# SINTESI VOCALE CONCATENATIVA PER L'ITALIANO TRAMITE MODELLO SINUSOIDALE

Giacomo Sommavilla, Carlo Drioli, Piero Cosi
Istituto di Scienze e Tecnologie della Cognizione - Sede di Padova "Fonetica e Dialettologia"
Consiglio Nazionale delle Ricerche
*giacomo.sommavilla@gmail.com, drioli@pd.istc.cnr.it, cosi@pd.istc.cnr.it*

## 1. SOMMARIO

Viene descritta una applicazione per la sintesi vocale dell'**italiano**. È progettata per implementare la parte di elaborazione del segnale di un sistema TTS, ovvero quella che si occupa di creare un *file* audio (.wav) a partire da un *file* fonetico (.pho) relativo alla frase da pronunciare.
A tale scopo viene utilizzato un *database* di difoni pre-registrati, dalla cui **concatenazione** si ottiene l'*output* audio. Le operazioni sui difoni vengono effettuate secondo **tecniche sinusoidali** (più precisamente tecniche SMS, *Spectral Modeling Synthesis*).
Il *database* di difoni è quello usato da MBROLA, applicazione di sintesi vocale nel dominio temporale. Tale programma è stato usato in alcuni test di confronto.
È stato utilizzato lo *Spectral Modeling Synthesis* (SMS) come modello spettrale per l'elaborazioni del segnale, ed in particolare il *framework* CLAM per l'implementazione *software*.

## 2. INTRODUZIONE

*2.1 Cosa significa Text-To-Speech (TTS)*
Un sintetizzatore *Text-To-Speech* (TTS)[1] è una applicazione *software* capace di leggere qualsiasi testo scritto. La differenza fondamentale con altri sistemi di riproduzione audio della voce umana (come un lettore CD per esempio) è che il sistema di cui stiamo parlando è in grado di leggere *nuove* frasi.
L'algoritmo di conversione *da testo a voce* è costituito da due parti fondamentali:

1. analisi del linguaggio naturale (*Natural Language Processing*, NLP) e
2. elaborazione del segnale (*Digital Signal Processing*, DSP).

Nel nostro caso abbiamo lavorato solamente sulla seconda parte, in particolare sulle seguenti tre sezioni:

- **Prosody matching** trasforma i parametri di prosodia (intonazione e durate delle parole) calcolati nell'analisi del linguaggio naturale in variabili che serviranno per pilotare le funzioni di *pitch shifting* e *time stretching*;
- **Diphone Concatenation** si occupa di concatenare difoni successivi;
- **Signal Synthesis** effettua la anti-trasformata del segnale dal dominio frequenziale a quello temporale.

Abbiamo fatto uso di *difoni*, ovvero "segmenti acustici che includono la transizione fra due fonemi consecutivi", come elementi base per la costruzione delle parole, di cui abbiamo a disposizione un *database* di circa 1300 *files* audio. Il numero è così elevato poiché è necessario un difono per ogni possibile transizione da un fonema all'altro.

[1] Per maggiori dettagli si veda (Dutoit, 2002).

*2.2 Sintesi in frequenza*

L'applicazione *software* lavora su un secondo *database*, creato a partire dal primo, che si occupa di salvare i parametri dell'analisi SMS *(Spectral Modeling Synthesis*) di ogni difono in un corrispondente *file* .sdif. Il formato SDIF (*Sound Description Interface Format*) è adatto ai nostri scopi perché permette una compressione *lossless* dei parametri ricavati dalla analisi SMS, incapsulati nella rappresentazione frequenziale di *frames* audio consecutivi. I più importanti di essi sono tre vettori che memorizzano ampiezze, frequenze e fasi delle parziali. Inoltre ad ogni *frame* è associato anche un inviluppo spettrale per parte residuale.

Per sviluppare le funzioni di analisi e sintesi abbiamo fatto uso del *framework* CLAM (ideale per l'approccio SMS), realizzato dal Music Technology Group (MTG) della Universitat Pompeu Fabra di Barcellona.

Nel nostro caso i vantaggi che derivano dall'operare nel dominio della frequenza (rispetto alle operazioni effettuate nel dominio temporale) e in particolare usando la tecnica SMS sono:

- una rappresentazione del segnale più versatile e potente per le elaborazioni in questione (tempo, *pitch*, e inviluppo spettrale)[2];
- possibilità di sfruttare i risultati dell'analisi SMS nella concatenazione dei difoni;
- la distinzione tra parte sinusoidale (deterministica) e residuale (stocastica) permette di trattare diversamente fonemi vocalizzati (*voiced phonemes*) e non.

Abbiamo sviluppato l'applicazione in modalità "Console" (ovvero da riga di comando, senza interfaccia grafica), in modo tale che le sue funzionalità possano essere espanse (struttura *modulare* del codice sorgente). Per questo motivo abbiamo anche ridotto al minimo le librerie esterne, ad esempio abbiamo tolto la dipendenza dal formato XML, normalmente in uso all'interno del *framework* di CLAM.

## 3. MODELLO SINUSOIDALE

*3.1 Elaborazione e rappresentazione spettrale*

Il modello sinusoidale che abbiamo adottato in questo lavoro si basa sulla *Short Time Fourier Transform* (STFT)[3]. Un aspetto molto importante da tenere in considerazione è che tale operazione divide il segnale da elaborare in una serie di *frames*, tecnica questa che permette di apprezzare le variazioni nel tempo delle caratteristiche spettrali del segnale da analizzare.

La formula matematica che rappresenta la STFT è data da

$$X_l(k) = \sum_{n=0}^{N-1} w(n)x(n+lH)e^{-j\omega_k n}$$

per *l=0,1,...*, dove *w(n)* è la finestra d'analisi nel tempo, *l* l'indice del frame e *H* l'*hop size*.

Introduciamo adesso il modello spettrale[4] secondo il quale il suono può essere visto come la somma di una parte sinusoidale (armonica) e di una parte residuale (stocastica, *rumorosa*):

---

[2] La complessità computazionale è maggiore visto che sono coinvolte la ricostruzione dello spettro e una IFFT.

[3] Si vedano, per approfondimenti, Deller *et al*. (1993); Rabiner & Schafer (1978).

[4] Strettamente legati a questo, si vedano McAulay & Quatieri (1984); Smith & Serra (1987), e soprattutto Stylianou (1998).

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

*3.2 Cos'è la tecnica SMS?*
La tecnica SMS (Spectral Modeling Synthesis), descritta in Serra (1997), è un insieme di tecniche e implementazioni *software* per l'analisi, la trasformazione e la sintesi sonora. Essa si basa principalmente sulla rappresentazione sinusoidi + residuo del segnale. Tramite questa modellizzazione si possono ottenere rappresentazioni generali e acusticamente significative basate sull'analisi, i cui parametri estratti possono essere manipolati mantenendo un'alta qualità sonora.
Queste tecniche vengono usate per l'analisi, la trasformazione e la sintesi del segnale, e sono particolarmente efficaci nel caso elaborazione dell'audio e della musica, come spiegato in Serra & Smith (1990).

*3.3 Analisi SMS*
La figura 1 mostra il diagramma a blocchi per l'analisi. Per prima cosa il segnale da analizzare viene moltiplicato per l'appropriata finestra d'analisi[5] e il suo spettro calcolato tramite la FFT (queste due operazioni equivalgono a una STFT), vengono individuati i principali picchi spettrali, tramite un algoritmo di *peak detection*, vedi Maher & Beauchamp (1994), e incorporati nelle traiettorie delle parziali grazie a un algoritmo di *peak continuation*.

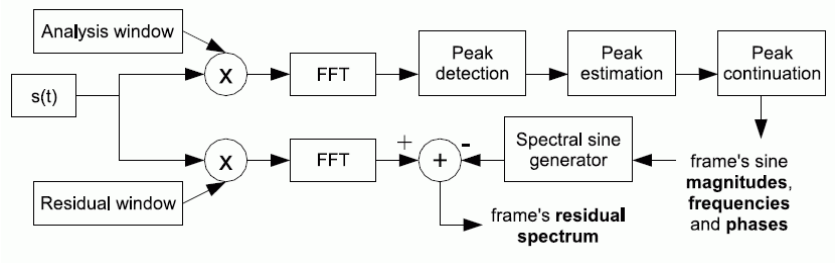


Figura 1: schema a blocchi dell'analisi SMS.

Questo algoritmo permette di conoscere l'ampiezza, la frequenza e la fase delle parziali presenti della **parte deterministica**. Assumendo il suono pseudo-armonico, l'eventuale calcolo del *pitch* può migliorare l'analisi utilizzando l'informazione relativa alla frequenza fondamentale (analisi *pitch*-sincrona[6], non utilizzata in questo lavoro).
La **componente stocastica** del *frame* corrente è calcolata generando il segnale deterministico tramite sintesi additiva e poi sottraendolo dalla forma d'onda originale nel dominio del tempo.

*3.4 Trasformazioni SMS*
Dallo *step* di analisi otteniamo le parziali e l'inviluppo spettrale del residuo. Possiamo così manipolare questi elementi prima di utilizzarli per il passo successivo (sintesi). Questo fa sì

[5] Nel nostro caso abbiamo usato una Blackman-Harris 92dB.
[6] Come in Moulines & Charpentier (1990).

che le trasformazioni siano indipendenti dalla sintesi e quindi anche *modulari*, in modo tale da poter essere riscritte o sostituite in un secondo tempo da operazioni *ad hoc*.
Basicamente le operazioni utilizzate in questo lavoro sono due: il *time stretching* (si applica sia alla parte sinusoidale che alla parte residuale) e il *pitch shifting* (solo alla parte sinusoidale)[7]. In figura 2 si trova una schematizzazione del processo di trasformazione SMS.

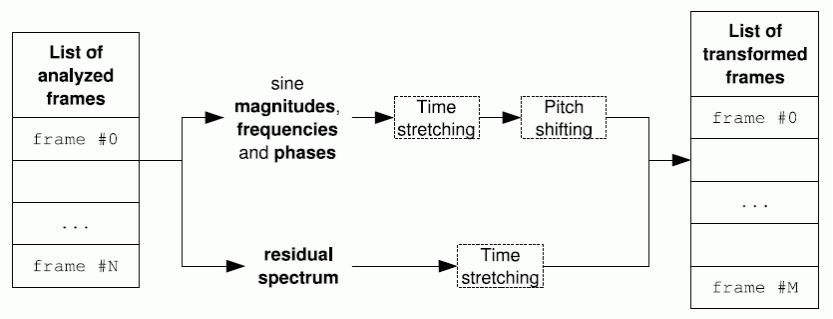


Figura 2: schema trasformazioni SMS.

*3.5 Sintesi SMS*
Consideriamo a questo punto la sintesi delle due componenti ottenute dai processi di analisi e trasformazione (in figura 3 lo schema a blocchi di queste operazioni):
Il **segnale deterministico** sintetizzato è calcolato a partire dalle traiettorie delle ampiezze e delle frequenze generando un'onda sinusoidale per ogni traiettoria (i.e., sintesi additiva) nel dominio frequenziale usando la trasformata inversa di Fourier. La finestra di risintesi utilizzata è la Blackman-Harris92dB, che contiene gran parte della sua energia nel lobo principale.
Il **segnale stocastico** sintetizzato risulta dal filtraggio di un segnale rumoroso con l'inviluppo spettrale del residuo ottenuto nell'analisi (i.e., sintesi sottrattiva). Può essere implementato nel dominio frequenziale creando uno spettro complesso (i.e., la coppia {spettro delle ampiezze, spettro delle fasi}) per ogni inviluppo spettrale del residuo e quindi calcolando la trasformata inversa.

[7] Queste trasformazioni verranno descritte più dettagliatamente nel prossimo paragrafo.

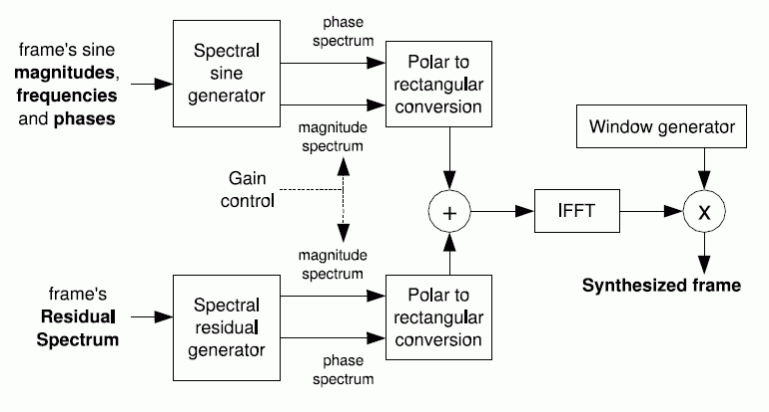


Figura 3: schema a blocchi della sintesi SMS.

*3.6 Il framework CLAM*

CLAM è un *framework* multi-piattaforma (Linux, Windows, Mac OS) in linguaggio C++ per l'elaborazione audio e musicale del segnale, sviluppato dal Music Technology Group (MTG)[8] dell'Universitat Pompeu Fabra di Barcellona. CLAM è un *software* Free *Software* rilasciato sotto licenza GNU. È progettato per essere orientato ad oggetti (OO), efficiente e per essere usato per applicazioni in tempo reale.

Tale *framework* si è rivelato estremamente utile per la nostra applicazione, dato che è basato sul modello sinusoidi + residuo: vi sono infatti classi e funzioni ottimizzate per eseguire le operazioni di analisi, traformazione e sintesi SMS.

Una caratteristica peculiare di CLAM è l'implementazione di classi *dynamic type*: si possono così creare e distruggere attributi di oggetti in esecuzione. Un esempio di *dynamic type* è dato dalla classe `CLAM::Segment` (la classe che nella nostra applicazione è associata ai parametri spettrali dei difoni).

In figura 4 si possono vedere le prime righe della sua definizione:

```
class Segment : public ProcessingData
{
   public:
   DYNAMIC_TYPE_USING_INTERFACE (Segment, 8, ProcessingData);
   DYN_ATTRIBUTE (0, public, TTime, BeginTime);
   DYN_ATTRIBUTE (1, public, TTime, EndTime);
   DYN_ATTRIBUTE (2, private, bool, prHoldsData);
   DYN_ATTRIBUTE (3, private, List<Frame>, prFramesArray);
   DYN_ATTRIBUTE (4, public, Audio, Audio);
   DYN_ATTRIBUTE (5, public, List<Segment>, Children);
   DYN_ATTRIBUTE (6, public, TData, SamplingRate);
   DYN_ATTRIBUTE (7, public, std::string, Id);
     (...)
}
```

Figura 4: definizione della struttura dati `CLAM::Segment`.

[8] (MTG) http://www.iua.upf.es/mtg.

## 4. *DATABASE* DI DIFONI

*4.1 Cos'è un difono?*

Un difono è un *segmento acustico che contiene la transizione tra due fonemi consecutivi*. È normalmente usato con riferimento alla registrazione di una transizione da un fonema ad un altro.

Se il numero di fonemi in un linguaggio è *P*, allora il numerico teorico di difoni possibili è $P^2$, sebbene in ogni idioma sia di norma molto minore di tale valore, poiché in tutte le lingue ci sono restrizioni circa quali suoni possono succedere ad un altro. Per esempio lo spagnolo ha circa 800 difoni, mentre il tedesco circa 2,500. Il nostro *database* italiano ne ha 1300, ed è il *database* usato per la sintesi vocale tramite il programma MBROLA[9].

I difoni sono utili nella sintesi vocale perché combinando difoni pre-registrati il parlato suona molto più naturale che combinando semplici fonemi, dato che la pronuncia di ogni di questi dipende dai suoni che lo precedono e lo seguono.

*4.2 Il database di difoni nel tempo (file .raw)*

Il *database* di difoni nel tempo è costituito da *file* .raw. Questo tipo di *file* non ha intestazione, come invece succede per i *file* .wav, cosicché dobbiamo ricordarne i parametri di registrazione per poterne interpretare correttamente i dati:

sampling rate: 16000 Hz
risoluzione: 16-Bit
canali: 1 (mono)

Questo *database* è fornito di un *file* (*export.dat*) che tiene memoria dei parametri di ciascun difono. Ad esempio, le prime righe di questo *file* sono:

```
d0.raw  #   #   500 3700    2100
d1.raw  l   e1  500 3333    1404
d2.raw  tS  tS  500 1370    842
d3.raw  z   g   500 2439    1759
d4.raw  z   m   500 2887    1973
...
```

In ogni riga, la prima colonna indica il nome del *file* (ad es. "d0.raw"), poi vengono esplicitati i due fonemi componenti il difono (nella prima riga vediamo quel particolare difono che concatena due *silenzi*, ognuno dei quali etichettato con "#"). Dopodiché incontriamo tre numeri (per ognuno di essi l'unità di misura è il *sample*, campione nel tempo): il primo (si noti che vale 500 per ogni difono) indica il numero di *samples* usati per l'analisi nel dominio del tempo di MBROLA; il secondo indica la lunghezza della forma d'onda meno i 500 campioni dell'analisi; il terzo ed ultimo numero è un indice che segna la divisione tra i due fonemi.

*4.3 Il database di difoni in frequenza (file .sdif)*

Un semplice *script* chiama la funzione di analisi SMS per ogni *file* .raw. Questo programma salva in un *file* .sdif[10] i parametri spettrali armonici e stocastici di ogni difono.

---

[9] Programma di sintesi vocale nel dominio del tempo sviluppato dalla *Faculte Polytechnique de Mons, TCTS Lab*, si vedano (MBROLA, 2005; Dutoit & Leich, 1993).

[10] SDIF (*Sound Description Interchange Format*) è il formato usato da CLAM per salvare i dati dell'analisi SMS, si veda De Boer *et al*. (2000).

Vale la pena notare che attualmente il *database* analizzato misura circa 14 volte di più di quello nel dominio del tempo (140 Mbytes di *file* .sdif contro i 10 di *file* .raw). Questo risultato può sembrare piuttosto grave. In questa sede non ci siamo comunque preoccupati di risparmiare spazio su disco, essendo molto più interessati all'aspetto di elaborazione del segnale. Comunque alcuni semplici test che abbiamo eseguito dimostrano come sia possibile diminuire sensibilmente le dimensioni dei *file* .sdif abbassando il limite superiore della frequenza delle parziali da cercare.

*4.4 Trasformazioni sui difoni*

Come anticipato precedentemente, le uniche due trasformazioni di cui abbiamo fatto uso sono il *pitch shifting* e il *time stretching*:

**Pitch Shifting**: è la trasformazione che si preoccupa di modulare l'intonazione della frase da pronunciare. Questa operazione semplicemente moltiplica tutte le frequenze per un numero fissato, mentre i valori di ampiezza meritano un algoritmo specifico, allo scopo di preservare il piú possibile il timbro vocale originale. Questo tipo di *pitch shifting* viene chiamato *Formant Preserving*, dato che interpola l'ampiezza di ogni picco trasformato tra le ampiezze dei due picchi (del suono originale) con frequenza più vicina. In figura 5 si può vedere un *pitch shifting* verso l'alto (voce più acuta): le sinusoidi in rosso sono quelle trasformate. Il caso di *pitch shifting* verso il basso (voce più grave) è più complicato, dato che l'ampiezza della frequenza fondamentale (e di altre eventualmente) dev'essere scelta arbitrariamente[11].

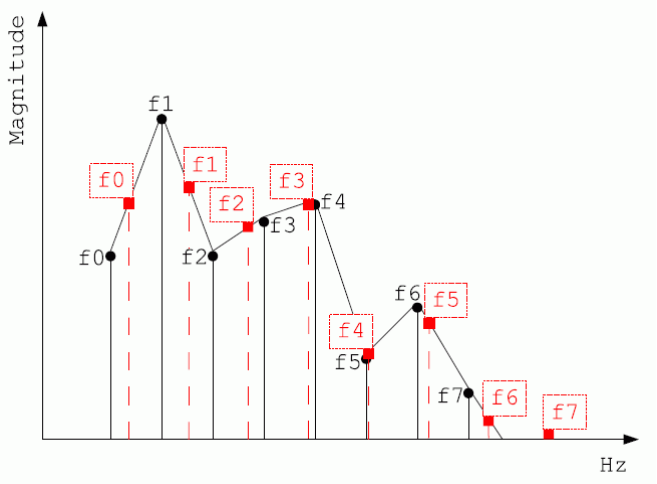


Figura 5: *Pitch Shifting* con *Formant Preserving*.

**Time Stretching**: questa operazione si basa su una funzione che opera su due *frame* consecutivi, crando da zero il nuovo *frame*. In sostanza questa funzione calcola i valori di ampiezza e frequenza del *nuovo frame* (chiamiamolo per semplicità *F**, con ampiezze $a_i$* e frequenze $f_i$*) a partire dai valori dei *frame* precedente (chiamato *F1*, con ampiezze $a_i1$ e frequenze $f_i1$) e successivo originali (*F2*, con ampiezze $a_i2$ e frequenze $f_i2$). I valori di ampiezza e frequenza di *F1* e *F2* peseranno di più o di meno a seconda della vicinanza nel

[11] Nella nostra implementazione abbiamo lasciato il valore di ampiezza del *pitch* trasformato uguale a quello del *pitch* originale.

tempo con *A** (ad es. se *F** e *F1* sono vicini allora i valori $f_i$* e $a_i$* saranno simili a $f_i1$ e $a_i1$).
Come si può notare, solamente ampiezze e frequenze vengono calcolate dagli algoritmi appena descritti. Per quanto riguarda le fasi, esse devono essere ri-allineate[12] tramite una operazione successiva alle trasformazioni di *pitch shifting* e *time stretching*. Tale procedimento, che prende il nome di *phase propagation*, implementa la seguente formula (*i* è l'indice del frame da calcolare):

$$phase(i) = phase(i-1) + \pi * [freq(i-1) + freq(i)] * hopSize$$

*4.5 Concatenazione dei difoni*
Una volta che due difoni consecutivi sono stati correttamente trasformati, bisogna concatenarli. L'algoritmo che si occupa di eseguire questa operazione deve rendere meno brusco il passaggio tra i due difoni mediando i *frames* finali del primo con i *frames* iniziali del secondo. Viene chiamata per questo la stessa funzione di interpolazione usata nell'algoritmo di *time stretching*.

## 5. ARCHITETTURA DEL SINTETIZZATORE VOCALE

*5.1 Struttura della sintesi vocale*
L'applicazione che implementa il modulo DSP (*Digital Signal Processing*) del nostro motore di sintesi si basa principalmente sulle seguenti operazioni:
`ProsodyProcess()` è un insieme di operazioni che possono essere divise in due parti: 1) **Elaborazione dei dati fonetici** e 2) **Prosody Matching**, che verranno esaminate in dettaglio nei prossimi sottoparagrafi. Il suo scopo è quello di trasformare ogni difono affinché la frase sia intonata seguendo i valori prosodici salvati nel *file* fonetico;
`SpeechSynthesisMainLoop()` si occupa di caricare i difoni in memoria, trasformarli e infine concatenarli;
`SynthesizeOneSegment()` chiama le funzioni del *framework* CLAM che produrranno i campioni audio nel tempo, salvati in tre strutture dati: mAudioOut (l'audio in *output*), mAudioOutSin (la componente sinusoidale dell'*output*) e mAudioOutRes (la componente residuale);
`StoreSynthesizedSounds()` è la funzione che semplicemente converte le sequenze di campioni nei tre corrispondenti .wav *files*.

*5.2 Elaborazione dei dati fonetici*
L'*input* fonetico è passato alla nostra applicazione tramite un *file* .pho, (di formato uguale a quelli usati da MBROLA) che contiene una sequenza di fonemi e valori prosodici.

[12] Le operazioni di *time stretching* e *pitch shifting* usate rendono privi di significato i valori originali di fase, che devono scartati e ricalcolati. La routine di *phase propagation* risolve il problema della fastidiosa distorsione provocata dall'uso delle fasi originali.

```
;;F=1.1
;;T=1.2
;F=1.3
;T=1.4

_    25    0 143   100 143
a1   224           20  139.45  60 122.34  100 119.53
v    74.66
a    213.3
_    25    0 150
```

Figura 6: esempio di *file* .pho.

Osserviamo l'esempio in figura 6: le prime righe del *file*, che cominciano con "`;;`", memorizzano dei valori di durata e frequenza che sono fattori moltiplicativi da applicare ad ogni fonema della frase. Tutti questi comandi sono una sorta di intestazione del *file*. Le righe che cominciano con "`;`" sono considerate *commenti* dalla *routine* di *parsing* (i loro valori non verranno quindi presi in considerazione).
La descrizione della frase e della prosodia è lasciata a quelle righe che cominciano con il simbolo che indica la rappresentazione grafica dei fonemi. Per ognuno di questi è data anche la durata in millisecondi e una sequenza di *Pitch Point Pairs* (coppie di numeri: il primo rappresenta la posizione in percentuale all'interno del fonema e il secondo il valore di frequenza richiesto in Hz).
Per esempio il fonema "a1" sarà pronunciato, al 20% della sua durata, alla frequenza di 139.45 Hz, e così via, finché il *parser* non incontra un fonema senza *Pitch Point Pair*. Questa mancanza di informazione frequenziale viene riempita con l'inserzione di due valori frequenziali per ogni fonema: uno iniziale e uno finale.

*5.3 Prosody Matching*
Una volta che il *file* fonetico è stato letto e le informazioni salvate in una apposita struttura dati bisogna riempirne un'altra, associata alla sequenza di difoni corrispondente. Basicamente, la struttura dati relativa ad ogni difono nella catena è costituita da due vettori di *floating point*, `timeControl[d][i]` e `pitchControl[d][i]` (`i` compreso tra 0 e il numero di *frame* del `d`-esimo difono meno uno). Questi *array* tengono memoria del fattore di *time stretching* e di quello di *pitch shifting*, rispettivamente, da applicare all'`i`-esimo *frame* del difono.

*5.4 Speech Synthesis Main Loop*
Una volta che l'informazione prosodica è stata correttamente inserita nella struttura dati relativa ai difoni, il sistema è pronto per eseguire l'ultima serie di operazioni.
La *routine* principale di sintesi vocale è basicamente un ciclo `while` che salva due difoni consecutivi, li trasforma e infine li concatena.
Questa è la sequenza di istruzioni eseguite:

1. inizializzazione delle variabili e caricamento del primo difono nella relativa struttura dati CLAM (`CLAM::Segment`);
2. trasformazione del difono;
3. inizio del ciclo while, il nuovo difono viene caricato nel segment#2 (previamente cancellato) e trasformato;

a questo punto è possibile concatenare i due difoni, salvare il risultato in un'altra struttura tipo *buffer*, copiare il segment#2 nel segment#1 e ricominciare il ciclo finché si esaurisce la catena di difoni.
La funzione si può riassumere in pseudocodice come in figura 7:

```
SpeechSynthesisMainLoop()
{
                                        i = 0;
        // index of diphoneINIT mNumberOfDiphones;
        // set lengthLOAD Diphone[ 0 ] INTO segment1;
        // load first diphone
        TRANSFORM( segment1, timeControl[0], pitchControl[0] );
        i = 1;
        while( i < numberOfDiphones )
        {
                FLUSH segment2;
                LOAD Diphone[ index ] INTO segment2;
                TRANSFORM( segment2, timeControl[i], pitchControl[i] );
                CONCATENATE( segment1, segment2 );
                COPY segment2 INTO segment1;
                i++;
        }
}
```

Figura 7: pseudocodice del *Synthesis Main Loop*.

## 6. TEST E CONCLUSIONI

In questo paragrafo proporremo alcuni risultati di sintesi vocale SMS (prodotti con la nostra applicazione). Queste frasi verranno messe a confronto con le corrispondenti prodotte dal *software* MBROLA per l'italiano. Bisogna sottolineare che i confronti si basano sui medesimi *file* fonetici di ingresso, ed inoltre il *database* di partenza è lo stesso.
La frase da sintetizzare è "*Era una giornata splendida di sole; e il mare tranquillo.*", tratta dal racconto "Il colombre" di Dino Buzzati (1906, 1972).
Il primo test effettuato fa riferimento al *file* fonetico (prodotto in *stile narrativo*)[13] riportato in figura 8.

```
;T=1.5     ;F=1.5
 _  715 0 79
E1 200 0 79 50 119 100 119      r 50 0 119             a 60 50 116 100 114
u1 50 0 114 50 114 100 113      n 70 0 113             a 40 50 107 100 102
dZ 120 0 102                    o 30 50 104            r 70 100 100
n 90 0 100                      a1 140 50 102 100 98   t 60 0 98
a 60 50 93 100 87               s 110 0 87             p 80
l 90                            E1 120 50 94           n 80 100 87
d 30 0 87                       i 50 50 85 100 80      d 90 0 80
a 30 50 74 100 73               d 60 0 73              i1 30 50 75 100 73
s 130 0 73                      o1 180 50 72 100 72    l 50 0 72
e 110 50 72 100 98              _ 250                  e1 100 0 98 50 98 100 101
i1 90 0 101 50 101              l 40 100 92            m  120 0 92
a1 190 50 72 100 71             r 50 0 71              e 90 50 72 100 92
_ 130                           t 60 0 92              r 50
a 50 50 81                      ng 110 100 71          k 70 0 71
w 60 i1 120 50 71               l 85 100 71            l 85 0 71
o 100 50 71 100 105               650 0 105
```

Figura 8: *file* fonetico relativo alla frase "*Era una giornata splendida di sole; e il mare, tranquillo*.".

[13] Per esigenze di spazio il *file* è stato qui formattato su tre colonne.

Di seguito alleghiamo la frase prodotta dai due *sofware*:

(1) MBROLA:  (2) sintetizzatore SMS: 

Il risultato è buono nel senso che la qualità delle due sintesi è paragonabile. Ciò che è importante osservare è la semplicità dei mezzi utilizzati dal sintetizzatore SMS. A questo proposito sono orientati anche i due esempi seguenti: infatti abbiamo implementato due semplicissime trasformazioni che fanno uso della tecnica SMS.
Nella prima sono stati cancellati dai *frame* i picchi spettrali di ordine dispari (il primo, il terzo, ...), mentre nella seconda sono stati cancellati quelli di ordine pari. Nel primo caso (3) si è ottenuta una trasformazione del tutto equivalente ad un *pitch shift* (sebbene di più semplice implementazione) verso l'alto di un'ottava, mentre nel secondo (4) il risultato assomiglia molto ad una sorta di *morphing* tra la voce del parlatore e il suono di uno strumento a fiato (clarinetto o flauto). Quest'ultima operazione è interessante per il fatto che la sua implementazione in un sintetizzatore nel dominio del tempo (come MBROLA) sarebbe molto più complicata.

(3) salto di un'ottava:  (4) *morphing* voce/flauto: 

Gli ultimi due esempi che riportiamo sono un altro confronto diretto con MBROLA. In questo caso sono state aggiunte due righe nell'intestazione del *file* .pho:
;;T=1,5
;;F=2
Ciò significa che la frase è stata velocizzata di 1,5 volte e la sua intonazione alzata di un'ottava. Questi i risultati:

(5) MBROLA:  (6) Sintetizzatore SMS: 

## 7. BIBLIOGRAFIA


De Boer, M.; Bonada, J.; Serra, X., 2000. Using the Sound Description Interchange Format within the SMS Applications. In *Proceedings of the International Computer Music Conference 2000*, Berlin, Germany.

Deller, J. R.; Proakis, J. G.; Hansen, J. H., 1993. *Discrete Time Processing of Speech Signals*. New York: Prentice Hall PTR.

Dutoit, T., 1997. High-quality text-to-speech synthesis: an overview. In *Journal of Electrical and Electronics Engineering, Australia: Special Issue on Speech Recognition and Synthesis*, 17, 1, 25-37.

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

Dutoit, T., 2002. A short introduction to Text-To-Speech synthesis, http://tcts.fpms.ac.be/synthesis/introtts.html.

MBROLA, 2005. The MBROLA Project, http://tcts.fpms.ac.be/synthesis/mbrola.

Maher, R. C.; Beauchamp, J. W., 1994. Fundamental Frequency Estimation of Musical Signals using a two-way Mismatch Procedure. *Journal of the Acoustical Society of America*.

McAulay, R. J.; Quatieri, T. F., 1984. Magnitude-only Reconstruction using a Sinusoidal Speech Model. In *Proceedings of the Institute of Electrical and Electronics Engineers International Conference on Acoustics, Speech and Signal Processing*. New-York: IEEE Press.

Moulines, E.; Charpentier, F., 1990. Pitch Synchronous waveform Processing techniques for Text-To-Speech Synthesis using diphones. *Speech Communication*, 9, 5-6, December 1990, 453-467.

Music Technology Group (MTG), Universitat Pompeu Fabra (UPF) Institut de l'Audiovisual (IUA) Barcelona, *CLAM On-line Documentation*, http://www.iua.upf.es/mtg/clam.

Rabiner, L. R.; Schafer, R. W., 1978. *Digital Processing of Speech Signals*. New Jersey: Prentice Hall PTR.

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

Serra, X., 1997. Musical Sound Modeling with Sinusoids plus Noise. In C. Roads, S. Pope, A. Picialli, G. De Poli (a c. d.) *Musical Signal Processing*, Swets & Zeitlinger Publishers.

Smith, J. O.;Serra, X., 1987. PARSHL: An Analysis/Synthesis Program for non-Harmonic Sounds Based on a Sinusoidal Representation. In *Proceedings of the International Computer Music Conference*, 290-297.

Stylianou, Y., 1998. Concatenative speech synthesis using a Harmonic plus Noise model. In *Proceedings of the III ESCA/COCOSDA International Workshop in Speech Synthesis*, Jenolan Caves, Australia, 261-266.