SMS-FESTIVAL: un nuovo ambiente di lavoro per la sintesi vocale da testo scritto

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

1. SOMMARIO

Si presenta un nuovo motore di sintesi audio che si occupa di eseguire le operazioni di Digital Signal Processing (DSP) di un sistema di Text-To-Speech basato su concatenazione di difoni. L'input fonetico (una sequenza di fonemi con lunghezze e valori di intonazione prodotta da FESTIVAL a partire da testo scritto) viene convertito nel segnale audio.

Il lavoro descritto vuole essere un'alternativa a MBROLA e fa uso della rappresentazione SMS ("Spectral Modeling Synthesis"), implementata dal framework CLAM (C++ Library for Audio and Music).

Il programma verrà pubblicato con licenza open source (GPL), e funzionerà su qualsiasi piattaforma che disponga di gcc e CLAM (per esempio i sistemi Windows, Linux e Mac OS X).

2. INTRODUZIONE

L'intero processo di elaborazione del segnale vocale è basato sul modello SMS e consiste di tre operazioni fondamentali: analisi di un database di unità concatenative, trasformazione e concatenazione dei difoni e infine sintesi. Questo lavoro è la naturale continuazione di (Sommavilla *et alii*, 2005).

In questa sezione presentiamo un breve excursus storico di modelli per l'analisi e la sintesi del segnale vocale e una veloce introduzione del modello sinusoidi più residuo. Nella sezione 2 descriveremo i passaggi di analisi e sintesi SMS. Nella sezione 3 focalizzeremo l'attenzione sui nostri algoritmi di trasformazione e concatenazione dei difoni.

2.1 Un breve excursus storico

I rapporti tra i modelli di analisi/sintesi e l'elaborazione del segnale vocale cominciarono verso la metà degli anni Trenta quando il VODER fu creato da Homer Dudley, ispirato dal VOCODER (Voice Coder) precedentemente sviluppato presso i Bell Laboratories come speech coder per applicazioni di telecomunicazione.

Il Phase Vocoder (PV) di Flanagan è uno dei primi algoritmi di elaborazione del segnale digitale che possa essere classificato come tecnica di analisi/sintesi. Nel PV, il segnale è modellato come somma di onde sinusoidali, e i parametri che devono essere determinati dall'analisi sono le ampiezze e le frequenze tempo-varianti di ognuna di queste. Tali parametri possono essere usati per effettuare trasformazioni di suoni registrati.

Verso la metà degli anni Ottanta Julius Smith sviluppò il programma PARSHL (Smith & Serra, 1987) con lo scopo di supportare suoni inarmonici e con variazioni di pitch. PARSHL è una semplice applicazione per tracciare i picchi calcolati tramite FFT usata nell'ambito dell'elaborazione di segnali navali. Questo approccio è più adatto per l'analisi di suoni inarmonici e pseudo-armonici con sensibili variazioni nel tempo della frequenza. Circa nello stesso periodo, indipendentemente, Quatieri and McAulay svilupparono un

tecnica (McAulay & Quatieri, 1986) simile a PARSHL per analizzare la voce. Entrambi questi approcci lavorano bene per la maggior parte dei suoni creati da semplici vibrazioni di sistemi fisici, ma sono inadatti a rappresentare segnali rumorosi come l'attacco di vari suoni prodotti da strumenti musicali. Quindi il naturale passaggio successivo nel campo della modelizzazione spettrale fu quello di rappresentare le sinusoidi e il rumore come due componenti separate.

Alla fine degli anni Novanta Yannis Stylianou lavorò al Harmonic plus Noise Model (HNM) (Stylianou, 1998) per sistemi di sintesi vocale concatenativi. I segnali vocali vengono ancora rappresentati da una componente armonica tempo-variante più una componente di rumore modulato. La decomposizione in queste due parti permette trasformazioni del segnale più naturali. Le due componenti sono separate nel dominio frequenziali dal parametro tempo-variante F_m, detto maximum voiced frequency. L'algoritmo inizia con una stima dei parametri, dopodiché le fasi dei frames vocalizzati vengono corrette. Il processo di sintesi comprende il calcolo dei parametri armonici e rumorosi, la loro concatenazione e smoothing. La sintesi viene effettuata a pitch sincrono usando un procedimento di overlap and add.

2.2 Modello sinusoidi più residuo

Prima di descrivere la tecnica SMS introduciamo brevemente la rappresentazione sinusoidi più residuo, che separa il segnale audio in una somma di parziali e in una parte non armonica (stocastica, rumorosa), chiamata residuo:

$$s(t) = \sum_{i=1}^{R} A_i \cos(2\pi f_i t - \theta_i) + e(t) \tag{1}$$

dove $s(t)$ è il segnale audio, $e(t)$ la componente residuale, A_r, f_r e θ_r sono rispettivamente l'ampiezza, la frequenza e la fase della r-esima sinusoide.

3. SMS - ANALISI E SINTESI

SMS (Serra & Smith, 1990) è un insieme di tecniche per elaborare segnali audio che implementa un modello sinusoidi più residuo. Il compito dell'analisi SMS è di estrarre parametri spettrali dal segnale nel dominio temporale. Da questo tipo di dati possiamo ottenere un segnale nel tempo attraverso la procedura di sintesi.

3.1 Analisi SMS

In figura 1 possiamo vedere il diagramma a blocchi per l'analisi SMS, che si basa sulla Short Time Fourier Transform (STFT): il segnale viene suddiviso in frames consecutivi che si sovrappongono l'un l'altro. I frames vengono finestrati per un'opportuna finestra d'analisi e viene calcolata la FFT di ognuno. Otteniamo così uno spettro da cui estrarre le componenti presenti nel suono originale. L'analisi armonica è costituita dalle operazioni di peak detection, pitch detection e spectral peak continuation.

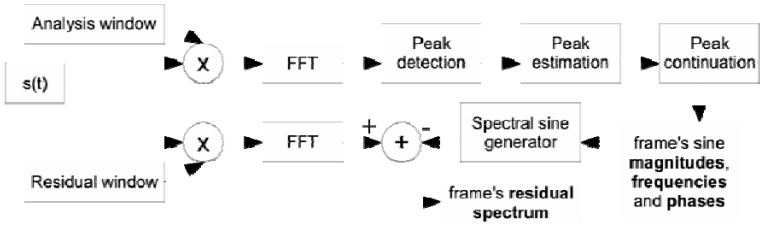

Figura 1: Diagramma di flusso per l'analisi SMS.

L'analisi a pitch sincrono potrebbe migliorare l'accuratezza di queste operazioni, ma non è stata ancora implementata nel nostro lavoro.

Una volta terminata l'analisi armonica la componente residuale può essere calcolata, completando così l'intera procedura di analisi.

3.1.1 Peak detection

Nell'elaborazione di segnali audio le sinusoidi tempo-varianti che compongono un suono vengono chiamate parziali, e ognuna di queste rappresenta un modo principale di vibrazione del sistema fisico che le genera. Una parziale nel dominio frequenziale può essere identificata tramite la sua forma spettrale (ampiezza e fase), la sua relazione con altre parziali e la sua evoluzione nel tempo. Anche conoscere la sorgente del suono e le sue caratteristiche può essere utile in questo processo. Il primo passo dell'analisi SMS è rilevare le parziali, che vengono ricercate tra i picchi di ampiezza dello spettro del frame in esame. L'accuratezza della rilevazione dei picchi viene migliorata dall'interpolazione spettrale tramite l'introduzione di zero-padding nel dominio del tempo.

Teoricamente, questo dovrebbe essere sufficiente per rilevare correttamente le sinusoidi, ma in pratica ciò accade raramente, dato che la maggior parte dei suoni non sono perfettamente periodici e non presentano picchi nettamente separati e definiti nel dominio frequenziale. Una soluzione pratica è quella di rilevare quanti più picchi possibile e delegare la decisione su quale sia una ben determinata parziale al successivo passo di analisi: l'algoritmo di peak continuation.

3.1.2 Pitch detection

Prima di poter "propagare" un insieme di traiettorie di picchi dal frame in esame, è utile cercare una possibile frequenza fondamentale. Se questa esiste, potremo disporre di più informazioni su cui lavorare, e ciò semplificherà e migliorerà l'operazione di tracciamento delle parziali.

La frequenza fondamentale può anche essere usata per regolare la dimensione della finestra di analisi, come sopra menzionato, in modo tale da mantenere costante il numero di periodi che vengono analizzati per ogni frame e ottenere il miglior compromesso possibile tra tempo e frequenza (analisi a pitch sincrono).

La frequenza fondamentale può essere definita come il divisore comune della serie armonica che meglio descrive i picchi spettrali. È possibile che il divisore comune non appartenga all'insieme dei picchi rilevati. Per questa ragione è preferibile parlare di *pitch*, cioè quella particolare frequenza che viene percepita come la principale di un suono. L'algoritmo che determina la frequenza fondamentale può essere descritto brevemente in tre passaggi pricipali: 1. scegliere i possibili candidati; 2. misurare la "bontà" della serie armonica risultante paragonata ai picchi spettrali; 3. ottenere il miglior candidato.

Un'adeguata stima d'errore per determinare la fondamentale è basata sulle differenze pesate tra i picchi calcolati e la serie armonica ideale (i picchi predetti).

3.1.3 Peak continuation

Dal processo di peak detection otteniamo alcune parziali "spurie" che non devono essere considerate. Una volta che i picchi spettrali del frame in esame sono stati rilevati, l'algoritmo di peak continuation li aggiunge alle traiettorie dei picchi del frame successivo. L'idea di base dell'algoritmo è che un insieme di "guide" avanza nel tempo attraverso i picchi spettrali, cercando le corrispondenze (rispettando i vincoli specificati) tra un frame e il successivo e generando infine da questi le traiettorie. Lo stato istantaneo delle guide, la loro frequenza e ampiezza vengono continuamente aggiornate: una guida viene generata, propagata ed infine terminata. Quando una frequenza fondamentale viene trovata in un frame le guide possono usare questa informazione per aggiornare i propri valori. Possono anche venire modificate a seconda dell'ultimo picco incorporato. Una volta che il procedimento di peak continuation è terminato per ogni frame del segnale in input l'analisi armonica è completata.

3.1.4 Analisi stocastica

Con riferimento alla formula (1), la componente stocastica *e(t)* del frame in esame è calcolata ri-generando il segnale deterministico tramite sintesi additiva, sottraendolo quindi dalla forma d'onda originale *s(t)* nel dominio del tempo.

Ciò è possibile perché nella sintesi del segnale deterministico vengono usate le fasi analizzate dal segnale originale e quindi la forma d'onda originale viene conservata. La rappresentazione stocastica si ottiene poi effettuando uno spectral fitting del segnale residuale.

3.2 Sintesi SMS

Il processo di sintesi SMS è descritto in figura 2, dove si possono vedere i due input (uno per la parte deterministica, l'altro per quella residuale) generati dall'analisi (ed eventualmente trasformati), come spiegato nella sezione 2.1.

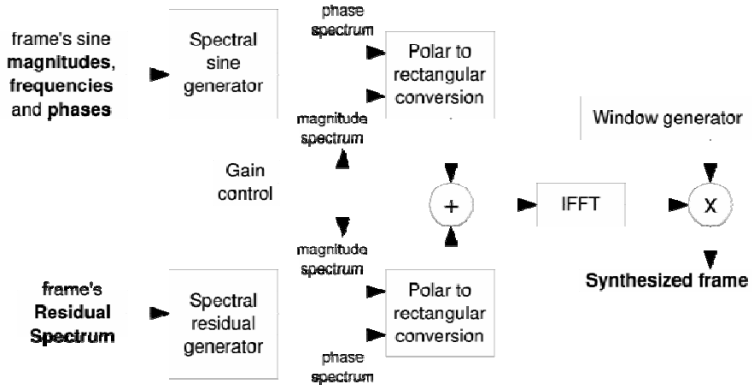

Figura 2: Diagramma di flusso per la sintesi SMS.

L'intero segnale viene elaborato nel dominio frequenziale, dove le due componenti vengono trattate separatamente, dopodiché tramite una FFT si ritorna nel dominio del tempo.

Per la componente deterministica l'obiettivo é ottenere lo spettro di una somma di sinusoidi moltiplicate per una finestra rettangolare, ma, per migliorare le prestazioni, preferiamo usare una Blackman-Harris 92dB. L'effetto di questa scelta viene corretto nel dominio temporale, dividendo per la stessa finestra. Scegliamo la finestra Blackman-Harris 92dB dato che permette di sintetizzare solo il lobo principale dello spettro, che contiene la maggior parte dell'energia del segnale.

Il segnale stocastico viene ottenuto creando uno spettro complesso per ogni inviluppo spettrale del residuo, cui viene filtrato del rumore bianco. Questa operazione viene effettuata nel dominio frequenziale creando uno spettro d'ampiezza a partire da quello analizzato (o una sua trasformazione) e generando uno spettro della fase con un random number generator.

Dato che vogliamo effettuare una singola iFFT per l'intero segnale, e dovremo correggere gli effetti della finestratura nel dominio temporale, abbiamo bisogno di applicare la stessa finestra anche alla componente stocastica. Così dobbiamo effettuare la convoluzione dello spettro del rumore con la finestra Blackman-Harris 92dB. Questo è implementato in modo molto efficiente perché coinvolge solo alcuni bins e la finestra è simmetrica.

A questo punto possiamo usare una singola iFFT per lo spettro combinato. Infine nel dominio temporale annulliamo gli effetti della Blackman-Harris 92dB e applichiamo la finestra triangolare nel processo di overlap and add, combinando frames consecutivi successivi per ottenere il segnale sintetizzato nel dominio del tempo.

4. ARCHITETTURA DEL SISTEMA DI SINTESI VOCALE

Il motore SMS effettua le operazioni di elaborazione digitale del segnale di un sistema text-to-speech, prendendo come input un file fonetico calcolato da FESTIVAL (Black & Taylor, 1997), che descrive la pronuncia di un testo scritto attraverso una sequenza di fonemi con lunghezza in ms e valori di intonazione in Hz.

Il nostro programma vuole essere una alternativa a MBROLA (Dutoit & Leich, 1993) e perciò utilizza lo stesso formato per l'input fonetico e gli stessi parametri da riga di comando. Entrambi i programmi implementano una sintesi concatenativa tramite difoni (che devono essere forniti separatamente) come unità base.

I tre passi logici su cui si basa l'intero processo di sintesi del segnale digitale sono:

1. **analisi**: è stata teoricamente descritta nella sezione 2.1 e consiste nel convertire il database di difoni (nel dominio temporale) in uno composto dai parametri spettrali salvati come file SDIF (Sound Description Interface Format);

2. **trasformazioni** e **concatenazione**: verrà descritto più avanti in questa sezione;

3. **sintesi**: è stata teoricamente descritta nella sezione 2.2.

La figura 3 mostra un diagramma a blocchi semplificato dei passi di trasformazione, concatenazione e sintesi.

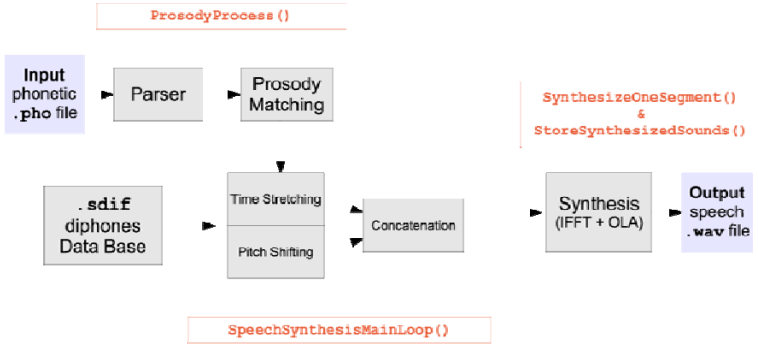

Figure 3: Schema del processo di sintesi vocale.

4.1 Trasformazioni SMS

Il compito del passo di trasformazione è di adattare ogni difono richiesto ai parametri specificati (per ogni fonema) nell'input fonetico. Attualmente tali parametri descrivono solamente la durata dei fonemi e l'evoluzione del pitch.

Le trasformazioni principali richieste sono time stretching e pitch shifting. Entrambe queste operazioni modificano l'ampiezza e la frequenza delle parziali del segnale. Questi nuovi valori diventano incoerenti con le fasi originali. Questo problema interessa il segnale ri-sintetizzato, deteriorandone la qualità audio.

Diventa così necessario ricalcolare le fasi delle sinusoidi; due modi per fare ciò sono disponibili: l'algoritmo di phase continuation e quello di relative phase delay propagation.

4.1.1 Time Stretching

Il processo di time stretching è basato sull'interpolazione e decimazione lineari di parametri spettrali tra i frames. Solo le frequenze e le ampiezze vengono interpolate.

L'algoritmo conserva l'integrità della transizione e l'intelligibilità tra fonemi differenti in ogni difono, senza però alterare le richieste di durata in input.

4.1.2 Pitch Shifting

Il pitch shifting è la trasformazione che si occupa di modulare l'intonazione della frase da pronunciare. Viene effettuata dopo il time stretching per meglio adattare i requisiti in input.

La routine di pitch shifting è implementata usando un algoritmo formant preserving, che cerca di preservare il timbro originale del suono. L'ampiezza di ogni parziale trasformata viene posizionata sull'inviluppo dello spettro originale, corrispondente alla frequenza originaria scalata di un fattore comune.

Alcune frequenze di parziali trasformate possono eccedere la larghezza di banda dello spettro originale e così la loro ampiezza dev'essere assegnata arbitrariamente. In particolare, ciò è significativo quando si debba effettuare lower pitch shifting, dato che le parziali "fuori banda" hanno un'ampiezza considerevole. In questo caso la nostra implementazione lascia il valore di ampiezza del pitch uguale a quello originale.

4.1.3 Phase Continuation

Il modo più semplice per ricostruire le fasi è chiamato phase continuation. Il suo comportamento è di assegnare arbitrariamente la fase di ogni parziale del primo frame e poi calcolare i valori frame dopo frame. In questa maniera l'algoritmo scarta tutti i dati d'analisi sulla fase. La formula usata per propagare i valori di fase è la seguente:

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

dove f_i^k e θ_i^k sono rispettivamente la frequenza e la fase della *k*-esima parziale dell'*i*-esimo frame, e *H* lo hop size.

4.1.4 Relative phase delay representation

Un metodo più complesso, teorizzato in (Di Federico, 1998), che aiuta a preservare la forma d'onda originale è basato sulla rappresentazione della fase tramite *relative phase delays*, definiti come differenza tra il *phase delay* (rapporto fase/pulsazione) delle parziali e il phase delay della fondamentale. Ciò rende la forma d'onda indipendente dalla fase della prima parziale.

Quando i relative phase delays sono stati calcolati per ogni frame è necessario propagare la fase della fondamentale modificata, come descritto dalla formula (2), e ricostruire la forma d'onda aggiungendo i relative phase delays al phase delay della nuova fondamentale.

Se questo metodo viene applicato senza pitch shifting e time stretching, l'unica differenza tra il suono originale e quello modificato sarà uno spostamento nel tempo della forma d'onda.

4.1.5 Concatenazione dei difoni

Una volta trasformati, due frames successivi devono essere concatenati. Questa operazione è basata principalmente sulla subroutine di interpolazione/decimazione usata per il time stretching, sebbene intervenga anche il pitch shifting.

Il comportamento di questa operazione è quello di effettuare un morphing tra gli ultimi frames di un difono e i primi del successivo. Ai pitch di tali frames vengono imposti uguali valori cosicché la frequenza e l'ampiezza possano essere interpolate. Ciò assicura una fusione più naturale tra quei frames di uno stesso fonema che appartengono a difoni consecutivi.

4.2 Il framework CLAM

Il framework CLAM (C++ Library for Audio and Music) (Amatriain *et alii*, 2002) offre soluzioni implementative per lo sviluppo di applicazioni audio attraverso un'architettura estensibile, generica ed efficiente. È perfettamente adatto per implementare il modello SMS.

CLAM ha semplificato molto il nostro lavoro grazie alla sua completezza e alla presenza di tutte le funzioni accessorie utili in un progetto per l'elaborazione audio (ad es. la gestione input/output dei file, salvataggio e visualizzazione di forme d'onda). Inoltre la sua ottima organizzazione ha permesso un facile adattamento al nostro scopo.

Il progetto CLAM è rilasciato sotto licenza GPL version 2 (o successive). È Platform Independent (compila sotto GNU/Linux, Windows e Mac) ed è così molto semplice creare applicazioni portabili.

5. CONFRONTI SMS-MBROLA

Sebbene il progetto sia ancora ad uno stadio preliminare, alcuni confronti con il sistema di sintesi concatenativa MBROLA sono già disponibili alla pagina "http://www.pd.istc.cnr.it/FESTIVAL/home/SMS.htm".

Abbiamo sintetizzato alcuni esempi dai medesimi file fonetici con il nostro motore e con MBROLA. Abbiamo anche usato database di difoni ottenuti dalle stesse registrazioni

(una raccolta di 1299 difoni della lingua italiana). Bisogna notare come la dimensione del database analizzato per il modello SMS è circa 10 volte maggiore rispetto a quella del database originale nel dominio del tempo (~70 MB contro 6.5 MB).

La sintesi tramite MBROLA è più rapida della nostra, ma vale la pena notare che non abbiamo ancora lavorato sull'ottimizzazione delle prestazioni.

Sia MBROLA che il nostro motore sintetizzano frasi ben intelligibili e il nostro sistema di concatenazione lavora bene anche in quei casi in cui vari fonemi vengono pronunciati rapidamente. In ogni caso la qualità audio della sintesi di MBROLA è spesso più nitida della nostra, che è spesso affetta da artefatti che la rendono più "roca". In figura 4 si possono osservare le forme d'onda e gli spettrogrammi della frase "il colombre" sintetizzata dal sistema SMS (a sinistra) e da MBROLA (a destra).

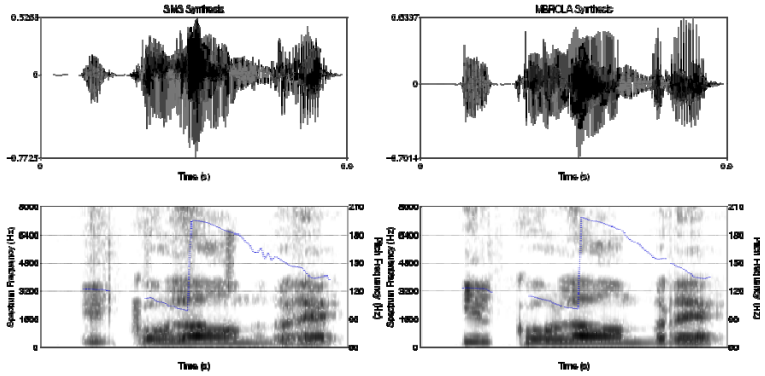

Figura 4: Confronto tra SMS e MBROLA.

6. CONCLUSIONI

Abbiamo presentato un nuovo motore di sintesi vocale concatenativa basato sul modello sinusoidale. Tale sistema si è dimostrato paragonabile, in termini di qualità audio e intelligibilità, con analoghi sistemi che costituiscono lo stato dell'arte per la sintesi concatenativa.

Stiamo valutando alcune strategie per migliorare qualità audio e prestazioni del motore SMS. Le più importanti sono:

- effettuare l'analisi in runtime, in modo da semplificare il processo stesso di analisi e diminuire la dimensione del database;
- introdurre un procedimento di verifica dell'analisi, allo scopo di correggere alcuni artefatti che possono verificarsi;
- implementare le operazioni del modello SMS a pitch sincrono;
- modificare o sostituire le funzioni di pitch shifting e time stretching;
- implementare trasformazioni audio basate su parametri di voice quality, come ad es. lo Spectral Tilt, allo scopo di effettuare sintesi emotiva.

7. RINGRAZIAMENTI

Vogliamo ringraziare gli sviluppatori di CLAM, in particolare Xavier Amatriain, David García Garzón e Pau Arumí per la loro disponibilità e i consigli preziosi.

8. BIBLIOGRAFIA

Sommavilla, G., Drioli, C. and Cosi, P. (2005), "Sintesi vocale concatenativa per l'italiano tramite modello sinusoidale", *Atti del Congresso AISV*.

Smith, J. O. and Serra, X. (1987), "PARSHL: An Analysis/Synthesis Program for non-Harmonic Sounds Based on a Sinusoidal Representation", *Proceedings of the International Computer Music Conference*.

McAulay, R. J. and Quatieri, T. F. (1986), "Speech analysis/synthesis based on a sinusoidal representation", *IEEE Transactions on Acoustics, Speech, Signal Processing*, vol. ASSP-34, pp. 744-754.

Stylianou, Y. (1998), "Concatenative speech synthesis using a Harmonic plus Noise Model", *The 3rd ESCA/COCOSDA Workshop on Speech Synthesis*, Jenolan Caves, NSW, Australia, Nov. 1998, Paper H.1.

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

Black, A. W. and Taylor, P. A. (1997), "The Festival Speech Synthesis System: System Documentation", *Human Communication Research Centre*, University of Edinburgh, http://www.cstr.ed.ac.uk/projects/festival.html.

Dutoit, T. and Leich, H. (1993), "MBR-PSOLA Text-To-Speech Synthesis based on an MBE Re-Synthesis of the Segments Database", *Speech Communication, Elsevier Publisher*.

Di Federico, R. (1998), "Waveform preserving time stretching and pitch shifting for sinusoidal models of sound", In *Proceedings of the COST-G6 Digital Audio Effects Workshop*.

Amatriain, X., Arumí, P. and Ramírez, M. (2002), "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).