Serra, X.; J. Smith. 1990. "Spectral Modeling Synthesis: A Sound Analysis/Synthesis System based on a Deterministic plus Stochastic Decomposition,'' *Computer Music Journal* 14(4):12-24.

Smith, J.O.; P. Gosset. 1984. "A flexible sampling-rate conversion method," *Proceedings of the ICASSP, San Diego, New York, vol. 2, pp 19.4.1-19.4.2. IEEE Press 1984.*

SMARTTALK, 2000. SmartTalk 3.0. Japanese Speech-Synthesis Engine with Singing Capability. http://www.oki.co.jp/OKI/Cng/Softnew/English/sm.htm

Verma, T. S.; T. H. Y. Meng. 2000. "Extending Spectral Modeling Synthesis With Transient Modeling Synthesis" *Computer Music Journal* 24:2, pp.47-59.

VOCALWRITER, 2000. Vocalwriter. Music and Vocal synthesis. http://www.kaelabs.com/

Singing Voice Synthesis Combining Excitation plus Resonance and Sinusoidal plus Residual Models

Jordi Bonada, Òscar Celma, Àlex Loscos, Jaume Ortolà, Xavier Serra
Music Technology Group, Audiovisual Institute, Pompeu Fabra University
Barcelona, Spain
{jordi.bonada, oscar.celma, alex.loscos, jaume.ortola, xavier.serra}@iua.upf.es
http://www.iua.upf.es/mtg

Yasuo Yoshioka, Hiraku Kayama, Yuji Hisaminato, Hideki Kenmochi
Advanced System Development Center, YAMAHA Corporation
Hamamatsu, Japan
{yoshioka, kayama, hisaminato, kenmochi}@beat.yamaha.co.jp

Abstract

This paper presents an approach to the modeling of the singing voice with a particular emphasis on the naturalness of the resulting synthetic voice. The underlying analysis/synthesis technique is based on the Spectral Modeling Synthesis (SMS) and a newly developed Excitation plus Resonance (EpR) model. With this approach a complete singing voice synthesizer is developed that generates a vocal melody out of the score and the phonetic transcription of a song.

1 Introduction

The human voice is clearly the most flexible and fascinating of the musical instruments. All through the history of music, it has attracted the attention of composers and has captivated audiences. Already organ builders had the dream of imitating as faithfully as possible the sound of human voice, as we can hear in the Vox Humana stop in some organs. With the use of the new digital technologies and our current understanding of the voice, the possibility of synthesizing a natural singing voice has become more feasible.

The work presented here is a continuation of an automatic singing voice impersonator system for karaoke [Cano, Loscos, Bonada, de Boer, Serra, 2000]. That system morphed the voice attributes of a user (such as pitch, timbre, vibrato and articulations) with the ones from a prerecorded singer in real time.

The focus of our current work is to generate a performance of an artificial singer out of the musical score (melody) and the phonetic transcription (lyrics) of a song. To achieve such goal we have defined a quality evaluation criteria which takes naturalness as the fundamental consideration.

2 Introduction to Singing Voice Synthesis

Singing voice synthesis has been an active research field for almost fifty years [Cook, 1996]. Traditionally, the voice has been modeled as a linear system consisting of one or more sound sources and a set of filters which shape the spectrum of these sources. The sound source can be a periodic signal, a noisy signal, or a mixture of both, and the set of filters can be regarded as the vocal tract filters. The resulting spectrum is mainly characterized by resonant peaks called formants. Thus a vocal synthesizer has to allow the control of the resonant peaks of the spectrum and of the source parameters.

With regard to the synthesis models used in singing voice synthesis, they can be classified into two groups: Spectral models, which can be viewed as based on perceptual mechanisms, and Physical models, which can be viewed as based on production mechanisms. Any of these two models might be suitable depending on the specific requirements of the application, or they may even be combined to take advantage of both approaches.

The main benefit of using Physical models is that the parameters of the model are closely related to the ones a singer uses to control his/her own vocal system. As such, some knowledge of the real-world mechanism can be introduced in the design. The model itself can provide intuitive parameters if it is constructed with the intention that it sufficiently matches the physical system. Conversely, such a system usually has a large number of parameters. This turns the mapping of the controls of the production mechanism to the final output, and so to the listener's perceived quality, into something not trivial.

On the other hand, Spectral models are closely related to some aspects of the human perceptual mechanism. Changes in the parameters of a spectral model can be more easily mapped to a change of sensation in the listener. Yet parameter spaces yielded by these systems are not necessarily the most natural ones for manipulation. The methods based on spectral models include Frequency Modulation, FOFs, Vocoder and sinusoidal models. Acoustic tube models are an example of physical models. Linear Predictive Coding (LPC) and formant synthesizers can be considered as spectral models and also as pseudo-physical, not strictly physical because of the source/filter decomposition they use.

Some commercial software products for singing synthesis have been released in recent years. For example, Vocalwriter [VOCALWRITER, 2000] for the English language and SmartTalk 3.0 for the Japanese language have to be mentioned [SMARTTALK, 2000]. However, the systems developed until now are far from providing enough quality to meet the practical requirements of real-world commercial applications.

The goal of a singing voice synthesis indistinguishable from a real human voice is still remote. Thus, there is a lot of room for improvement in this research area, and naturalness is one of the keywords for the work to be done. Moreover, it seems that one of the main issues behind singing voice synthesis is to offer not only quality but flexible and musically meaningful controls over the vocal sound. In that sense, we may think of applications where impossible singing voices can be synthesized or where existing voices can be enhanced.

In the next section we present the EpR voice model that we have developed and in the following ones we describe the complete voice synthesis system built around the model.

3 The EpR voice model

Our singing voice synthesizer is based on an extension of the well known source/filter approach [Childers, 1994] that we call Excitation plus Resonance (EpR). This EpR model is built on top of the sinusoidal plus residual representation obtained by the SMS analysis [Serra, 1990]. Thus the model parameters are extracted from real voice sounds that have been analyzed with SMS and in the synthesis stage the model parameters are converted to SMS parameters, from which the output sound is obtained. In this article we concentrate on the EpR model part of the method developed.

Figure 1 shows a diagram with the three types of excitation used and the corresponding filters. For the case of a voiced phonation, the filter applied to each excitation generates the appropriate spectral shape by using a frequency domain filtering function that is decomposed into two cascaded operations: an exponential decay curve plus several resonances. After filtering, the voiced residual excitation needs to be transposed to the synthesis pitch because it is a filtered SMS residual recording and has traces of the original pitch. Otherwise, in the case of an unvoiced phonation, we apply a filter that just changes the tilt curve and the gain of the STFT (Short-time Fourier Transform) of an original recording.

3.1 The EpR excitation

Voiced harmonic excitation

The inputs that control the voiced harmonic excitation generation are the desired pitch and gain envelopes. The actual excitation signal can either be generated in the time or the frequency domains. The most flexible excitation is obtained by generating a delta train in the time domain, thus allowing to achieve the best period resolution and to use some simple excitation templates. This can also be useful to generate jitter or different types of vocal disorders. This delta train can be seen as a glottal source wave previous to a convolution with the differentiated glottal pulse.

A fractional delay filter is needed to position the excitation deltas between samples, since we have to go beyond the sampling rate resolution. The filter is implemented using a windowed sinc-like function situated at each pulse location with the offset subtracted [Smith, Gosset, 1984]. Then the signal is windowed and the FFT computed. The result is a spectrum approximately flat that contains the harmonics approximately synchronized in phase.

When the harmonic excitation is generated directly is the frequency domain, which is good enough for many voiced sounds, the excitation spectrum will be perfectly flat and the phase synchronization precise.

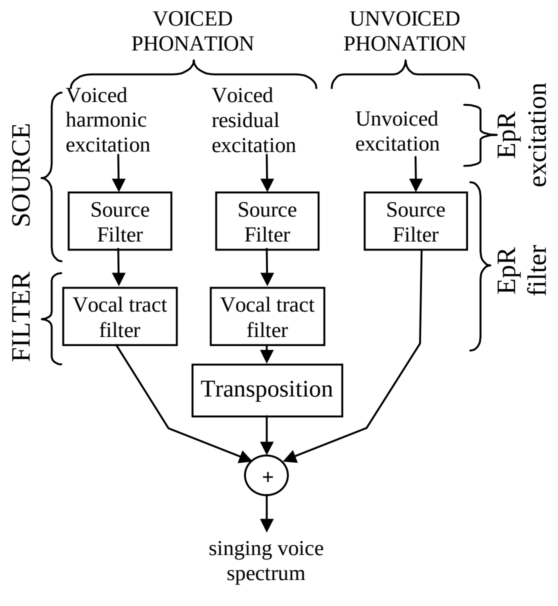


Figure 1. The EpR voice model.

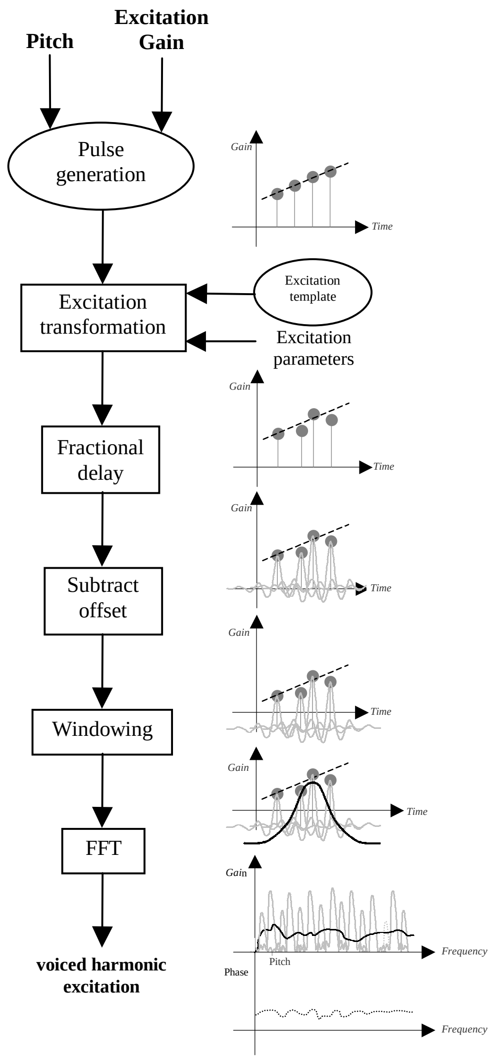


Figure 2. The EpR voiced harmonic excitation.

Voiced residual excitation

The voiced residual excitation is obtained from the residual of the SMS analysis of a long steady state vowel recorded from a real singer. The SMS residual is then inverse-filtered by its short-time average spectral shape envelope to get an approximately flat excitation magnitude spectrum.

Unvoiced excitation

The excitation in the unvoiced parts of the sounds uses directly the original recording of a singer's performance.

3.2 The EpR filter

The EpR filter is the combination of two cascaded filters. The first of them models the differentiated glottal pulse frequency response, and the second the vocal tract (resonance filter).

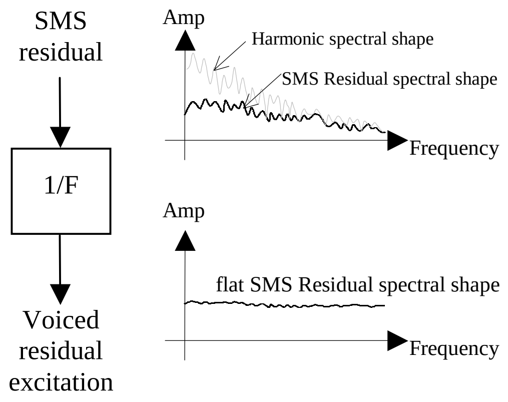

SMS residual
1/F
Voiced residual excitation
Amp
Harmonic spectral shape
SMS Residual spectral shape
Frequency
Amp
flat SMS Residual spectral shape
Frequency

Figure 3. The voiced residual excitation.

The EpR source filter

The EpR source filter is modeled as a frequency domain curve and one source resonance applied to the input frequency domain flat excitation described in the previous section.

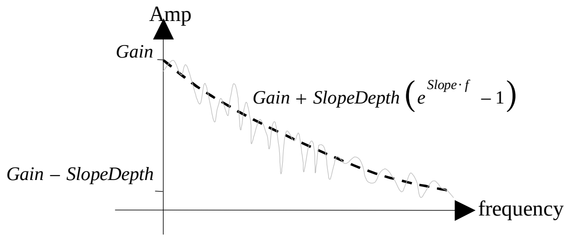


Figure 4. The EpR source curve.

This source curve is defined by a gain and an exponential decay as follows:

$$Source_{dB} = Gain_{dB} + SlopeDepth_{dB}\left(e^{Slope\cdot f}-1\right) \quad (1)$$

This curve is obtained from an approximation to the harmonic spectral shape (*HSS*) determined by the harmonics identified in the SMS analysis

$$HSS\left(f\right) = envelope_{i=0..n}\left[f_i, 20\log\left(a_i\right)\right] \quad (2)$$

where i is the index of the harmonic, n is the number of harmonics, f_i and a_i are the frequency and amplitude of the i^{th} harmonic.

On top of the source curve, we add a second resonance in order to model the low frequency content of the spectrum below the first formant. This resonance affects the synthesis in a different way than the vocal tract resonances, as will be explained later.

The source resonance is modeled as a symmetric second order filter (based on the Klatt formant synthesizer [Klatt, 1980]) with center frequency *F*, bandwidth *Bw* and linear amplitude *Amp*. The transfer function of the resonance *R(f)* can be expressed as follows:

$$H(z) = \frac{A}{1 - Bz^{-1} - Cz^{-2}}$$

$$R(f) = Amp \frac{H\left(e^{j2\pi\left(0.5+\frac{f-F}{fs}\right)}\right)}{H\left(e^{j\pi}\right)} \quad (3)$$

where

$$fs = \text{Sampling rate}$$

$$C = -e^{-\frac{2\pi Bw}{fs}}$$

$$B = 2\cos(\pi)e^{-\frac{\pi Bw}{fs}}$$

$$A = 1 - B - C$$

The amplitude parameter (*Amp*) is relative to the source curve (a value of 1 means the resonance maximum is just over the source curve).

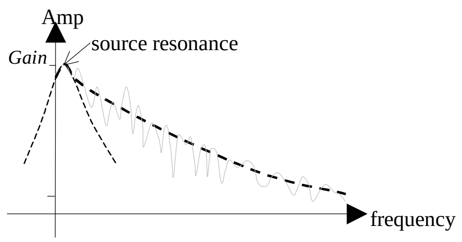


Figure 5. The EpR source resonance.

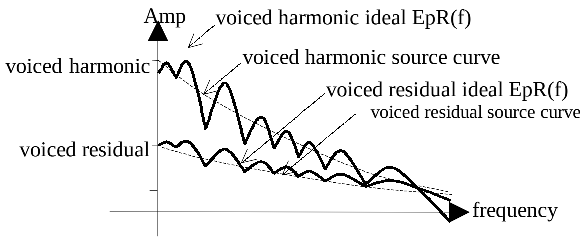

Amp
voiced harmonic ideal EpR(f)
voiced harmonic
voiced harmonic source curve
voiced residual ideal EpR(f)
voiced residual source curve
voiced residual
frequency

Figure 6. The EpR filter resonances.

The EpR vocal tract filter

The vocal tract is modeled by a collection of resonances plus a differential spectral shape envelope. These filter resonances are modeled in the same way as the source resonance, where the lower frequency resonances are somewhat equivalent to the vocal tract formants.

The EpR filters for voiced harmonic and residual excitations are basically the same, they just differ in the gain and slope depth parameters. This approximation has been obtained after comparing the harmonic and residual spectral shape of several SMS analysis of singer recordings. Figure 7 shows these differences.

The differential spectral shape envelope actually stores the differences (in dB) between the ideal EpR model (*iEpR*) and the real harmonic spectral shape (*HSS*) of a singer's performance. We calculate it as a 30 Hz equidistant step envelope.

$$DSS(f) = envelope_{i=0..}\left[30i, HSS_{dB}(30i) - iEpR_{dB}(30i)\right] \quad (4)$$

The EpR phase alignment

The phase alignment of the harmonics at the beginning of each period is obtained from the EpR spectral phase envelope. A time shift is applied just before the synthesis, in order to get the actual phase envelope (usually it will not match the beginning of the period). This phase alignment is then added to the voiced harmonic excitation spectrum phase envelope. The EpR spectral phase model states that each vocal tract resonance produces a linear shift of π on the flat phase envelope with a bandwidth depending on the estimated resonance bandwidth. This phase model is especially important for the intelligibility and in order to get more natural low pitched male voices.

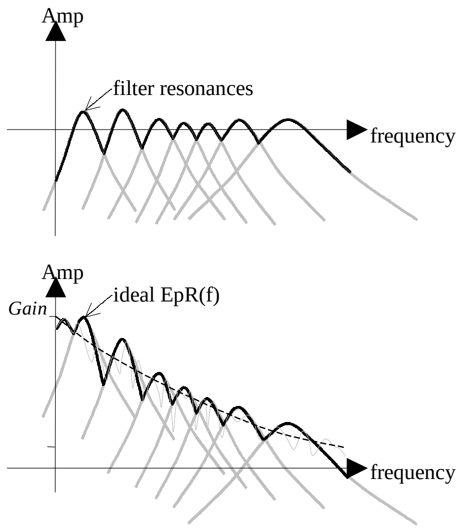


Figure 7. Differences between harmonic and residual EpR filters.

The EpR filter implementation

The EpR filters are implemented in the frequency domain. The input is the spectrum that results from the voiced harmonic excitation or from the voiced residual excitation. Both inputs are approximately flat spectrums, so we just need to add the EpR resonances, the source curve and the differential spectral shape to the amplitude spectrum. In the case of the voiced harmonic excitation we also need to add the EpR phase alignment to the phase spectrum.

For each frequency bin we have to compute the value of the EpR filter. This implies a considerable computational cost, because we have to calculate the value of all the resonances. However, we can optimize this process by assuming that the value of the sum of all the resonances is equal to the maximum amplitude (dB) of the whole filter and excitation resonances (over the source curve) at that bin. Then we can even do better by only using the two neighboring resonances for each frequency bin. This is not a low-quality approximation of the original method because the differential spectral shape envelope, which is always kept, takes care of all the differences between the model and the real spectrum.

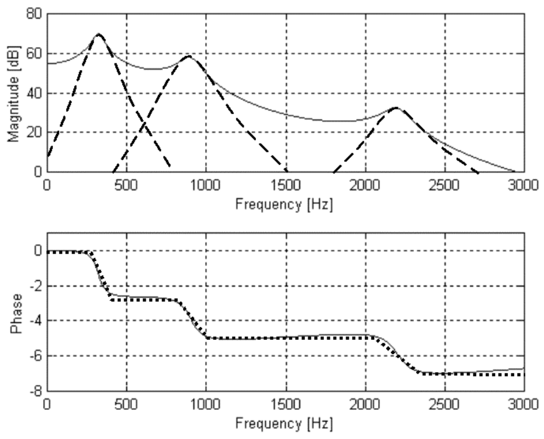


Figure 8. Phase alignment of the EpR resonances.

If we want to avoid the generation of the time domain voiced excitation, especially because of the computational cost of the fractional delay and the FFT, we can generate it in the frequency domain. From the pitch and gain input we can generate a train of deltas in frequency domain (sinusoids) that will be convolved with the transform of the synthesis window and then synthesized with the standard frame based SMS synthesis, using the IFFT and overlap-add method. However the voice quality may suffer some degradation due to the fact that the sinusoids are assumed to have constant amplitude and frequency throughout the frame duration.

The EpR filter transformation

We can transform the EpR by changing its parameters: excitation gain, slope and slope depth frequency, amplitude and bandwidth of the resonances. However, we have to take into account that the differential spectral shape is related to the resonances position. Therefore, if we change the frequency of the resonances we should stretch or compress the differential spectral shape envelope according to the resonances frequency change (using the resonances center frequency as anchor points).

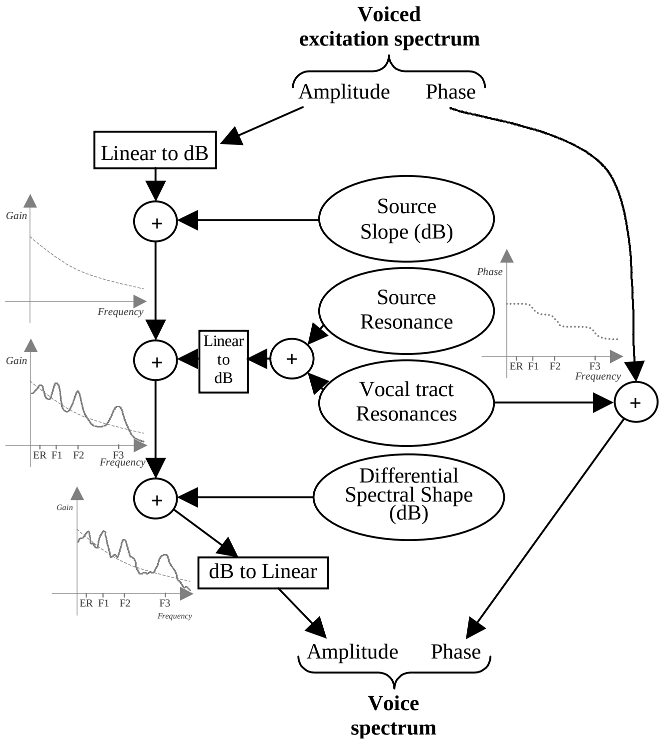


Figure 9. Frequency domain implementation of the EpR model.

4 System Overview

From the synthesis model presented in the previous section we have developed a complete singing voice synthesizer. The inputs to the system are the melody and the lyrics (in English or Japanese language), with its phonetic transcription in SAMPA format [SAMPA, 2000]. The system is able to read standard MIDI files as well as METRIX files (an ascii text file format designed as a sound synthesis control language [Amatriain, 1998]). The output of the system is a wave file with the synthesized singing voice.

The system is composed of two modules. The first one, Expressiveness module, controls the performance of the synthetic voice giving some expressiveness and naturalness to the input melody. The output of the Expressiveness module is a detailed musical score with all the needed features to characterize an expressive performance of a singer, which we call MicroScore. The second module, Synthesis module, is in charge of the actual synthesis process and uses a database containing the voice and timbre characteristics of a real singer. The analyzed voice data of the database is the result of the SMS analysis and the EpR modelization.

5 Musical Controls

To achieve naturalness on the output voice, the system defines a set of musically meaningful controls that are either related to individual notes (note parameters) or to the whole song (general control parameters). With these parameters the system attempts to cover as many situations as possible in a singing performance. In addition to the common musical parameters (pitch, duration and dynamics), the system uses other parameters to control vocal characteristics such as attack, release, vibrato, articulation between notes, etc. The specification of all these controls has been thought of with the end-user in mind, so as to be as easy as possible to control.

The table below shows a list of the note parameters as well as the general controls. These parameters are defined in the MicroScore structure:

Note parameters	**Observations**
Pitch	Midi number (0-127) or G#3
Begin Time	of the note, in milliseconds
Duration	of the note, in milliseconds
Loudness	Normalized value [0, 1]
Lyrics	Syllable associate in a note
Dynamic Envelope	(time, normalized value)
Pitch Envelope	(time, cents)
Attack Type	{normal, sharp, soft, high}
Attack Duration	Normalized value [0, 1]
Release Type	{normal, soft, high}
Release Duration	Normalized value [0, 1]
Transition Type	{legato, staccato, marcato, portamento, glissando}
Transition Duration	Normalized value [0, 1]
Vibrato Type	---
Vibrato Depth	Envelope (time, cents)
Vibrato Rate	Envelope (time, Hz)
Opening of vowels	Envelope
Hoarseness	Envelope
Whisper	Envelope

Control parameters	**Observations**
Singer Type	Change singer of the DB
Gender Type	Change gender (male/female)
Transposition	To transpose input melody

5.1 Expressiveness Controls

In order to obtain a good synthesis, it is essential to take into account the high-level expressive controls used by real performers. In our system, this is done by the

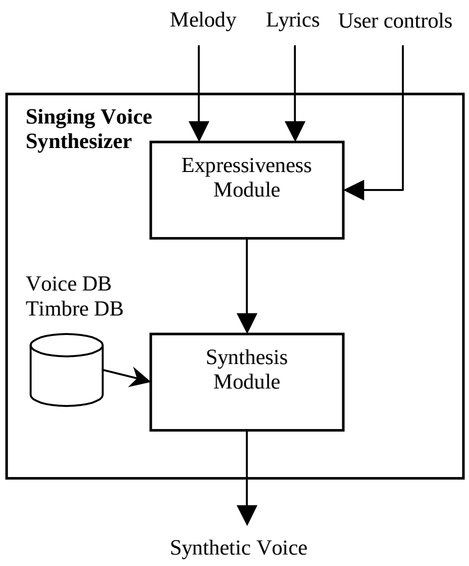


Figure 10. System overview.

expressiveness module, which is in charge of applying certain deviations to the input score with the purpose of making it more natural and expressive.

Following the research made by Sundberg and Friberg on musical expressiveness [Friberg, 1991], we have implemented a very basic rule-based system for expressive control. These rules were designed as a general tool for any kind of instruments, and need adaptations and proper tuning to be suitable for the singing voice [Berndtsson, 1996]. So far, only a few rules have been successfully tested in our system. We still need more experimentation to include rules specific to voice parameters such as vibrato, breathiness, hoarseness or special kinds of attack.

An important contribution to the expressive control of sound synthesis systems has been made by Manfred Clynes [Clynes, 1987]. In view of the fact that the human ear is specially sensitive to the amplitude contour of a note, Clynes has developed a mechanism to shape individually the amplitude of each note to give more authenticity to the synthesis. In this method, called *Predictive Amplitude Shaping*, there is a global amplitude contour for every note within a musical fragment which is slightly modified for each individual note according to the pitch interval to the next note. This creates a sense of continuity and phrasing. We have implemented and tested this technique in our system, adapting the parametric controls to the human voice, with excellent results.

The pitch contour of the singing voice has to be also carefully generated in order to obtain a faithful synthesis. So we have designed a mathematical model for reproducing the smooth pitch transitions between notes. This model allows us to control the transition duration and the tuning deviations at the end and the beginning of the notes in accordance with the musical context. In the note to note transitions, the synchronization between phonetics and musical rhythm is assured by reaching always the target pitch at the onset of the vowel of each syllable.

6 Singer Database

The creation of the database is one of the critical steps in most speech and singing voice synthesizers. There are no singing voice databases available, thus we recorded our own samples by first defining a set of musical exercises to be performed by a singer (i.e. different type of attacks, releases, note to note transitions, etc). Needless to say, the voice had to be recorded in a dry and noiseless environment to get the best possible SMS analysis.

In terms of what information is needed to be stored in the database, we can either attempt to record as many samples as possible (all possible pitches, attacks, …) or start from fewer samples and obtain the rest through transformations from the recorded/analyzed ones. This second approach gives more flexibility to the system at the possible expense of sound quality. Our approach has been this second one.

In order to choose the most useful English phonetic articulations to be included in the database, we ordered all possible articulations according to its frequency of use in actual songs. A statistical analysis was made from 76000 English songs, with close to three million words. With this information, now we know how many articulations are needed to cover a fixed percentage of all the possible articulations.

Number of articulations	Covered percentage
71	50 %
308	90 %
395	95 %
573	99 %
785	99.9 %
1129	100 %

The database is organized into two parts; timbre and voice DB.

6.1 Timbre DB

The timbre database stores the voice model (EpR) for each of the voiced phonemes. For each of these, we store several samples at different pitches and at different dynamics. When a phoneme with intermediate pitch or dynamics is required, the EpR parameters of the neighboring samples are interpolated to synthesize the phoneme at the desired pitch.

6.2 Voice DB

The voice database includes analysis data from the time varying characteristics of the voice in the form of templates. It is divided into several categories: steady states, phonetic articulations, note attacks, note-to-note articulations, note releases and vibratos. It is also possible to have different templates for the different pitches of the same type of sound.

Steady states

There are steady states stored for each of the phonemes. They model the behavior of the stationary part of a note. To do so, we store the time-varying evolution of the EpR parameters (if the phoneme is voiced) and the SMS residual along the steady state.

Phonetic articulations

All the articulations are segmented in two regions, the end of the first sound and the beginning of the next. If the regions are different in terms of voiceness, each region is analyzed with different specific parameters. Otherwise, if both regions are voiced or unvoiced the segmentation is used in synthesis to control the onset of the articulation. The EpR parameters are estimated only in the voiced part regions.

Vibratos

The vibrato templates characterize the vibrato characteristics by keeping the behavior of the voice excitation parameters. In particular, the fundamental frequency evolution, gain and source curve changes. Each time-varying function is segmented into attack, body and release. There can be different template labels according to some musical or expression classifications.

Note attacks, note transitions and note releases

These templates model the amplitude and fundamental frequency evolution at the note boundaries. They can easily be represented by a model (such as the one proposed for the amplitude by Clynes [Clynes, 1987]) which will give a smoother time evolution of the synthetic vocal sound. These templates, or models, are phoneme independent since they do not model the EpR changes. They can be organized into different expression categories.

7 Conclusions

With the purpose of demonstrating the potential of our system, two small databases with a male singer and a female singer have been created. With the female voice, we have synthesized fragments of two different songs ("We've only just begun", by The Carpenters, and "Natural woman", by Carol King). With the male voice, we have also synthesized the same song by The Carpenters and some choruses for accompanying the female songs.

The system evaluation was made by a group of musicians that had never heard about the project. Taking the Vocalwriter synthesizer as a comparison for the English synthesis, our demo songs were considered more intelligible and more natural. In contrast, our synthesis showed a lack of timbre uniformity. As the evaluation conclusion, although the synthesis obtained was not comparable with a real singer performance, the system was judged to be of a higher quality than the available commercial systems.

From our evaluation the system still presents important drawbacks. The most important one are that unnatural artifacts appear in the synthesis, specially in the voiced consonants phonemes. Also the low register timbres of a male voice suffer from unnaturalness and sharp attacks, specially the ones belonging to plosive phonemes are smeared. We can think of some solutions to these problems by incorporating some recently proposed enhancements to the sinusoidal modeling of musical sounds [Fitz, Haken, Christensen, 2000; Verma, Meng, 2000].

The creation of a complete voice database is for now a demanding process that has to be supervised by a user. More automated ways have to be developed to facilitate and speed up the database creation process.

Certainly there is room for improvement in every step of the system. Even so, assuming, naturalness as the essential feature for evaluating the quality of a singing synthesizer, results are promising and prove the suitability of our approach.

8 References

Amatriain, X. 1998. "METRIX: A Musical Data Definition Language and Data Structure for a Spectral Modeling Based Synthesizer," *Proceedings of 98 Digital Audio Effects Workshop.*

Berndtsson, G. 1996. "The KTH Rule System for Singing Synthesis," *Computer Music Journal, 20:1, 1996.*

Cano, P.; A. Loscos; J. Bonada; M. de Boer; X. Serra. 2000. "Voice Morphing System for Impersonating in Karaoke Applications," *Proceedings of the 2000 International Computer Music Conference*. San Francisco: Computer Music Association.

Childers, D. G. 1994. "Measuring and Modeling Vocal Source-Tract Interaction," *IEEE Transactions on Biomedical Engineering 1994.*

Clynes, M. 1987. "What can a musician learn about music performance from newly discovered microstructure principles (PM and PAS)?," *Action and Perception in Rhythm and Music. Royal Swedish Academy of Music No. 55, 1987.*

Cook, P. 1996. "Singing Voice Synthesis History, Current Work, and Future Directions," *Computer Music Journal, 20:2 1996.*

Fitz, K.; L. Haken; P. Christensen. 2000. "A New Algorithm for Bandwidth Association in Bandwidth-Enhanced Additive Sound Modeling," *Proceedings of the 2000 International Computer Music Conference*.

Friberg, A. 1991. "Generative Rules for Music Performance: A Formal Description of a Rule System," *Computer Music Journal 15:2, 1991.*

Klatt, D. H. 1980. "Software for a cascade/parallel formant synthesizer," *Journal of the Acoustical Society of America, 971-995, 1980.*

SAMPA, 2000. Speech Assessment Methods Phonetic Alphabet machine-readable phonetic alphabet. http://www.phon.ucl.ac.uk/home/sampa/home.htm

Serra, X.; J. Smith. 1990. "Spectral Modeling Synthesis: A Sound Analysis/Synthesis System based on a Deterministic plus Stochastic Decomposition,'' *Computer Music Journal* 14(4):12-24.

Smith, J.O.; P. Gosset. 1984. "A flexible sampling-rate conversion method," *Proceedings of the ICASSP, San Diego, New York, vol. 2, pp 19.4.1-19.4.2. IEEE Press 1984.*

SMARTTALK, 2000. SmartTalk 3.0. Japanese Speech-Synthesis Engine with Singing Capability. http://www.oki.co.jp/OKI/Cng/Softnew/English/sm.htm

Verma, T. S.; T. H. Y. Meng. 2000. "Extending Spectral Modeling Synthesis With Transient Modeling Synthesis" *Computer Music Journal* 24:2, pp.47-59.

VOCALWRITER, 2000. Vocalwriter. Music and Vocal synthesis. http://www.kaelabs.com/

Spectral Modeling for Higher-level Sound Transformations

Xavier Amatriain, Jordi Bonada, Alex Loscos, Xavier Serra
Music Technology Group, Pompeu Fabra University
{xavier.amatriain, jordi.bonada, alex.loscos, xavier.serra}@iua.upf.es,
http://www.iua.upf.es/mtg

Abstract

When designing audio effects for music processing, we are always aiming at providing higher-level representations that may somehow fill in the gap between the signal processing world and the end-user. Spectral models in general, and the Sinusoidal plus Residual model in particular, can sometimes offer ways to implement such schemes.

1 Introduction

When dealing with digital audio effects, we are looking for representations of sound signals and signal processing systems that can give us ways to design sound transformations in a variety of music applications and contexts.

The basic idea of spectral processing is that we can analyze a sound to obtain alternative frequency domain representations, which can then be transformed and inverted to produce new sounds (see Figure 1). Most of the approaches start by developing an analysis/synthesis system from which the input sound is reconstructed without any perceptual loss of sound quality. Then, the main issue is what the intermediate representation is and what parameters are available for applying the desired transformations.

By understanding the basic concepts of frequency domain analysis, we will be able to acquire the tools to use a large number of effects processors and to understand many types of sound transformations systems. Moreover, being the frequency domain analysis a somewhat similar process than the one performed by the human hearing system, it yields fairly intuitive intermediate representations.

Perceptual or musical concepts such as timbre or pitch are clearly related to the spectral characteristics of a sound. Some common processes for sound effects are also better explained using a frequency domain representation. We usually think on the frequency axis when we talk about equalizing, filtering, pitch shifting, harmonizing... In fact, some of them are specific to this signal processing approach and do not have an immediate counterpart on the time domain. Another issue is whether or not this approach is the most efficient, or practical, for a given application. The process of transforming a time domain signal into a frequency domain representation is, by itself, not an immediate step. Some parameters are difficult to adjust and force us to take several compromises. Some settings, such as the size of the analysis window, have little or nothing to do with the high-level approach we intend to favor, and require the user to have a basic signal processing understanding.

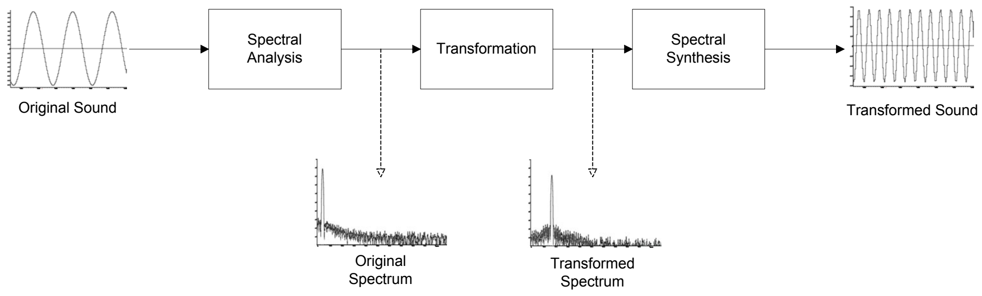


Fig 1. Block diagram of a simple spectral analysis

In that sense, when we talk about higher level spectral processing we are thinking of an intermediate analysis step in which relevant features are extracted or computed from the spectrum. These relevant features should be much closer to a musical or high-level approach. We can then process the features themselves or even apply transformations that keep some of the features unchanged. For example, we can extract the fundamental frequency and the spectral shape from a sound and then modify the fundamental frequency without affecting the shape of the spectrum. [Serra and Bonada 98]

In this section, we will briefly mention the Sinusoidal Model and will concentrate in the Sinusoidal plus Residual Model. Anyhow, the decision as to what spectral representation to use in a particular situation is not an easy one. The boundaries are not clear and there are always compromises to take into account, such as: (1) sound fidelity, (2) flexibility, (3) coding efficiency, and (4) computational requirements. Ideally, we want to maximize fidelity and flexibility while minimizing memory consumption and computational requirements. The best choice for maximum fidelity and minimum compute time is the STFT that, anyhow, yields a rather inflexible representation and inefficient coding scheme. Thus our interest in finding higher-level representations as the ones we present in this section.

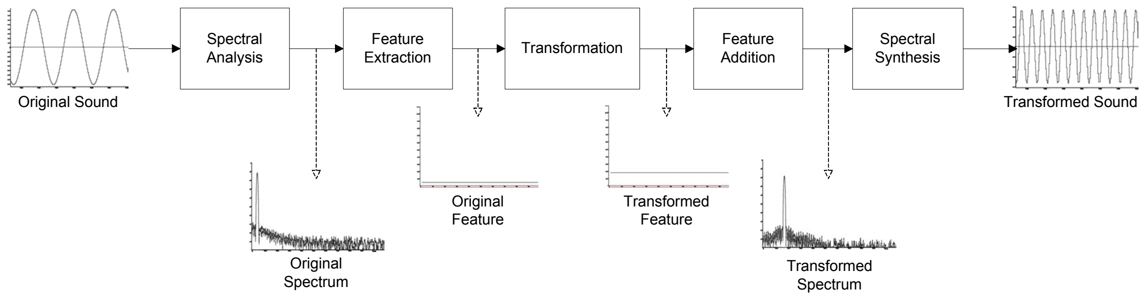


Fig 2. Block diagram of a higher-level spectral processing framework

Assuming the fact that there is no single representation and processing system optimal for everything, our approach will be to present a set of complementary spectral models that can be combined to be used for the largest possible set of sounds and musical applications.

In the next sections, we introduce two spectral models: Sinusoidal and Sinusoidal plus Residual. These models already represent a step up on the abstraction ladder and, from either of them, we can identify and extract higher-level information of a sound, such as: harmonics, pitch, spectral shape, vibrato, or note boundaries, that is Higher Level Features. This analysis step brings the representation closer to our perceptual understanding of a sound. The complexity of the analysis will depend on the type of feature that we want to identify and the sound to analyze. The benefits of going to this higher level of analysis are enormous and open up a wide range of new musical applications.

In section 3 we will provide a set of basic audio effects and transformations based on the implemented Sinusoidal plus Residual analysis/synthesis. We will finish with an explanation of content dependant processing implementations: a real-time singing voice conversion application that has been developed for use in Karaoke applications, and a nearly loss less Time Scaling algorithm.

2 Spectral Models

The most common approach for converting a time domain signal into its frequency domain representation is the Short-Time Fourier Transform (STFT). It is a general technique from which we can implement loss-less analysis/synthesis systems. Many sound transformation systems are based on direct implementations of the basic algorithm.

2.1 Sinusoidal Model

Using the output of the STFT, the Sinusoidal model represents a step towards a more flexible representations while compromising both sound fidelity and computing time. It is based on modeling the time-varying spectral characteristics of a sound as sums of time-varying sinusoids. The input sound is modeled by,

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

where $A_r(t)$ and $\theta_r(t)$ are the instantaneous amplitude and phase of the r^{th} sinusoid, respectively. [McAulay and Quatieri 86; Smith and Serra 87].

To obtain a sinusoidal representation from a sound, an analysis is performed in order to estimate the instantaneous amplitudes and phases of the sinusoids. This estimation is generally done by first computing the STFT of the sound, then detecting the spectral peaks (and measuring the magnitude, frequency and phase of each one), and finally organizing them as time-varying sinusoidal tracks.

It is a quite general technique that can be used in a wide range of sounds and offers a gain in flexibility compared with the direct STFT implementation.

2.2 Sinusoidal plus Residual Model

The Sinusoidal plus Residual model can cover a wide "compromise space" and can in fact be seen as the generalization of both the STFT and the Sinusoidal models. Using this approach, we can decide what part of the spectral information is modeled as sinusoids and what is left as STFT. With a good analysis, the Sinusoidal plus Residual representation is very flexible while maintaining a good sound fidelity and the representation is quite efficient. In this approach, the Sinusoidal representation is used to model only the stable partials of a sound. The residual, or its approximation, models what is left, which should ideally be a stochastic component. This model is less general than either the STFT or the Sinusoidal representations but it results in an enormous gain in flexibility [Serra 96; Serra and Smith 90].

The input sound $s(t)$ is modeled by,

$$s(t)=\sum_{r=1}^{R} A_r(t)\cos\left[\theta_r(t)\right]+e(t) \quad (2)$$

where $A_r(t)$ and $\theta_r(t)$ are the instantaneous amplitude and phase of the r^{th} sinusoid, respectively, and $e(t)$ is the noise component at time t (in seconds).

The sinusoidal plus residual model assumes that the sinusoids are stable partials of the sound with a slowly changing amplitude and frequency. With this restriction, we are able to add major constraints to the detection of sinusoids in the spectrum and omit the detection of the phase of each peak. The instantaneous phase that appears in the equation is taken to be the integral of the instantaneous frequency $\omega_r(t)$, and therefore satisfies

$$\theta_r(t)=\int_0^t \omega_r(\tau)d\tau \quad (3)$$

where $\omega(t)$ is the frequency in radians, and r is the sinusoid number. When the sinusoids are used to model only the stable partials of the sound, we refer to this part of the sound as the deterministic component.

Within this model we can either leave the residual signal, $e(t)$, to be the difference between the original sound and the sinusoidal component, resulting into an identity system, or we can assume that $e(t)$ is a stochastic signal. In this case, the residual can be described as filtered white noise,

$$e(t)=\int_0^t h(t,\tau)u(\tau)d\tau \quad (4)$$

where $u(t)$ is white noise and $h(t,\tau)$ is the response of a time varying filter to an impulse at time t . That is, the residual is modeled by the time-domain convolution of white noise with a time-varying frequency-shaping filter.

The implementation of the analysis for the Sinusoidal plus Residual Model is more complex than the one for the Sinusoidal Model. Figure 3 shows a simplified block- diagram of this analysis.

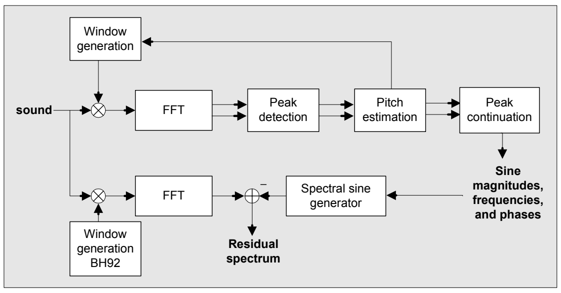


Fig 3. Block diagram of the Sinusoidal plus residual analysis.

The first few steps are the same than in a sinusoidal-only analysis. The major differences start on the peak continuation process since in order to have a good partial-residual decomposition we have to refine the this process in such a way as to be able to identify the stable partials of the sound.

The residual component is obtained by first generating the sinusoidal component with additive synthesis, and then subtracting it from the original waveform. A spectral analysis of this time domain residual is done by first windowing it using a window which is independent of the one used to find sinusoids, and thus we are free to choose a different time-frequency compromise. Then the FFT is computed and the resulting spectrum can be modeled using several existing techniques.

The original sinusoidal plus residual model has led to other different spectral models that still share some of its basis. [Ding and Qian, 97; Fitz, Haken and Christensen, 00; Verma,00]

2.2.1 Feature analysis

The accomplishment of a meaningful parameterization for sound transformation applications is a difficult task. We want a parameterization that offers an intuitive control over the sound transformation process, with which we can access most of the perceptual attributes of a sound. The analysis techniques described so far result in a simple parameterization, appropriate for describing the lower physical characteristics of the sound. In the Sinusoidal plus Residual model, these parameters are the instantaneous frequency, amplitude and phase of each partial and the instantaneous spectral characteristics of the residual signal.

There are other useful instantaneous attributes that give a higher-level abstraction of the sound characteristics. For example we can describe fundamental frequency, amplitude and spectral shape of sinusoidal component, amplitude and spectral shape of residual component, and overall amplitude. These attributes are calculated at each analysis frame from the output of the basic Sinusoidal plus Residual analysis. Afterwards, some of them can be extracted.

From a digital effects designer point of view, the extraction of such attributes allows us to implement transformations that modify only one of those features without affecting the rest. A clear example is illustrated in Fig 2 where the fundamental frequency is extracted, multiplied by a scaling factor, and then incorporated back to the original spectral data.

Apart from the instantaneous, or frame, values, it is also useful to have parameters that characterize the time evolution of the sound. The time changes can be described by the derivatives of each one of the instantaneous attributes computed as follows:

$$(5) \qquad \Delta = \frac{Val(l) - Val(l-1)}{H/SR}$$

where $Val(l)$ is the attribute value for the current frame, $Val(l-1)$ is the attribute value for the previous one, H is the hop-size and SR the sampling rate.

Another important step towards a musically useful parameterization is the segmentation of a sound into regions that are homogeneous in terms of its sound attributes. Then we can identify and extract region attributes that will give higher-level control over the sound. For our purposes it is very valuable as a way to apply region dependent transformations. For example, a time stretching algorithm would be able to transform the steady state regions, leaving the rest unmodified.

Once a given sound has been segmented into regions we can compute the attributes that describe each one. Most of the interesting attributes are the mean and variance of each of the frame attributes for the whole region.

Global attributes that can characterize attacks and releases make use of the average variation of each of the instantaneous attributes, such as average fundamental frequency variation, average amplitude variation, or average spectral shape change. In the steady state regions it is important to extract the average value of each of the instantaneous attributes and measure other global attributes such as time-varying rate and depth of vibrato.

Some region attributes can be extracted from the frame attributes in the same way that these were extracted from the Sinusoidal plus Residual data. The result of the extraction of the frame and region attributes is a hierarchical multi-level data structure where each level represents a different sound abstraction.

From the basic sinusoidal plus residual representation it is possible to extract some of the attributes mentioned above. The critical issue is how to extract them while minimizing interferences, thus obtaining significant high level attributes free of correlations [Serra and Bonada 98]. The general process will be to first extract instantaneous attributes and their derivatives, then segment the sound based on that information, and finally extract region attributes.

2.2.2 Synthesis

From the output of the analysis techniques presented we can synthesize a new sound. The similarity with respect to the original sound will depend on how well the input sound fits the implicit model of the analysis technique and the settings of the different variables that the given technique has. In the context of this paper we are interested in transforming the analysis output in order to produce a specified effect in the synthesized sound.

All these transformations can be done in the frequency domain. Afterwards, the output sound can be synthesized using the techniques presented in this section. The sinusoidal component will be generated using some type of additive synthesis approach and the residual, if present, will be synthesized using some type of subtractive synthesis approach.

Thus, the transformation and synthesis of a sound is done in the frequency domain; generating sinusoids, noise, or arbitrary spectral components, and adding them all to a spectral frame. Then, we compute a single IFFT for each frame, which can yield efficient implementations.

Fig 4 shows a block diagram of the final part of the synthesis process. Previous to that, we have to transform and add all the High Level Features, if they have been extracted, and obtain the lower level data (sine and residual) for the frame to be synthesized. Since the stored data might have a different frame rate, or a variable one, we may also have to generate the appropriate frame by interpolating the stored ones.

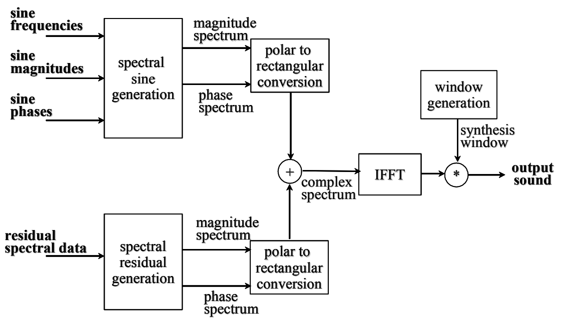


Fig.4. Diagram of the spectral synthesis.

3 FX and Transformations

In this section we intend to give a brief catalog of effects that can be implemented using the Sinusoidal plus Residual model and a transformation scheme like the one depicted in Figure 3.

3.1 Filtering with arbitrary resolution

Filters are probably the paradigm of a "classical" effect. Many different implementations are provided in the general DSP literature. Here we introduce a different approach that differs in many aspects from the classical one.

For our "filter" implementation, we take advantage of the sinusoidal plus residual model in order to modify the amplitude of any arbitrary partial present in the sinusoidal component.

For example, we can implement a band-pass filter defined by (x,y) points where x is the frequency value in Hertzs and y is the amplitude factor to apply. Thus, our filter does not need to be characterized by a traditional transfer function, and a more complex function can be defined by summing delta-functions. The transfer function is the defined by:

$$(6) \qquad H(f) = \sum \delta(f_i) \cdot g_i$$

where g_i is the gain applied to the i^{th} partial of frequency f_i.

Many applications derive from this sort of filtering scheme. For instance, we can filter out the even partials of a sound with a broadband spectrum, like a vocal sound, converting it to a clarinet-like sound.

3.2 Partial dependent frequency scaling

Another possible related effect that arises from the particularities of our model is the possibility to apply a frequency scaling to the sinusoidal components of our modeled sound, being able to process the residual or noisy component in a completely different way.

We can, for example, introduce a frequency shift factor to all the partials of our sound (see formula 7). Note, though, that if a constant is added to every partial of a harmonic spectrum, the resulting sound will be inharmonic.

$$(7) \qquad f_i = f_i + k$$

In the same way, we can scale all the partials multiplying them by a given scaling factor. Note that this effect will act as a pitch shifter without timbre preservation.

$$(8) \qquad f_i = f_i \cdot k$$

Another effect we can implement following this same idea is to add a stretching factor to the frequency of every partial. The relative shift of every partial can be computed, for exaple, depending on its original partial index, following the formula:

$$(9) \qquad f_i = f_i \cdot fstretch^{(i-1)}$$

with *i=1..N* where *N* is the number of sinusoids. This kind of stretching can be observed in a real piano sound. Thus, if we ever intended to create a piano synthesizer we could make use of this transformation.

3.3 Pitch Transposition with Timbre Preservation

Pitch transposition is the scaling of all the partials of a sound by the same multiplying factor. Here we introduce the concept of timbre preservation by leaving the spectral shape unmodified. For that reason we scale the frequency of each partial applying the original spectral shape.

The spectral shape of the sinusoidal component is the envelope described by the amplitudes and frequencies of the harmonics, or its approximation,

$$(10) \quad Sshape = \{(f_1, a_1)(f_2, a_2)...(f_N, a_N)\}$$

And so, the resulting sinusoidal spectrum $X_{transp}(f)$ after a transposition of value *k* will be of the form:

$$(11) \quad X_{transp}(f) = \sum \delta(k \cdot f_i) \cdot Sshape(k \cdot f_i)$$

formula that obviously implies that we must be able to find intermediate points of the spectral shape using some interpolation algorithm.

An overall more realistic effect is accomplished if we comb filter the original residual using the new pitch before we merge it with the transposed sinusoidal part.

3.4 Pitch Discretization to Temperate Scale

An interesting effect can be accomplished by forcing the pitch to take the nearest frequency value of the temperate scale. It is indeed a very particular case of pitch transposition where the pitch is cuantified to one of the 12 semitones in which an octave is divided. This effect is widely used on vocal sounds for dance music and is many times referred to with the misleading name of *vocoder effect*.

For a perfectly harmonic sound we have:

$$f_i = f_0 \cdot i \qquad (12)$$

with $i=1..N$ where N is the number of sinusoids. The frequency of the i^{th} harmonic is just i times the frequency of the fundamental f_0. To find the new fundamental frequency, we apply the following formula:

$$f_0' = 55 \cdot \left(\left(2^{\left(\frac{1}{12}\right)} \right)^{\left(round \left(\frac{12 \cdot \log\left(\frac{f_0}{55}\right)}{\log(2)} \right) \right)} \right) \qquad (13)$$

where 55 is the frequency in *Hz* that corresponds to an A0. From this new fundamental f_0', we can compute the transposition factor, defined as:

$$k = \frac{f_0'}{f_0} \qquad (14)$$

Finally, we only need to apply the pitch transposition algorithm defined in the previous section.

3.5 Vibrato and Tremolo

Vibrato and tremolo are common effects used in different kinds of acoustical instruments, including the human voice. Both are low frequency modulations: vibrato is applied to the frequency and tremolo to the amplitude of the partials.

In fact, vibrato usually implies a tremolo modulation. Both modulations share in this case the resonating frequency f_m. For example, to apply a vibrato with tremolo effect, we can modulate the fundamental frequency f_0 (as shown in formula 15):

$$f_0' = f_0 \cdot (1 + c \cdot \sin(2\pi \cdot f_m)) \qquad (15)$$

where c is the vibrato depth (usually around 75 cent). This would give us a transposition factor (formula 14) and we would apply the corresponding transposition with timbre preservation. For the tremolo we could apply the following modulation to each of the siunoids amplitude a_i:

$$a_i' = a_i + t(i) \cdot \frac{\sin(2\pi \cdot f_m) - 1}{2} \text{ (dB)} \qquad (16)$$

the modulation depth *t(i)* would apply different depths at different frequencies emulating the spectral tilt variations suffered in a real tremolo sound. This curve could be a sampled version of the curve shown in Fig.5 where *SR* is the sampling rate.

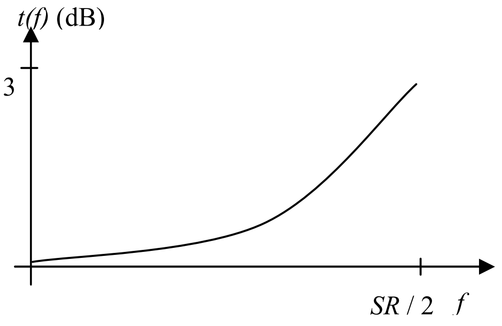


Fig. 5. *t(f)* curve

3.6 Spectral Shape Shift

Many interesting effects can be accomplished by shifting the spectral shape or spectral envelope of the sinusoidal component of a sound. This shift is performed in such a way that no new partials are generated, just the amplitude envelope of the spectrum is modified.

Such transformation results in a sinusoidal spectrum $X_{shifted}(f)$ such as:

$$X_{shifted}(f) = \sum \delta(f_i) \cdot Sshape(f_i - D) \qquad (17)$$

where D is the spectral shift in *Hz* (positive if we move the spectrum to the right and negative if we move it to the left) and *Sshape* is the original spectral shape and it is defined as in equation 10.

3.7 Gender Change

Using the results of 3.3 and 3.6 we can change the gender of a given vocal sound. Note how by combining different "basic" effects we are able to step higher in the level of abstraction and get closer to what a naive user could ask for in a sound transformation environment, such as: imagine having a gender control on a vocal processor...

We apply two transformations in order to convert a male voice into a female one. The first one is a pitch transposition an octave higher. The other one is a shift in the spectral shape. The theoretical explanation to this effect is that women change their formant (resonant filters) frequencies depending on the pitch. That is, when a female singer rises up the pitch, the formants move along with the fundamental.

To convert a female into a male voice we also apply a pitch transposition and a shift in the spectral shape. This shifting has to be applied in a way the formants of the female voice remain stable along different pitches

3.7 Harmonizer

In order to create the effect of a harmonizing vocal chorus, we can add pitch-shifted versions of the original voice (with the same timbre) and force them to be in tune with the original melody.

So for a number *H* of harmonies we have that the resulting sinusoidal spectrum can be described as:

$$X'(f) = X(f) + \sum_{h=1}^{H} X_{transp}(f,h) \quad (18)$$

where $X_{transp}(f,h)$ is the original sinusoidal spectrum transposed (with timbre preservation) by a factor that depends on *h*.

3.8 Hoarseness

Although hoarseness is sometimes thought of as a symptom of some kind of vocal disorder, this effect has been widely used by singers in order to resemble the voice of famous performers (Louis Armstrong or Tom Waits, for example). We can accomplish a similar effect through a very basic transformation by just applying a gain to the residual component of our analysis.

3.9 Morphing

Morphing is a transformation with which, out of two or more elements, we can generate new ones with hybrid properties.

Most of the interpolation techniques are based on the interpolation of sound parameterizations resulting from analysis/synthesis techniques, such as the Short-time Fourier Transform (STFT), Linear Predictive Coding (LPC) or Sinusoidal Models.

A first approach to such transformation could consist in interpolating two sinusoidal spectral shapes to obtain the hybrid $X'(f)$:

$$X'(f) = \alpha \cdot X_1(f) + (1-\alpha) \cdot X_2(f) \quad (19)$$

However, more musical meaningful results can be achieved in a more flexible way by moving the interpolation to a higher-level features plane, in which we could, for example interpolate between pitches:

$$f_0' = \alpha \cdot f_{01} + (1-\alpha) \cdot f_{02} \quad (20)$$

In both examples (equations 19 and 20) α would be the interpolation factor and would take values between 0 and 1.

4 Content dependent processing

The hierarchical data structure that includes a complete description of a given sound offers many possibilities for sound transformations. Modifying several attributes at the same time and at different abstraction levels achieve, as it has already been pointed out in the previous section, most musically or end-user meaningful transformations.

Higher-level transformations can refer to aspects like sound character, articulation or expressive phrasing. These ideas lead to the development of front ends such as graphical interfaces or knowledge-based systems [Arcos 98] that are able to deal with the complexity of this sound representation.

In this section we introduce two applications that have been developed with these ideas in mind: an automatic singing voice conversion application and a time scaling module.

4.1 Real-Time Singing Voice Conversion

Our automatic voice conversion application implements a very particular case of audio morphing, pursuing the possibility of morphing, in real-time, two singing voice signals in such a way we can control the resulting synthetic voice by mixing some characteristics of the two sources. Whenever this control is performed by means of modifying a reference voice signal matching its individuality parameters to another, we can refer to it as voice conversion.

In such a context, a karaoke-type application, in which the user can sing like his/her favorite singers, was developed [Cano et al 00]. The result is basically an automatic impersonating system that allows the user to morph his/her voice attributes (such as pitch, timbre, vibrato and articulations) with the ones from a prerecorded singer, which from now on we will refer to as target.

In this particular implementation, the target's performance of the complete song to be morphed is recorded and analyzed beforehand. In order to incorporate the corresponding characteristics of the target's voice to the user's voice, the system first recognizes what the user is singing (phonemes and notes), looks for the same sounds in the target performance (i.e. synchronizing the sounds), interpolates the selected voice attributes, and synthesizes the output morphed voice. All this is accomplished in real-time.

Fig 6 shows the general block diagram of the voice impersonator system. The system relies on two main techniques that define and constrict the architecture: the SMS framework (see 2.2) and a Hidden Markov Model based Speech Recognizer (SR). The SMS implementation is responsible of providing a suitable parameterization of the singing voice in order to perform the morph in a flexible and musical-meaningful way. On the other hand, the SR is responsible of matching the singing voice of the user with the target.

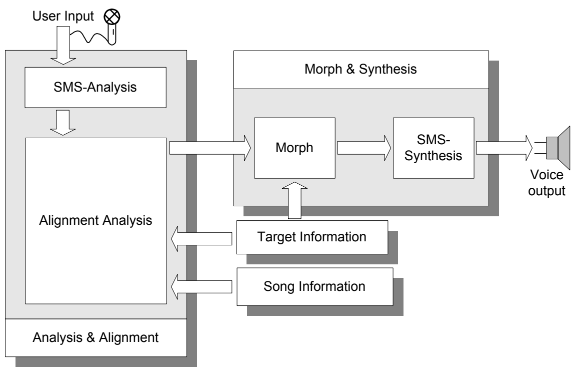


Fig.6. System block diagram

4.2 Time scaling

Time-scaling an audio signal means changing the length of the sound without affecting other perceptual features, such as pitch or timbre. Many different techniques, both in time and frequency domain, have been proposed to implement this effect. Some frequency domain techniques yield high-quality results and can work with large scaling factors. However, they are bound to present some artifacts, like phasiness, loss of attack sharpness and loss of stereo image. In this section we will present a frequency domain technique for near loss-less time-scale modification of a general musical stereo mix [Bonada 00].

The general block diagram of the system is represented in Fig 7. First, the input sound is windowed and goes through the FFT giving as a result the analysis frame (AFn), that is, the spectrum bins and the amplitude and phase envelopes. Then the time-scale module generates the synthesis frame (SFm) that is fed to the inverse FFT (IFFT). Finally, the Windowing&Overlap-Add block divides the sound segment by the analysis window and multiplies it by the overlap-add window, to reconstruct the output sound.

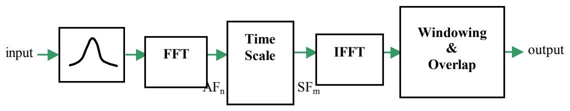


Fig. 7. General diagram

It is important to remark that the frame rate used in both the analysis and synthesis modules is the same, as opposed to the most broadly used time-scale techniques in which a change of frame rate in synthesis is used in order to achieve the effect. The window size and type must also be the same in both processes.

In some cases, an analysis frame is used twice (or more) while on other cases some frames are never used. This technique will not add any artifacts, provided the frame size we use is small enough and the sound does not present abrupt changes in that particular region. In the case of a percussive attack, though, a frame repetition or omission can be noticed regardless the analysis frame size. Therefore, some knowledge of the sound segment features is needed to decide where this technique can or cannot be applied.

In Fig 8, a more detailed block diagram of the time-scale module is depicted. The analysis frames (AFn), containing the spectrum amplitude and phase envelopes, are fed to the time-scaling module. This module performs a peak detection and a peak continuation algorithm on the current and previous (Z-1) amplitude envelopes. Then, only the peaks that belong to a sinusoidal track are used as inputs to the spectrum phase generation module. Note that the time-scale module only changes the phase, leaving the spectral amplitude envelope as it is.

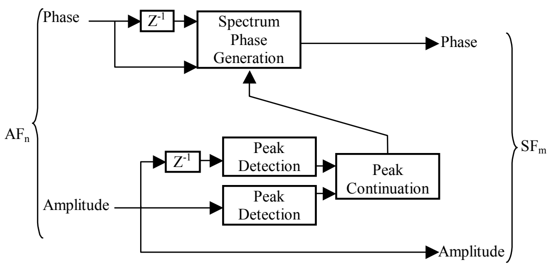


Fig. 8. The time-scale module

The phase of each peak is computed supposing that the frequency varies linearly between two consecutives frames and that there is some phase deviation. The usage of the same frame rate in analysis and synthesis allows us to suppose that the phase variation between two consecutives frames is also the same.

5 Conclusions

Throughout this paper, we have shown how the use of higher-level spectral models can lead to new and interesting sound effects and transformations. We have also seen that it is not easy nor immediate to get a good spectral representation of a sound, so the usage of this kind of approach needs to be carefully considered bearing in mind the application and the type of sounds we want to process.

For example, most of the techniques here presented work well only on monophonic sounds and some rely on the pseudo-harmonicity of the input signal.

Nevertheless, the use of spectral models for musical processing has not been around too long and it has already proven useful for many applications, as the ones presented in this paper. Under many circumstances, higher-level spectral models, such as the Sinusoidal plus Residual, offer much more flexibility and processing capabilities than more immediate representations of the sound signal.

In general, higher-level sound representations will offer more flexibility at the cost of a more complex and time-consuming analysis process. It is important to remember that the model of the sound we choose will surely have great effect on the kind of transformations we will be able to achieve and on the complexity and efficiency of our implementation.

6 References

Arcos, J. L.; R. López de Mántaras; X. Serra. 1998. "Saxex: a Case-Based Reasoning System for Generating Expressive Musical Performances", Journal of New Music Research, Vol. 27, N. 3, Sept. 1998.

Bonada, J. 2000. “Automatic technique in frequency domain for near-lossless time-scale modification of audio.” Proceedings of the 2000 International Computer Music Conference. San Francisco: Computer Music Association.

Cano, P.; A. Loscos; J. Bonada; M. de Boer; X. Serra; 2000. “Voice Morphing System for Impersonating in Karaoke Applications.” Proceedings of the 2000 International Computer Music Conference. San Francisco: Computer Music Association.

Ding Y.; X. Qian. 1997. "Sinusoidal and Residual Decomposition and Residual Modeling of Musical Tones Using the QUASAR Signal Model". Proceedings of the 1997 International Computer Music Conference. San Francisco: Computer Music Association.

Fitz, K.; L. Haken, and P. Christensen. 2000. “A New Algorithm for Bandwidth Association in Bandwidth-Enhanced Additive Sound Modeling”. Proceedings of the 2000 International Computer Music Conference. San Francisco: Computer Music Association.

McAulay, R. J. and T. F. Quatieri. 1986. “Speech Analysis/Synthesis based on a Sinusoidal Representation.” IEEE Transactions on Acoustics, Speech and Signal Processing 34(4):744--754.

Serra, X. 1987 A System for Sound Analysis/Transformation/Synthesis based on a Deterministic plus Stochastic Decomposition. Ph.D. Dissertation, Stanford University.

Serra, X. and J. Smith. 1990. “Spectral Modeling Synthesis: A Sound Analysis/Synthesis System based on a Deterministic plus Stochastic Decomposition.’’ Computer Music Journal 14(4):12--24.

Serra, X. 1996. “Musical Sound Modeling with Sinusoids plus Noise”, in G. D. Poli, A. Picialli, S. T. Pope, and C. Roads, editors, Musical Signal Processing. Swets & Zeitlinger Publishers.

Serra, X.; J. Bonada. 1998. "Sound Transformations Based on the SMS High Level Attributes", Proceedings of the 98 Digital Audio Effects Workshop.

Smith, J.O. and X. Serra. 1987. “PARSHL: An Analysis/Synthesis Program for Non-Harmonic Sounds based on a Sinusoidal Representation.” Proceedings of the 1987 International Computer Music Conference. San Francisco: Computer Music Association.

Verma, T. S. ; T. H. Y. Meng. 2000. “Extending Spectral Modeling Synthesis With Transient Modeling Synthesis”, Computer Music Journal 24:2, pp.47-59.

SAMPLE-BASED SINGING VOICE SYNTHESIZER USING SPECTRAL MODELS AND SOURCE-FILTER DECOMPOSITION

J. Bonada[1], A. Loscos[1], O. Mayor[1], H. Kenmochi[2]
[1] Music Technology Group, Audiovisual Institute, Universitat Pompeu Fabra, Barcelona, Spain
[2]Advanced System Development Center, YAMAHA corporation, Hamamatsu, Japan

Abstract: **This paper is a review of the work contained in the insides of a sample-based virtual singing synthesizer. Starting with a narrative of the evolution of the techniques involved in it, the paper focuses mainly on the description of its current components and processes and its most relevant features: from the singer databases creation to the final synthesis concatenation step.**

I. INTRODUCTION

The voice generation is typically explained as a source/filter system, in which a voiced/unvoiced excitation is filtered by the vocal tract resonances. The voiced excitation corresponds to the glottal pulses that originate the vocal fold vibrations whether the unvoiced excitation corresponds to the turbulent airflow that arises from the lungs. The voice filter is characterized by a set of resonances called formants that have their origin in the voice organs lengths and shapes (trachea, esophagus, larynx, …). This filter modulates the timbre of the sound by dynamically changing the amplitude, center frequencies and bandwidths of the resonances by moving the voice organs.

Some of the singing synthesizers developed since the beginnings of such discipline have focused in the source/filter decomposition (physical models based); others, rather than focusing on how the sound is produced, have focused on the perception of the sound (spectral models based); and others, such as the synthesizer we present in this paper, have tried to combine both models.

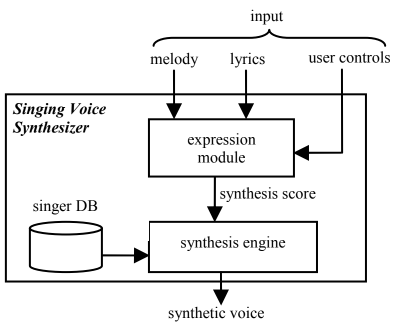


Figure 1: *General system diagram*

The system can be roughly described by a singer database, an input, an expression module and a synthesis engine (see Fig. 1). The input contains the melody and the lyrics of the song plus some expression controls. The expression module converts this input into an internal low-level synthesis score, and the synthesis engine reads this synthesis score, fetches the required samples from the singer database and transforms and concatenates them to obtain the synthetic output signal.

II. VOICE AND SPECTRAL MODELING

Since our system is a sample based synthesizer in which samples of a singer database are transformed and concatenated along time to compose the resulting audio, we have always considered the task of finding the most appropriate and the highest quality transformation techniques a crucial issue.

We initially used SMS [1] as the basic transformation technique with the addition of a time domain delta-based excitation to mimic the singer's voiced excitation [2]. SMS had the advantage of decomposing the voice into harmonics and residual. Both components were independently transformed, so the system yielded a great flexibility. But although the results were quite encouraging in voiced sustained parts, in transitory parts and consonants, especially in voiced fricatives, harmonic and residual components were not perceived as one.

Intending to improve our results, we moved to a spectral technique based on the phase-locked vocoder [3] where the magnitude spectrum is segmented into regions, each of which contains a spectral peak and its surroundings. These regions can be then freely shifted in amplitude and frequency. Regarding the phase spectrum, the relation between harmonics found at the beginning of each glottal period is kept after transformations [4]. On top of this technique we developed a frequency domain voice model that consists of an excitation curve, a set of resonances and a residual envelope. We call it EpR (Excitation plus Resonances) [2]. The excitation curve models the voiced source using an exponential decay function and a low frequency resonance. The vocal tract is modeled using the rest of the resonances and the

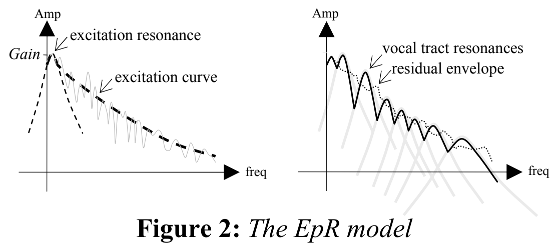


Figure 2: *The EpR model*

residual envelope stores the differences between the model and the spectral shape defined by the harmonics (see Fig. 2).

III. Singer database

About two hours of dry singer performance recordings are required to build a database. The singer is asked to follow a detailed recording script that covers most possible phonetic contexts and several expression aspects [5]. These recordings are then segmented and analyzed using the spectral analysis algorithms. In order to speed up this process two free software toolkits [6, 7] are used as phonetic aligners between the audio files and the recording scripts. The resulting data fills the phonetic and the expression libraries and is stored in a set of files organized in tree structured folders.

The phonetic database contains timbres, steady-states and articulations. The timbre section stores the voice model (EpR) of different vowels at different pitches and dynamics, the steady-state section contains long sustained vowels at different pitches, and the articulation section contains an organized list of diphonemes samples at different pitches.

The expression database contains note and vibrato templates intended to keep some basic expression aspects of the singer's voice and therefore increase the naturalness of the synthesis. Note templates model singer's attacks, releases and transition behaviors in different musical and intentional contexts. These contexts are described by a set of meaningful labels, like sharp attack, legato transition or sexy release. Each template stores a set of controls (pitch, loudness, EpR excitation curve, breathiness, roughness) obtained from the analysis of the sample, each of which can be later used in synthesis to reproduce the voice excitation changes for each expressive context. Vibrato templates store the singer's excitation behavior for different types of vibrato and tremolos; basically they keep the pitch and the EpR excitation curve. Each template is segmented into attack, body and release parts. The body segment is mirror-looped at synthesis if needed.

IV. Input score

The input score is an ASCII text file based on the METRIX control language [8] that contains the score of the song. Not only lyrics and notes can be specified, but also high level controls and all the possible music information that the system is capable to interpret. To achieve naturalness in the synthetic voice, the system defines some musically meaningful controls [5]. The idea is to cover the maximum situations that can appear in a real singing performance in order to avoid a lack of expression control that could bring about non-natural results.

The input score contain the so-called note parameters and control parameters. The note parameters refer to a specific note of the score and describe note attributes such as pitch, duration, loudness, lyrics, dynamics, vibrato, attack / release types, roughness, etc., while the control parameters refer to the whole song and describe song attributes such as singer, tempo, etc. Below you can see an example of input score where the lyrics are *fly me*.

```
Score_Info
{
    Tempo: 90
    Meter: 4/4
}
InstrumentInfo { Robert }

begin
  t1     Robert   NoteNumber: Ab2
                  Duration: t0.5
                  Lyrics: "f l aI"
                  Loudness: 0.6
                  AttackType: "soft"
  t1.5   Robert   NoteNumber: G2
                  Duration: t1
                  Lyrics: "m I"
                  Loudness: 0.3
                  VibratoType: "wet"
                  VibratoRate: [(0,1)(1,0.6)]
                  VibratoDepth: [(0,0) (0.5,1)(1,0.7)]
                  ReleaseType: "long"
end
```

V. Building the synthesis score

The expression module generates an internal low-level score (*synthesis score*) out of the input METRIX. This score is structured into several tracks and control envelopes, some of which are shown in Fig. 3. The phonetic track shows the articulations and steady-states to be fetched from the DB and their corresponding durations, which are calculated trying to make them as close to the original database sample durations as possible. The note and vibrato tracks contain information on the note and vibrato templates that must be applied at synthesis and their corresponding durations. The envelope controls (*vibrato depth and rate, pitch, pitch var, loudness, etc*) express their behavior along the performance with a time-varying function.

In addition to the note and vibrato templates, several models have been created to cover a wide variety of possibilities. However, templates extracted from real recordings are preferable to get a more authentic expressivity, although they may not sound natural when the synthesis context in which they are applied is far from the template context.

The phonetic track is filled out taking into account that the vowel onset should match the begin time of the note. Besides, as already mentioned, taking the original sample duration is preferable since this way we avoid time-scaling transformations, but this is not always possible because all required articulations must fit into the note segment. On the other hand, whenever the added duration of the articulations is less than the target note segment duration, a steady-state is added to fill out what is left.

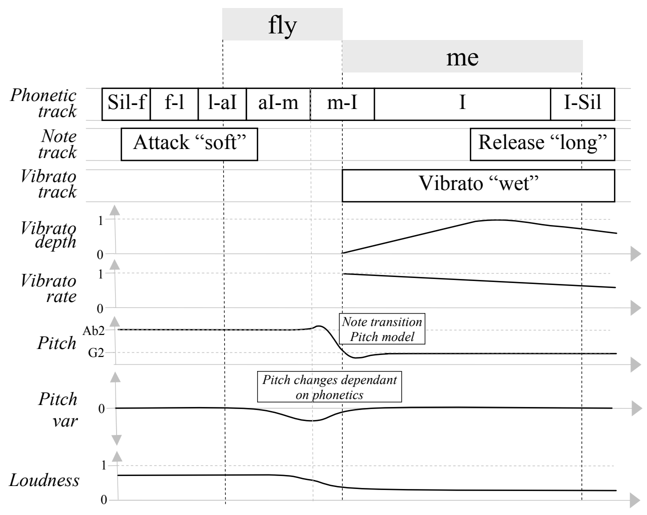


Figure 3: *Synthesis score*

In the synthesis score there are two envelope controls that specify the output synthesis pitch. The first envelope (*Pitch*) stores the absolute pitch values that come out from the notes specified by the input score. On the other hand *Pitch var* stores relative pitch variations due to changes originated by some phonetic combinations, such as certain voiced consonant - vowel combinations (b-a) in which the pitch decreases during the consonant sound.

In synthesis, the relative values of the *pitch var* envelope and the expression templates are added together to the absolute pitch values. In the case that an attack or release template is specified, the pitch variations of this template are applied when synthesizing to obtain a pitch curve similar to the one in the template. In the case of note transitions, the process is the same but whenever no template is specified, a pitch model is applied that overwrites the absolute pitch track of the score, like shown in Fig. 3, so to avoid pitch discontinuities. This pitch model has to be carefully generated to obtain a natural sounding pitch curve in the output synthesis. A mathematical model has been designed to produce smooth pitch transitions between notes and allow the control of some parameters like duration, shape and synchronization to phonetics and musical rhythm. This synchronization is basically attained by reaching the target pitch at the onset of the vowel of each syllable. In Fig. 4 we can see a more detailed drawing of this pitch model. The distance between *begin pitch* and *max pitch*, as well as between *min pitch* and *end pitch*, depends on the note interval (the bigger the interval, the bigger the distance, but with some limitations for big intervals). On the other hand, the transition curvature depends on both the note interval and the transition duration and its slope is restricted to a maximum value in order to guarantee smooth pitch variations in short transitions.

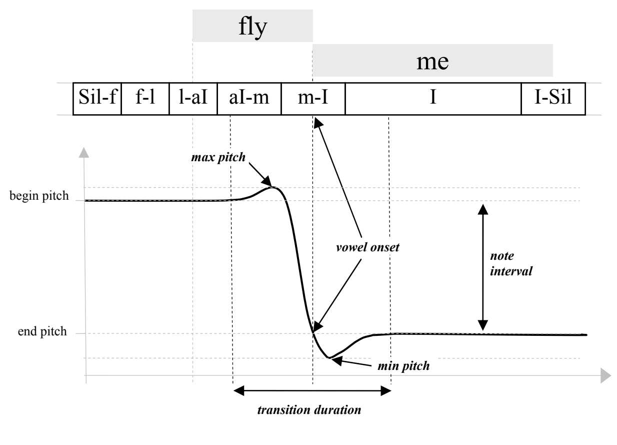


Figure 4: *Pitch model for note transitions*

VI. Synthesis engine

A. Sample transformations

The synthesis engine reads the synthesis score and retrieves the required samples and templates from the singer database selecting those units that are closer to the synthesis context (mainly pitch is considered). Once we have retrieved the samples, some transformations [4] are applied to match the synthesis score: transposition, equalization, time-scaling, loudness modification, vibrato and voice excitation based transformations. Finally, the transformed samples are concatenated to compose the resulting synthetic performance**.**

Transposition is applied to match the synthesis score pitch. Therefore, the transposition factor is calculated as the synthesis pitch divided by the sample pitch. This factor is calculated frame by frame. In terms of the spectral technique, harmonic peak's regions are shifted in frequency and harmonic peak's phases are corrected without altering the phase synchronization between harmonics.

Equalization is used to obtain transformations on timbre. When transposing, it is used to keep the original timbre but it can be applied as well to get generic timbre transformations. Equalization is achieved by shifting in amplitude the harmonic peak's regions so to match the desired timbre envelope.

Time-Scaling is applied to samples in order to match their durations with the synthesis score durations. The time-scale ratio is sometimes applied in a non-uniform way so that the synchronization between control parameters, phonetic and note tracks is not altered. For example, the phonetic articulation that contains the vowel onset should not change the timing of the vowel onset. Besides, in the case of abrupt phonetic changes, these should not be smoothed so not to degrade the intelligibility. The transformation is obtained by repeating or dropping some frames and interpolating them [9].

For loudness modification, database samples are considered to be sung at normal loudness, unless otherwise specified. Thus, sample loudness is changed to match the synthesis score value. The transformation can be achieved by applying an equalization filter obtained from a recorded template where the singer sang a crescendo or a decrescendo. This filter represents the timbre envelope differences between the sample estimated loudness and the target loudness.

For vibrato transformation, the pitch and EpR excitation changes enclosed in the vibrato template are applied to the audio samples. The little nuances of the singer's vibrato are kept even after altering its depth and rate, and the EpR voice model allows the harmonics to follow the resonances as their frequency is modified, thus emulating the real situation.

Besides, some voice excitation based transformations can be applied to improve the naturalness and expressiveness of the synthetic voice, such as roughness, whisper and breathiness. Roughness is obtained by adding sinusoids to the spectrum in a way that glottal periods become irregular. Whisper comes out of equalizing an unvoiced excitation with the timbre envelope. Finally, breathiness is succeeded by adding together whisper and equalization effects and lowering the harmonic's peak adjoining bins.

B. Sample concatenation

The last step in the synthesis engine is the concatenation of samples. Once we have transformed the samples, we have to deal with the spectral shape and phase discontinuities that appear when connecting them. With the aim of minimizing such discontinuities, amplitude and phase corrections are spread out along a set of transition frames that surround the boundary [4]. The results are quite smooth and good enough in most cases. Sometimes, however, a gap in brightness can be heard, especially when connecting samples that have been transposed with rather different factors, due to the fact that although there are no harmonic peak's amplitude or phase discontinuities, there do exist harmonic peak's regions amplitude and phase shape gaps. This problem is inherent to only consider harmonic peak's discontinuities when connecting samples, thus our algorithm should be expanded to consider inside region characteristics.

VII. Conclusion

The system we present is able to generate synthetic performances with quite successful results. However, the more different from the database the synthesizer is asked to sing, the more artificial synthesis gets (it is difficult to make the system sing hip-hop using an opera singer database). Some of this difficulty arises from the fact that the synthesizer has been thought to preserve not only the timbre personality of the singer from which the database is created but also his/her expressivity.

In this sense, work has to be done to improve transformations naturalness, especially when the synthesis context is far from the original context in which the sample that is being transformed was recorded.

Other improvements directions include working on expression dependent timbre transformations and getting into a higher level transformation description in which the system could generate an expressive performance automatically out of the melody, the lyrics, the singer, and an expressive label such as sweet or aggressive.

References

[1] Serra, X, "A system for sound analysis-transformation-synthesis based on a deterministic plus stochastic decomposition" *PhD thesis,* CCRMA, Dept. of Music, Stanford University, 1989.

[2] Bonada, J., Loscos, A., Cano, P., and Serra, X, "Spectral Approach to the Modeling of the Singing Voice" *Proceedings of the 111th AES Convention*, New York, USA, 2001.

[3] Laroche, J. and Dolson, M., "New phase-vocoder techniques for pitch-shifting, harmonizing and other exotic effects" *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, New Paltz, New York, 1999.

[4] Bonada, J., Loscos, A., and Kenmochi, K, "Sample-based singing voice synthesizer by spectral concatenation" *Proceedings of the Stockholm Music Acoustics Conference SMAC03*, Stockholm, Sweden, 2003.

[5] Bonada, J., Celma, O., Loscos, A., Ortolà, J., and Serra, X., "Singing Voice Synthesis Combining Excitation plus Resonance and Sinusoidal plus Residual Models" *Proceedings of the International Computer Music Conference*, Havana, Cuba, 2001.

[6] Akinobu Lee et al, "Julius - An Open Source Real-Time Large Vocabulary Recognition Engine" *Proceedings of the European Conference on Speech Communication and Technology (EUROSPEECH)*, pp. 1691-1694, 2001.

[7] Huang, X., Alleva, F., Hon, H., Hwang, M., and Rosenfeld, R., "The SPHINX-II speech recognition system: an overview" *Computer Speech and Language*, vol. 7, no. 2, pp. 137-148, 1993.

[8] Amatriain, X, "METRIX: A Musical Data Definition Language and Data Structure for a Spectral Modeling Based Synthesizer" *Proceedings of 98th Digital Audio Effects Workshop DAFX98*, Barcelona, Spain, 1998.

[9] Bonada, J., "Automatic Technique in Frequency Domain for Near-Lossless Time-Scale Modification of Audio" *Proceedings of the International Computer Music Conference*, Berlin, Germany, 2000.

Sample-based singing voice synthesizer by spectral concatenation

Jordi Bonada, Alex Loscos
Music Technology Group, Audiovisual Institute, Universitat Pompeu Fabra
Barcelona, Spain
{jordi.bonada, alex.loscos}@iua.upf.es
http://www.iua.upf.es/mtg

Hideki Kenmochi
Advanced System Development Center, YAMAHA corporation
Hamamatsu, Japan
kenmochi@beat.yamaha.co.jp

ABSTRACT

The singing synthesis system we present generates a performance of an artificial singer out of the musical score and the phonetic transcription of a song using a frame-based frequency domain technique. This performance mimics the real singing of a singer that has been previously recorded, analyzed and stored in a database. To synthesize such performance the systems concatenates a set of elemental synthesis units. These units are obtained by transposing and time-scaling the database samples. The concatenation of these transformed samples is performed by spreading out the spectral shape and phase discontinuities of the boundaries along a set of transition frames that surround the joint frames. The expression of the singing is applied through a Voice Model built up on top of a Spectral Peak Processing (SPP) technique.

1. INTRODUCTION

The voice is considered to be the most flexible, rich and powerful of all instruments and it has always been an inexorable source of inspiration for musicians, sound designers and audio programmers. Music computer people have been interested in vocal sound manipulation and synthesis ever since the appearing of the very first techniques that allowed such sort of processing. Nowadays this interest has spread to most of the music communities reaching a point in which it is hard to find a popular musical production without any vocal alienation.

The synthesis of voice has been approached form many different directions and though it is a slack categorization, we can split synthesis models into two groups: spectral models, which are based on the perception of the sound, and physical models, which are based on the production of the sound.

Both spectral and physical models have their own benefits and drawbacks [1]. Physical models such as acoustic tube models use the control parameters humans use to control their own vocal system and can generate time-varying behaviors by the model itself. On the other hand some parameters might not have intuitive significance and might interact in non-obvious ways. Spectral models such as those based on phase vocoders or sinusoidal tracks have precise analysis / synthesis methods and work close to the human perception but their parameterization is not that suitable for study, manipulation, or composition since they don't match any auditory system structure.

Models such as the one used in formant synthesizers are considered to be pseudo-physical models because even though these are mainly spectral models they make use of the source / filter decomposition. The singing synthesizer we present would be part of this model group.

2. SYSTEM DESCRIPTION

The system is composed of two modules: the expression module, which is in charge of giving expressiveness and naturalness to the input melody, and the synthesis module, which is in charge of the synthesis process itself.

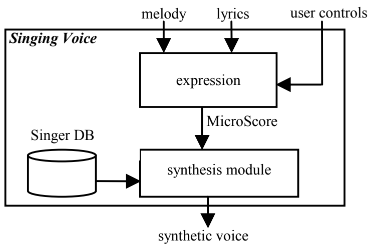


Figure 1: *General system diagram*

The indispensable inputs of the expression module are the lyrics and the musical score of the voice melody, that is to say, the phonetics and the fundamental frequency, dynamics and quantized duration values of each note. Out of these, the synthesizer can generate a virtual performance with default expression parameters. However, user control parameters can also be inputted to modify the expression through specifications on different types of note attacks, note transitions, vibratos, etcetera. The output of the expression module is a detailed musical score that characterizes the expressivity of the virtual singer performance through an ordered list of temporal events, each of which describes the local expression of the performance through the control parameters. This score is called Microscore.

The inputs of the synthesis module are the Microscore and the singer database. The synthesis module reads the Microscore information and synthesizes the virtual performance by taking the corresponding samples from the database, transforming them according to the inputted score and concatenating them [2].

3. VOICE AND SPECTRAL MODELING

In this section we will introduce both EpR Voice Model and Spectral Peak Processing (SPP) technique on which the system is based.

3.1. Voice Model (EpR)

Our Voice Model is based on an extension of the well known source/filter approach [3] we call EpR (Excitation plus Resonances). The EpR filter can be decomposed in two cascade filters. The first of them models the differentiated glottal pulse frequency response and the second the vocal tract (resonance filter).

The EpR Source filter

The EpR source is modeled as a frequency domain curve and one source resonance. The curve is defined by a gain and an exponential decay as follows:

$$Source_{dB} = Gain_{dB} + SlopeDepth_{dB}\left(e^{Slope\cdot f} - 1\right)$$

It is obtained from an approximation to the harmonic spectral shape (*HSS*) determined by the harmonics identified in the SMS analysis

$$HSS(f) = envelope_{i=0..n-1}\left[f_i, 20\log(a_i)\right]$$

where i is the index of the harmonic, n is the number of harmonics, f_i and a_i are the frequency and amplitude of the i^{th} harmonic.

On top of the curve, we add a resonance in order to model the low frequency content of the spectrum below the first formant. This source resonance is modeled as a symmetric second order filter (based on the Klatt formant synthesizer [4]) with center frequency *F*, bandwidth *Bw* and linear amplitude *Amp*, which is relative to the source curve.

The EpR vocal tract filter

The vocal tract is modeled by a vector of resonances plus a differential spectral shape envelope. It can be understood as an approximation to the vocal tract filter. These filter resonances are modeled just like the source resonance, where the lower frequency resonances are somewhat equivalent to the vocal tract formants.

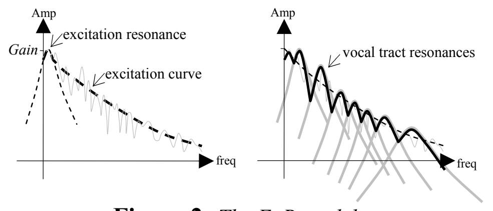


Figure 2: *The EpR model*

The differential spectral shape envelope actually stores the differences (in dB) between the ideal EpR model and the real harmonic spectral shape (*HSS*) of a singer's performance. We calculate it as a 30 Hz equidistant step envelope.

$$DSS(f) = envelope_{i=0..}\left[30i, HSS_{dB}(30i) - iEpR_{dB}(30i)\right] \qquad (1)$$

Thus, the original singer's spectrum can be obtained if no transformations are applied to the EpR model.

3.2. Spectral model (SPP)

Spectral Peak Processing (SPP) is used as the sample transformation tool. It performs a frame based spectral analysis of audio, giving as output the STFT, the harmonic peaks and the pitch. This technique is based on the phase-locked vocoder [5], but spectral peaks are calculated by parabolic interpolation [6] and a pitch analysis is performed. SPP considers the spectrum as a set of regions, each of which belongs to one harmonic peak and its surroundings. The goal of such technique is to preserve the local convolution of the analysis window after transposition and equalization transformations. A basic diagram can be seen in Figure 3.

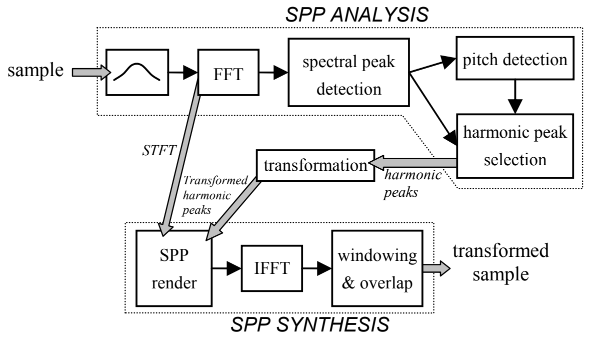


Figure 3: *SPP sample transformation*

4. SINGER DATABASE

The singer databases are created from dry and noiseless recordings of real singers. These databases hold both phonetic and expression information about the singer.

The phonetic database represents and stores the time varying phonetic characteristics of the singer so it stores the singer timbres in most representative phonetic contexts. It contains steady-states and articulations at different pitches.

The expression database characterizes the low level expression of the singer so it tries to retain the singer expressivity in different musical contexts. This database includes vibratos and note templates, which model attacks, releases and note transitions at different pitches. They are considered phoneme independent and organized into a musical meaningful classification [7].

5. TRANSFORMING SAMPLES

Basically, three kinds of transformation are applied to the samples: transposition, equalization and time-scaling. Time-scaling issues are described in detail in [8].

5.1. Transposition

Transposition means to multiply the harmonic's frequencies by a constant value. In terms of SPP, this operation can be done by shifting SPP regions in frequency. The linear frequency displacement for all the bins of each region will be the same. In most cases, this frequency displacement will be a non integer value. Thus we interpolate the spectrum with a 3rd order spline. This might not be the best method but it is a good compromise between quality and computational cost.

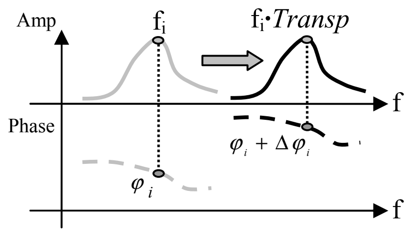


Figure 4: *Phase shift in SPP transposition*

In Figure 4, we can see an example of transposition to a higher pitch. In this case, the SPP regions will be separated in the resulting spectrum by gaps. Whenever transposing to a lower pitch, the SPP regions overlap in the resulting spectrum. This overlapping is computed by adding the complex values at each spectral bin.

Phase correction

When a SPP region is shifted in frequency in a frame by frame process, the phase needs to be corrected in order to continue the harmonics. For the ith harmonic, assuming linear frequency, the ideal phase increment between two consecutive frames is

$$\Delta\varphi_i = 2\pi f_i \Delta t$$

where the time increment between frames is Δt . If we transpose a sample by a factor *transp*, then the ideal phase increment should be

$$\Delta\varphi_i = 2\pi f_i \cdot transp \cdot \Delta t$$

Therefore, the amount of phase that should be added to the spectrum phase in order to continue the ith harmonic is

$$\Delta\varphi_i = 2\pi f_i \cdot transp \cdot \Delta t - 2\pi f_i \Delta t = 2\pi f_i \left(transp - 1\right)\Delta t$$

This phase increment is added to all the bins that belong to the ith region (see Figure 4). This way we can preserve the phase consistency across bins, i.e. the vertical phase coherence [5].

The phase increment applied to each harmonic is added to an accumulated phase frame by frame. This accumulated phase is the phase we finally add to each SPP region. However, this implementation results in the loss of the harmonic phase alignment after several frames because the frequencies do not follow a perfect harmonic scale. This loss of phase alignment produces some phasiness and loss of presence in the synthesized voice, especially for low pitches. To solve such problem, we consider the voice as perfectly harmonic. In that case, the equation of the phase increment will be

$$\Delta\varphi_i = 2\pi \cdot pitch \cdot \left(i+1\right)\left(transp - 1\right)\Delta t \tag{2}$$

where *i* is the index of the harmonic (*0* for the fundamental) and *pitch* is the estimated fundamental frequency (different from f_0).

5.2. Equalization

Equalization means timbre change. On one hand the SPP analysis outputs a spectrum with the harmonic peaks and the SPP regions. On the other hand we have an envelope that defines the desired timbre. The SPP equalization is done by calculating for each region the amplitude amount needed to match the timbre envelope, and adding it to all the bins that belong to the region. Therefore, the spectrum amplitude of a region will be just shifted up or down and the phase will not be changed.

6. CONCATENATING SAMPLES

When connecting transformed samples there will appear spectral shape and phase discontinuities. In order to minimize them, phase and amplitude corrections can be spread out along a set of transition frames that surround the joint frame.

6.1. Phase concatenation

In order to avoid phase discontinuities at the segment boundaries, we have come out with a phase continuity condition that takes

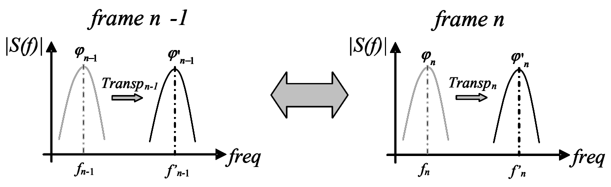


Figure 5: *Concatenation boundary*

care of the boundary phases and the possibly different transposition factors applied to each segment. In Figure 5 we can see the joint frames, where *frame n-1* is the last frame of the left segment and *frame n* is the first frame of the right segment. f_{n-1} and f_n refer to the frequencies of the *i*th harmonic.

The basic condition for phase continuity comes from the assumption that the frequency of each harmonic varies linearly between these two consecutive frames. In that case the phase relation between both frames should be

$$\varphi_n^i = \varphi_{n-1}^i + 2\pi \frac{f_{n-1}^i + f_n^i}{2}\Delta t$$

where φ_n^i and φ_{n-1}^i are the phases of the *i*th harmonic at the right and left frame respectively. Thus, the desired phase for the left frame ϕ_{n-1}^i should be

$$\phi_{n-1}^i = \varphi_n^i - 2\pi \frac{f_{n-1}^i + f_{n-1}^i}{2}\Delta t$$

But in fact we are not just concatenating two segments, but also transposing them with different transposition factors ($transp_{n-1}$ and $transp_n$). We should distinct then between the original frequency of each segment (f_{n-1} and f_n) and the transposed ones (f'_{n-1} and f'_n), where $f'_{n-1}=f_{n-1}\cdot transp_{n-1}$ and $f'_n=f_n\cdot transp_n$. The basic condition should be applied to the transposed frequencies and phases. This can be expressed as

$$\begin{aligned}\phi_{n-1}^i = \varphi_n^i - 2\pi \cdot \frac{f_{n-1}^i \cdot transp_{n-1} + f_n^i \cdot transp_n}{2} \cdot \Delta t \\ + 2\pi\left(i+1\right)\Delta t \cdot pitch_n \cdot \left(transp_n - 1\right)\end{aligned}$$

by using the transposition phase correction equation (2) and taking into account that the phase correction due to transposition is accumulated frame by frame. This correction is applied either to the left or to the right segment around the boundary, and spread along several frames in order to get a smooth transition. We can rewrite the previous equation as

$$\phi_{n-1}^i = \varphi_n^i - \Delta\varphi_c \tag{3}$$

where $\Delta\varphi_c$ is the phase correction that guarantees the phase continuation of the *i*th harmonic. In Figure 6 we can see an example where this phase correction is spread along 5 frames on the left part of the boundary.

Since the impulsive periods of left ant right segments are not aligned, we will often have big phase correction values. Therefore it's better if we calculate how much we should move the right segment (Δt_{sync}) in order to align the beginning of both periods, so that the phase correction to be applied is minimized. We could

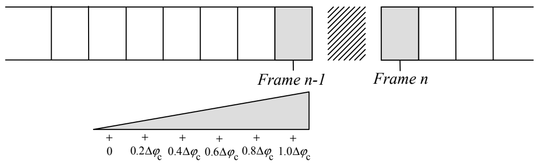


Figure 6: *Phase spreading in concatenation*

approximate this time shifting by assuming that the beginning of the periods will be aligned if the phase of the fundamental is continued. This can be expressed as

$$\Delta t_{sync} = \frac{-\Delta \varphi_c}{2\pi\, pitch_n \cdot transp_n} \qquad (4)$$

where $\Delta\varphi_c$ is calculated as in equation (3) for the particular case of *i*=0 (the fundamental). Finally, if we combine equations (3) and (4) we obtain

$$\begin{aligned}\phi_{n-1}^{i} &= \varphi_n^i + 2\pi(i+1)\, pitch_n \cdot transp_n \cdot \Delta t_{sync} \\ &-2\pi \cdot \frac{f_{n-1}^{i} \cdot transp_{n-1} + f_n^i \cdot transp_n}{2} \cdot \Delta t \\ &+2\pi(i+1)\Delta t \cdot pitch_n \cdot (transp_n - 1)\end{aligned}$$

6.2. Spectral shape concatenation

In order to avoid spectral shape discontinuities at the segment boundaries we make use of the EpR model. First we estimate the EpR of the boundary frames. The left EpR then is stretched using the resonance frequencies as mapping points ($F0_{left}$ *to* $F0_{right}$, $F1_{left}$ *to* $F1_{right}$,…) and it is subtracted from the right EpR. The differential envelope obtained accounts for the spectral shape differences between the two joint frames.

For each one of the transition frames at the left of the boundary, a frequency mapping function is obtained from the interpolation between the above resonance frequency mapping and the identity mapping (*y*=*x*) with a factor *1-SSIntp* which stands for the distance from the frame to the boundary (*SSIntp* is 0 at the beginning of the interpolation zone and 1 at the boundary). This mapping is applied to each transition frame spectral shape and finally the differential envelope (weighted by *SSIntp*) is added to it (see Figure 7).

Notice that the spectral shape interpolation is spread along several frames in a similar way to the phase concatenation, but with the addition of the spectrum stretching.

7. CONCLUSIONS

The singing synthesizer we have presented in this paper has proven to comprise suitable methods and techniques for the generation of synthetic vocal tracks.

The SPP has brought a significant improvement in comparison to our previous analysis / synthesis technique [7] based on sinusoidal plus residual decomposition [6]. SPP avoids the tricky sinusoids / noise decomposition and preserves accurately and consistently the voice quality of the original singer. The preservation of the voice quality is a very valued feature when it comes to create databases of singers with vocal disorders such as hoarseness or breathiness.

Even tough the system is in a stage in which a synthetic voice is distinguishable from a real voice the quality is good enough to use it for backing vocals with no synthetic voice perception at all. Anyway we believe we will not have to wait that long to hear synthesized lead vocals that sound indistinguishable from a human singer.

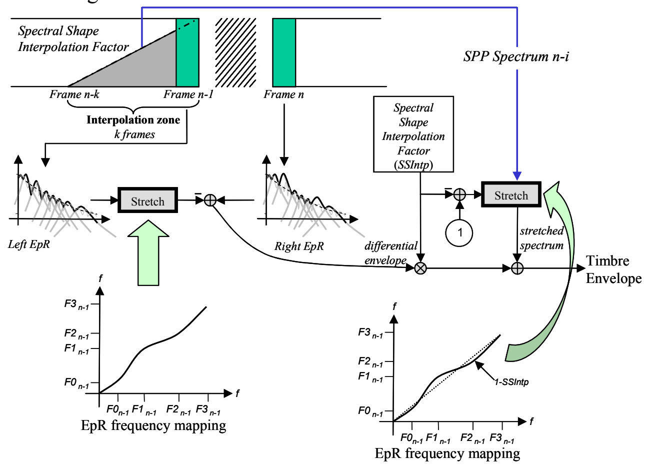


Figure 7: *The sample transformation*

8. REFERENCES

[1] Cook, R., *Toward the Perfect Audio Morph? Singing Voice Synthesis and Processing*, Proceedings of the Digital Audio Effects Workshop DAFX, Barcelona 1998.

[2] Bonada, J., Loscos, A., Cano, P., and Serra, X, *Spectral Approach to the Modeling of the Singing Voice*, Proceedings of the 111th AES Convention, New York, USA 2001.

[3] Childers, D. G., *Measuring and Modeling Vocal Source-Tract Interaction*, IEEE Transactions on Biomedical Engineering, 1994.

[4] Klatt, D. H., *Software for a cascade/parallel formant synthesizer*, Journal of the Acoustical Society of America, pp. 971-995, 1980.

[5] Laroche, J. and Dolson, M., *Improved phase-vocoder. Time-Scale Modification of Audio*, IEEE Transactions on Speech and Audio Processing, vol. 7, no. 3, pp. 323-332, May1999.

[6] Serra, X, *A system for sound analysis/transformation/synthesis based on a deterministic plus stochastic decomposition.*, PhD thesis, CCRMA, Dept. of Music, Standford University, 1989.

[7] Bonada, J., Celma, O., Loscos, A., Ortolà, J., and Serra, X., *Singing Voice Synthesis Combining Excitation plus Resonance and Sinusoidal plus Residual Models*, Proceedings of the International Computer Music Conference, Havana, Cuba 2001.

[8] Bonada, J., *Automatic Technique in Frequency Domain for Near-Lossless Time-Scale Modification of Audio*, Proceedings of the International Computer Music Conference, Berlin, Germay 2000.

Appendix B

Patents

[Pat1] JP2001117580 Device and method for sound signal processing. Inventors: Nakagawa, T., Cano, P., Loscos, A. Applicants: Yamaha Corp. Publication date: 2001-04-27.

[Pat2] JP2001117582 Voice Processor and karaoke device. Inventors: Kawashima, T., Cano, P., Loscos, A. Applicants: Yamaha Corp., Univ. Pompeu Fabra. Publication date: 2001-04-27.

[Pat3] JP2001117580 Device and method for sound signal processing. Inventors: Nakagawa, T., Cano, P., Loscos, A. Applicants: Yamaha Corp., Univ. Pompeu Fabra. Publication date: 2001-04-27.

[Pat4] JP2001117578 Device and method for adding harmony sound. Inventors: Nakagawa, T., Cano, P., Loscos, A., Bonada, J. Applicants: Yamaha Corp., Univ. Pompeu Fabra. Publication date: 2001-04-27.

[Pat5] JP2001117568 Singing Evaluation device and karaoke device. Inventors: Kageyama, Y., Cano, P., Loscos, A. Applicants: Yamaha Corp., Univ. Pompeu Fabra. Publication date: 2001-04-27.

[Pat6] EP1291846 Voice synthesizing apparatus capable of adding vibrato effect to synthesized voice. Inventors: Loscos, A., Yoshioka, Y., Applicants: Yamaha Corp. Publication date: 2003-03-12.

[Pat7] JP2003076387 Method and device for voice synthesis and program. Inventors: Yoshioka, Y., Loscos, A. Applicants: Yamaha Corp. Publication date: 2003-03-14.

References

[ABLAV03] Amatriain, X., Bonada, J., Loscos, A., Arcos, J., Verfaille, V., *Content-based Transformations*, Journal of New Music Research Vol.32 .1 2003.

[ABLS01] Amatriain, X., Bonada, J., Loscos, A., Serra, X., *Spectral Modeling for Higher-level Sound Transformation,* Proceedings of MOSART Workshop on Current Research Directions in Computer Music, Barcelona 2001.

[ABLS02] Amatriain, X., Bonada, J., Loscos, A., Serra, X., *Spectral Processing*, Udo Zölzer Ed., *DAFX: Digital Audio Effects*, p.554 John Wiley & Sons Publishers, 2002.

[BBCLS00] de Boer, M., Bonada, J., Cano, P., Loscos, A., Serra, X., *Singing Voice Impersonator Application for PC*', Proceedings of International Computer Music Conference, Berlin 2000.

[BCLOS01] Bonada, J., Celma, O., Loscos, A., Ortolà, J., Serra, X., *Singing Voice Synthesis Combining Excitation plus Resonance and Sinusoidal plus Residual Models*, Proceedings of International Computer Music Conference, Havana 2001.

[BL03] Bonada, J., Loscos, A., *Sample-based singing voice synthesizer by spectral concatenation*, Proceedings of Stockholm Music Acoustics Conference, Stockholm 2003.

[BLCS01] Bonada, J., Loscos, A., Cano, P., Serra, X., *Spectral Approach to the Modeling of the Singing Voice*, Proceedings of 111th AES Convention, New York 2001.

[BLMK03] Bonada, J., Loscos, A., Mayor, O., Kenmochi, H., *Sample-based singing voice synthesizer using spectral models and source-filter decomposition*, Proceedings of 3rd International Workshop on Models and Analysis of Vocal Emissions for Biomedical Applications, Firenze, 2003.

[Bon00] Bonada, J., *Automatic Technique in Frequency Domain for Near-Lossless Time-Scale Modification of Audio*, Proceedings of International Computer Music Conference, Berlin, 2000.

[Bon97] Bonada, J., *Desenvolupament d`un entorn gráfic per a l`análisi, transformació i síntesi de sons mitjanant models espectrals,* UPC, Barcelona, 1997.

[Cano98] Cano, P., *Fundamental Frequency Estimation in the SMS analysis*, Proceedings of the COST G6 Conference on Digital Audio Effects, Barcelona 1998.

[Chil94] Childers, D. G., *Measuring and Modeling Vocal Source-Tract Interaction*, IEEE Transactions on Biomedical Engineering, 1994.

[CLB99] Cano, P., Loscos, A., Bonada, J., *Score-Performance Matching using HMMs,* Proceedings of International Computer Music Conference, Beijing 1999.

[CLBBS00] Cano, P., Loscos, A., Bonada, J., de Boer, M., Serra, X., *Voice Morphing System for Impersonating in Karaoke Applications*, Proceedings of International Computer Music Conference, Berlin 2000.

[Cook91] P. Cook, *Identification of control parameters in an articulatory vocal tract model with applications to the synthesis of singing*, Stanford University, 1991.

[Cook92] P. Cook, *SPASM: a Real-Time Vocal Tract Physical Model Editor/Controller and Singer: the Companion Software Synthesis System*, Computer Music Journal, 17: 1, pp 30-44, 1992.

[Cook96] P. Cook, *Singing Voice Synthesis History, Current Work, and Future Directions*, Computer Music Journal, 20:2 1996.

[Cook98] P. Cook, *Toward the Perfect Audio Morph? Singing Voice Synthesis and Processing*, Proceedings of the Digital Audio Effects Workshop DAFX, Barcelona 1998.

[DGR93] Depalle, Ph., G. Garcia and X. Rodet. 1993. "Analysis of Sound for Additive Synthesis: Tracking of Partials Using Hidden Markov Models." *Proceedings of the 1993 International Computer Music Conference*, San Francisco: Computer Music Association.

[DQ97] Y. Ding and X. Qian, *Processing of musical tones using a combined quadratic polynomial-phase sinusoid and residual (quasar) signal model*, Journal of the Audio Engineering Society, July/August, 1997.

[FG66] Flanagan, J. L. and Golden, R. M., *Phase Vocoder*, Bell System Technical Journal, pp. 1493-1509, 1966.

[Fla72] Flanagan J., *Speech Analysis, Synthesis, and Perception*, Springer-Verlag, Berlin-Heidelberg-New York, 1972.

[Gar92] Garcia G., *Analyse des Signaux Sonores en Termes de Partiels et de Bruit. Extraction Automatique des Trajets Frèquentiels par des Modèles de Markov Cachès*, Mèmoire de DEA en Automatique et Traitement du Signal, Orsay, 1992.

[Good97] M. M. Goodwin, *Adaptive Signal Models: Theory, Algorithms, and Audio Applications*, Ph.D. Thesis, University of California at Berkeley, 1997.

[GR94] Goodwin, M. and X. Rodet, *Efficient Fourier Synthesis of Nonstationary Sinusoids*, Proceedings of the 1994 International Computer Music Conference, San Francisco: Computer Music Association, 1994.

[GS92] E. B. George , M. J. Smith, *An analysis by synthesis approach to sinusoidal modeling applied to the analysis and synthesis of musical tones*, Journal of the Audio Engineering Society, vol. 40, pp. 497-516, June 1992.

[Har78] F. J. Harris, *On the use of windows for harmonic analysis with the discrete fourier transform*, Proceedings of the IEEE, 1978.

[Kla80] Klatt D, *Software for a Cascade/Parallel Formant Synthesizer*, Journal of the Acoustical Society of America, JASA, Vol. 67: 971-995, 1980.

[LB88] Lieberman, P. and Blumestein, S., *Speech Physiology, speech perception and acoustics phonetics*, Cambridge University Press, 1988.

[LCB99] Loscos, A., Cano, P., Bonada, J., *Low-Delay Singing Voice Alignment to Text*, Proceedings of International Computer Music Conference, Beijing 1999.

[LD99] Jean Laroche and Mark Dolson, *New phase-vocoder techniques for pitch-shifting, harmonizing and other exotic effects*, Proceedings of IEEE Workshop on Applications of Signal Processing to Audio and Acoustics, 1999.

[LR98] Loscos, A., Resina, E., *SMSPerformer: A real-time synthesis interface for SMS*', Proceedings of COST G6 Conference on Digital Audio Effects, Barcelona 1998.

[MB94] R. C. Maher and J. W. Beauchamp, *Fundamental Frequency Estimation of Musical Signals using a two-way Mismatch Procedure*, Journal of the Acoustical Society of America, (4):2254—2263, 1994.

[MJOCG97] M. W. Macon, L. Jensen-Link, J. Oliverio, M. Clements, and E. B. George, *Concatenation-based MIDI-to-Singing Voice Synthesis*, Proceedings 103 rd AES , 1997.

[MQ86] R.J. McAulay and T. F. Quatieri, *Speech analysis/synthesis based on a sinusoidal representation*, Proceedings of the IEEE Transactions on Acoustics, Speech, and Signal Processing, 1986.

[Puck01] M. Puckette, *Synthesizing Sounds with Specified, Time-Varying Spectra*, Proceedings of International Computer Music Conferences, Havana 2001.

[Puck95] Puckette, M. S., *Phase-locked vocoder*, Proceedings of IEEE Conference on Applications of Signal Processing to Audio and Acoustics, Mohonk 1995.

[RD92] X. Rodet, Ph. Depalle, *A new additive synthesis method using inverse Fourier transform and spectral envelopes*, Proceedings of ICMC, San Jose, California, Oct. 1992.

[Rod02] X. Rodet, *Synthesis and Processing of the Singing Voice*, Proc.1st IEEE Benelux Workshop on Model based Processing and Coding of Audio (MPCA-2002), Leuven, Belgium, November 15, 2002.

[Rod85] Xavier Rodet, Yves Potard, and Jean-Baptiste Barrière, *CHANT - de la synthèse de la voix chantée à la synthèse en général,* Rapport Ircam 35/85, 1985.

[Rod84] Xavier Rodet, *Time-domain formant-wave-function synthesis*, Computer Music Journal, 8(3):9--14, 1984.

[Rod97] X. Rodet, *Musical Sound Signal Analysis/Synthesis Sinusoidal+Residual and Elementary Waveform Models*, Proceedings of the IEEE Time-Frequency and Time Scale Workshop 97, Coventry, Grande Bretagne, 1997.

[Ser97] X. Serra, *Sound Modelling with Sinusoids plus Noise*, Swets & Zeitlinger Publishers, 1997.

[SS90] X. Serra and J.O. Smith, *A system for sound analysis/transformation/synthesis based on a deterministic plus stochastic decomposition*, Computer Music Journal, 1990.

[Sun87] J. Sundberg, *The Science of Singing Voice*, Illinois Universitary Press, 1987.

[Vas96] S.V. Vaseghi, *Advanced Signal Processing and Digital Noise Reduction*, Wiley Teubner, 1996.

[VM98] T.S. Verma and T.H.Y. Meg, *Time scale modifications using a sines + transients + noise signal model*, Proceeding of the DAX98 Digital Audio Effects Workshop, November 1998.

[h1] University of Washington Department of Otolaryngology website, *http://depts.washington.edu/otoweb/patients/pts_specialties/pts_voice-prob/pts_voice-prob.htm*

[h2] Perry Cook website, *http://www.cs.princeton.edu/~prc/SingingSynth.html*

[h3] SMS web page. *http://www.iua.upf.es/sms*