SMS-FESTIVAL: a New TTS Framework

Giacomo Sommavilla, Piero Cosi, Carlo Drioli, Giulio Paci

Istituto di Scienze e Tecnologie della Cognizione - Sede di Padova “Fonetica e Dialettologia”,
Consiglio Nazionale delle Ricerche, Padova, Italy
`{sommavilla, cosi, drioli, paci}@pd.istc.cnr.it`

Abstract

A new sinusoidal model based engine for FESTIVAL TTS system which performs the DSP (Digital Signal Processing) operations (i.e. converting a phonetic input into audio signal) of a diphone-based TTS concatenative system, taking as input the NLP (Natural Language Processing) data (a sequence of phonemes with length and intonation values elaborated from the text script) computed by FESTIVAL is described.

The engine aims to be an alternative to MBROLA and makes use of SMS (“Spectral Modeling Synthesis”) representation, implemented with the CLAM (C++ Library for Audio and Music) framework.

This program will be released with open source license (GPL), and will compile everywhere gcc and CLAM do (i.e.: Windows, Linux and Mac OS X operating systems).
Index Terms: TTS, SMS, MBROLA, FESTIVAL, GPL.

1. Introduction

The whole DSP speech synthesis process is based upon the SMS model and consists in three logical steps: analysis of concatenative unit database, diphone transformations plus concatenation and synthesis.

In this section we provide a brief history of analysis/synthesis models for speech synthesis and a short introduction of the sinusoidal plus residual model. In section 2 we will describe the SMS analysis and synthesis steps. In section 3 we will focus on our custom diphone transformation and concatenation algorithms.

1.1. Preamble: a brief history

Analysis/synthesis models for speech signal processing appeared in mid-thirties when the VODER was created by Homer Dudley, inspired by VOCODER; later, in sixties, Flanagan invented his Phase Vocoder (PV). In the mid eighties Julius Smith developed the program PARSHL for the purpose of supporting inharmonic and pitch changing sounds. This approach is better suited for analysis of inharmonic and pseudo-harmonic sounds. At the same time, independently, Quatieri and McAulay developed a similar technique for analyzing speech. In late nineties Yannis Stylianou worked on Harmonic plus Noise Model (HNM) for concatenative TTS synthesis systems. Harmonic and modulated noise components are separated in the frequency domain by a time-varying parameter, referred to as *maximum voiced frequency*, F_m.

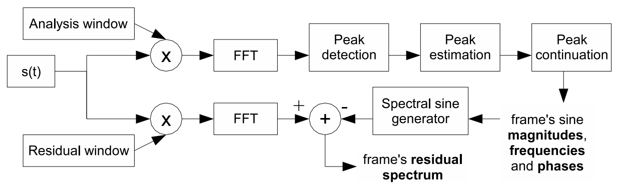

Figure 1: *SMS Analysis scheme flow-chart.*

1.2. Sinusoidal plus residual model

We briefly introduce the sinusoidal plus residual representation, which separates the input audio signal into a sum of partials plus an inharmonic, noise-like part, called *residual*:

$$s(t) = \sum_{r=1}^{R} A_r \cos[2\pi f_r t + \theta_r] + e(t) \qquad (1)$$

where $s(t)$ is the audio signal, $e(t)$ the residual component, A_r, f_r and θ_r are respectively the amplitude, frequency and phase of the r-th sinusoid.

2. SMS - Analysis and Synthesis

SMS [1] is a set of techniques for processing audio signals which implements the sinusoidal plus residual model. The task of SMS analysis is to extract some spectral parameters from the time-domain signal. From this kind of data we can obtain a time domain signal through the SMS synthesis step.

2.1. SMS Analysis

In Fig. 1 we can see the block diagram of the whole SMS analysis behavior. It is based upon the Short Time Fourier Transform (STFT): the signal is cutted into consecutive overlapping frames, which are multiplied by an analysis window and for each of these chunks a FFT is computed. We obtain a spectrum from which we detect the components present in the original sound. The harmonic analysis consists of peak detection, pitch detection and spectral peak continuation. When the harmonic analysis is completed, the residual component can be computed, and thus the whole SMS analysis step is achieved.

2.1.1. Peak detection

In audio processing time-varying sinusoids are called partials, and each of them is the result of a main mode of vibration of the generating system. A partial in the frequency domain can be identified by its spectral shape (magnitude and phase), its relation to other partials, and its time evolution. So the first step of SMS analysis is the detection of partials, which are searched among prominent magnitude peaks of the current frame spectrum.

Most natural sounds are not perfectly periodic and do not have nicely spaced and clearly defined peaks in the frequency domain. A practical solution is to detect as many peaks as possible and delay the decision of what is a deterministic, or “well behaved” partial, to the next step in the analysis: the peak continuation algorithm.

2.1.2. Pitch detection

Before continuing a set of peak trajectories through the current frame it is useful to search for a possible fundamental frequency. If it exists, we will have more information to work with, and it will simplify and improve the tracking of partials.

The fundamental frequency can be defined as the common divisor of the harmonic series that best explains the spectral peaks. It is possible that the common divisor does not belong to the set of detected peaks. For this reason the fundamental frequency is better called pitch (i.e. that particular frequency heard to be the main frequency of a sound). The algorithm that choose the fundamental frequency can be simply described in three main steps: 1. Choose possible fundamental candidates. 2. Measure the “goodness” of the resulting harmonic series compared with the spectral peaks. 3. Get the best candidate.

2.1.3. Peak continuation

From peak detection we obtain some “wrong behaved” partials that shouldn’t be considered. The basic idea of the algorithm is that a set of “guides” advances in time through the spectral peaks, looking for the appropriate ones (according to the specified constraints) and forming trajectories out of them. The instantaneous state of the guides, their frequency and magnitude, are continuously updated as the guides are turned on, advanced, and finally turned off. When a fundamental has been found in the current frame, the guides can use this information to update their values. Peak continuation algorithm completes harmonic analysis.

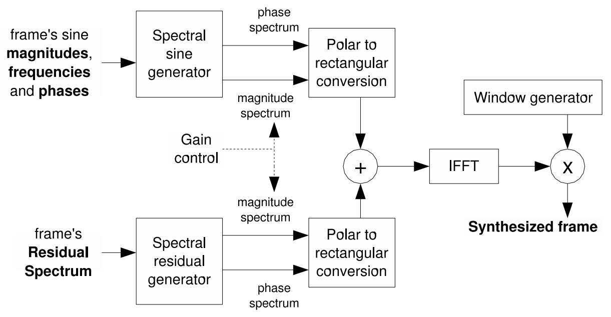
frame's sine **magnitudes**, **frequencies** and **phases**
Spectral sine generator
phase spectrum
magnitude spectrum
Polar to rectangular conversion
Gain control
Window generator
+
IFFT
x
Synthesized frame
frame's **Residual Spectrum**
Spectral residual generator
magnitude spectrum
phase spectrum
Polar to rectangular conversion

Figure 2: *SMS Synthesis scheme flow-chart.*

2.1.4. Stochastic Analysis

Referring to formula (1), the stochastic component $e(t)$ of the current frame is calculated by first re-generating the deterministic signal with additive synthesis, and then subtracting it from the original waveform $s(t)$ in the time domain. This is possible because the phases of the original sound are matched, so the shape of the original waveform is preserved. The stochastic representation is then obtained by performing a spectral fitting of the residual signal.

2.2. SMS Synthesis

The SMS synthesis process is described in Fig. 2 where we can see the two inputs that come from the analysis (possibly transformed) representing the deterministic (harmonic) component and the residual one as described in section 2.1.

The whole signal is processed in the frequency domain, where the two components are treated independently, then we return to time domain performing an inverse FFT. For the deterministic component the goal is to obtain the spectrum of a sum of sinusoids. The stochastic signal is obtained by filtering white noise with residual spectral envelope. Then we can use a single iFFT for the combined spectrum. Finally in the time domain we impose the triangular window in the overlap-add process, combining successive frames to get the time-varying characteristics of the sound.

3. Speech Synthesis Architecture

The SMS engine performs the DSP operations of a text-to-speech system, taking as input a phonetic file computed by FESTIVAL [2], which describes the pronunciation of the text script through a sequence of phonemes with length in ms and intonation values in Hz.

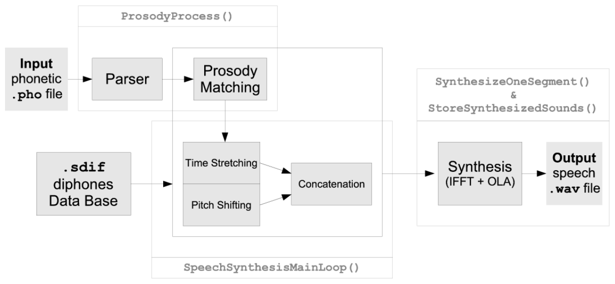

Figure 3: *Scheme of speech synthesis architecture operations.*

Our program aims to be an MBROLA [3] alternative and thus it makes use of the same phonetic input format and command line parameters. Both programs implement concatenative synthesis using diphones (that must be externally supplied) as base units.

The three logical steps upon which the whole DSP speech synthesis process is based are:

1. **analysis**: it has been theoretically described in section 2.1 and consists in converting the time domain diphones' database into a spectral parameters one stored in Sound Description Interface Format (SDIF);

2. **transformations** plus **concatenation**: it will be described further in this section;

3. **synthesis**: it has been described in section 2.2.

Fig. 3 shows a simplified block diagram of the transformations plus concatenation and the synthesis steps.

3.1. SMS Transformations

The task of the transformation step is to adapt every required diphone to the parameters specified (for each phoneme) by the phonetic input. By now those parameters describe only the duration of the phoneme and its pitch evolution.

So the main spectral transformations needed are time stretching and pitch shifting. Both these operations modify magnitude and frequency of signal partials. These new values become incoherent with original phases. This problem affects the re-synthesized signal, deteriorating its audio quality.

Thus it is necessary to re-calculate the sinusoids' phases; two ways for doing this are available: the phase continuation and the relative phase delay algorithms.

3.1.1. Time Stretching

The time stretching process is based upon linear interpolation and decimation of frequencies and magnitudes. The algorithm preserves transition integrity and intelligibility between different phonemes in every diphone, as much as the input time requirements are satisfied.

3.1.2. Pitch Shifting

Pitch Shifting is the transformation that takes care of modulating the intonation of the sentence to be uttered. It is performed after time stretching to better fit the intonation requirements.

The Pitch Shifting routine is implemented using a *formant preserving* algorithm, that tries to maintain the original timbre of sound. The magnitude of each transformed partial is placed upon the original spectrum envelope, corresponding to the original frequency scaled by a common factor.

3.1.3. Phase Continuation

The simplest way to reconstruct the phases is called phase continuation. Its behavior is to arbitrarily set the phase for every partial of the first frame and then compute the values frame-by-frame. In this way the algorithm discards *all* original analyzed phase data. The formula used to propagate phase values is the following:

$$\theta_i^k = \theta_{i-1}^k + \pi H(f_{i-1}^k + f_i^k) \qquad (2)$$

being f_i^k and θ_i^k respectively the frequency and phase of the k-th partial of the i-th frame, and H the hop size.

3.1.4. Relative phase delay representation

A more complex method, theorized in [4], that helps to preserve the original waveform is based on the *relative phase delay* representation of the phase, defined as the difference between the phase delay (phase/radian frequency ratio) of the partials and the phase delay of the fundamental. This makes the waveform characterization independent from the phase of the first partial.

Once the relative phase delays are computed for each frame it is therefore possible to propagate the phase of the modified fundamental, as described in equation (2), and rebuild the waveform by adding the relative phase delays to the new fundamental phase delay.

3.1.5. SMS Diphones Concatenation

Once transformed, two successive diphones have to be concatenated. This operation is mainly based on the time stretching interpolation and decimation subroutine, although also pitch shifting is used. Basically the behavior of this operation is to morph the last frames of a diphone with the first frames of the following one. The pitch of those frames is matched and then magnitude and frequency

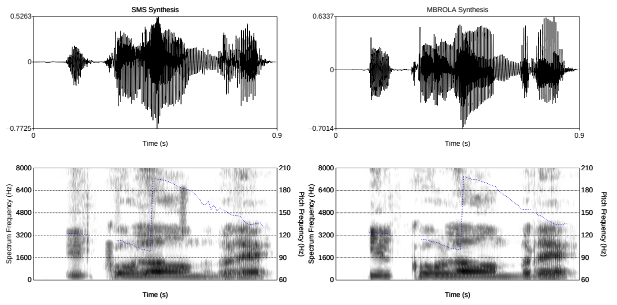

Figure 4: *SMS-MBROLA Synthesis comparison.*

are interpolated. This assures a smoothing between frames of the phoneme belonging to consecutive diphones.

3.2. The CLAM framework

The CLAM (C++ Library for Audio and Music) framework [5] aims to offer extensible, generic and efficient design and implementation solutions for developing Audio and Music applications and it is perfectly suited for implementing the SMS model.

It simplified a lot our task since it is quite complete and includes all utilities needed in a Sound Processing Project (input/output processing, storage, display...). Moreover its good design allows easy adaptation to any kind of need.

The project is released under GPL version 2 (or later). It is Platform Independent (compiles under GNU/Linux, Windows and Mac platforms) and thus it is quite simple to create portable applications.

4. SMS-MBROLA comparison

Even if the system is still at a preliminary stage, some comparisons with the state of the art MBROLA concatenative speech synthesis are already available at “http://www.pd.istc.cnr.it/FESTIVAL/home/SMS.htm”.

We have synthesized some samples from the same phonetic files with our engine and MBROLA. We have also used diphone databases obtained from the same audio recordings (a collection of 1299 diphones of the Italian language.)

However it must be observed that the size of the SMS analyzed database is about 10 times more than the original time domain one ($\sim$70 MB against 6.5 MB). MBROLA synthesis engine is faster than ours, however performance was out of the scope of this work.

Either MBROLA and our engine synthesize well intelligible phrases and our concatenation routine works quite good even in those cases in which several phonemes are rapidly spoken. Anyway MBROLA synthesis audio quality is often cleaner than ours, which is sometimes affected by some hoarseness. In Fig. 4 the sentence “il colombre” has been synthesized by SMS (on the left) and MBROLA (on the right).

5. Conclusions

A new sinusoidal model based engine for concatenative TTS system has been presented. This engine has been proved to be comparable, in terms of audio quality and intelligibility, with similar, state of the art systems.

We are evaluating some strategies in order to improve the engine. The most important ones are:

- analysis verification process, in order to correct some artifacts that may occur in the analysis step;
- SMS pitch synchronous operations;
- alternative implementation of pitch shifting and time stretching, in order to obtain better audio quality;
- voice quality parameters support, such as Spectral Tilt, in order to perform emotional TTS synthesis.

6. Acknowledgements

Grateful acknowledgement goes to the CLAM team, in particular to Xavier Amatriain, David García Garzón and Pau Arumí for being always very helpful with our work.

7. References

[1] Serra, X. and Smith, J. O., “Spectral Modeling Synthesis: A Sound Analysis/Synthesis Based on a Deterministic plus Stochastic Decomposition”, Computer Music Journal, vol. 14(4), 1990.

[2] Black, A. W. and Taylor, P. A., “The Festival Speech Synthesis System: System Documentation”, Human Communication Research Centre, University of Edinburgh, http://www.cstr.ed.ac.uk/projects/festival.html, 1997.

[3] Dutoit, T. and Leich, H., “MBR-PSOLA Text-To-Speech Synthesis based on an MBE Re-Synthesis of the Segments Database”, Speech Communication, Elsevier Publisher, 1993.

[4] Di Federico, R., “Waveform preserving time stretching and pitch shifting for sinusoidal models of sound”, In Proceedings of the COST-G6 Digital Audio Effects Workshop, 1998.

[5] Amatriain, X., Arumí, P. and Ramírez, M., “CLAM, Yet Another Library for Audio and Music Processing?”, Proceedings of 17th Annual ACM Conference on Object-Oriented Programming, Systems, Languages and Applications, Seattle (USA), 2002.