

Capitolo 8

Analisi dei suoni

versione 2004

8.1 Introduzione

In questo capitolo si descrivono i principali metodi di analisi del suono con lo scopo di estrarre informazione dal suono. Nei capitoli sui modelli del suono, lo scopo era quello di ottenere delle rappresentazioni che ne consentissero una auspicabilmente completa ricostruzione, almeno dal punto di vista percettivo o operativo. Ai modelli erano quindi associati metodi di analisi, per individuare i parametri dei modelli, che ne consentissero questa descrizione senza perdere, per quanto possibile nessuna caratteristica acustico o timbrica del suono. Spesso infatti i modelli con i parametri individuati, vengono usati per la sintesi o trasformazione dei suoni. Ad esempio dalla rappresentazione mediante modelli spettrale si riesce a ottenere variazioni di durata e altezza molto naturali.

Se lo scopo è invece quello di estrarre informazione dal suono, allora si vuole scartare tutto quello che non è rilevante al proprio scopo. Si parla allora di estrazione di caratteristiche (feature extraction). I passi principali per la estrazione di informazione sono pre-elaborazione del suono, selezione dei frames mediante finestre, estrazione di caratteristiche, post-elaborazione.

- La prima fase di pre-elaborazione consiste nel modificare il segnale, in modo ad facilitare gli algoritmi di estrazione delle caratteristiche. Ad esempio, riduzione del rumore, equalizzazione, filtraggio passa basso. Nei suoni vocali si usa spesso la pre-enfasi mediante un filtro passalto per appiattare lo spettro, cioè per avere una distribuzione più omogenea dell'energia nelle varie zone dello spettro. Infatti la voce ha un andamento spettrale di tipo passabasso.
- Il secondo passo consiste nel dividere il segnale in blocchi (detti frames) parzialmente sovrapposti ed applicare una finestra per ogni blocco, come discusso nei modelli spettrali.
- Il terzo passo consiste nel ricavare, per ogni frame un vettore di caratteristiche, con gli algoritmi descritti nelle prossime sezioni.
- Infine nella fase di post-elaborazione, si scelgono le caratteristiche più significative dal vettore, eventualmente pesandole diversamente, per dare loro maggiore o minor importanza. Ad esempio si può pesare meno i primi coefficienti mel-cepstrali, se è presente un rumore a bassa

frequenza. Infine spesso i vettori delle caratteristiche sono normalizzati rispetto al tempo, in modo ad avere media nulla. questa normalizzazione costringe i vettori ad essere nello stesso ordine di grandezza numerica.

I metodi di analisi procedono direttamente dal segnale rappresentato nel tempo oppure derivano una rappresentazione dello spettro. Nel secondo caso si inizia applicando le tecniche viste, quando sono stati descritti i modelli spettrali e i loro derivati come parte armonica + parte stocastica, parte armonica + rumore + transitori si ottiene una rappresentazione a basso livello, che costituisce la premessa per la estrazione di parametri e attributi a livello superiore. In questo capitolo verranno prima presentati alcuni metodi base per la stima di parametri nel dominio temporale, poi verrà discusso l'importante problema della stima dell'involuppo spettrale. Verranno quindi presentati vari descrittori di parametri e attributi a vari livelli ricavabili da queste rappresentazioni e utilizzabili per la descrizione dei suoni e per un'ulteriore analisi volta a separare e comprendere i suoni e la loro organizzazione. Infine verranno presentati alcuni algoritmi base per la determinazione dell'inizio (onset) degli eventi musicali,

8.2 Parametri nel dominio del tempo

Nell'ambito dell'elaborazione del segnale è spesso conveniente ricorrere all'uso di parametri che ne descrivano alcune caratteristiche fondamentali; ciò è importante in molti casi di archiviazione o trattamento del suono, perchè elimina le ridondanze presenti nel segnale audio e permette di ottenere una efficiente rappresentazione e quindi una semplificazione nella manipolazione dei dati. Un aspetto importante in molte applicazioni è la variazione dei valori di questi parametri nel tempo (per esempio nella sintesi del parlato); nel seguito, quindi, tratteremo i metodi applicabili al segnale audio nel dominio del tempo cioè operando direttamente sulla sua forma d'onda. I risultati che otterremo saranno esemplificati mediante l'uso di procedure specifiche realizzate con MATLAB e applicate a segnali per lo più vocali, anche se i risultati trovati hanno validità del tutto generale.

Bisogna dire fin d'ora che questi metodi (usati per esempio per discriminare il rumore di fondo dal parlato, oppure per distinguere i suoni vocalizzati da quelli non vocalizzati) non danno risultati assolutamente certi sull'informazione che il segnale porta con sé e che sovente vengono usati in combinazione. Il loro vantaggio sta nella facilità di implementazione e nelle modeste capacità di calcolo richieste.

L'ipotesi che ora facciamo è che le proprietà del segnale audio cambino lentamente nel tempo (almeno rispetto al periodo di campionamento); questo ci permette di definire una serie di parametri nel dominio del tempo per cui brevi segmenti di segnale (*frames*) vengono elaborati come se fossero suoni con proprietà costanti all'interno del frame. Se consideriamo per esempio il segnale vocale, questa assunzione si può giustificare con il fatto che nella generazione delle parole contribuiscono sia le corde vocali sia tutte le modificazioni dell'apparato fonatorio (laringe, lingua, bocca) che avvengono con una rapidità non molto elevata tanto da poterle ritenere costanti entro i 100-200ms.

Nel seguito per il calcolo dei parametri useremo alcune sequenze di campioni audio che possono essere importate come vettori in MATLAB direttamente da file audio in formato PCM mono (.WAV) usando i comandi:

```
[s,fs]=wavread('finesunn.wav');

%   s  -->  vettore dei campioni del segnale
```

```

% fS --> frequenza di campionamento

% disegna s
s1=s/max(abs(s)); % normalizza al valore massimo
tempi = (1/fS)*[1:max(size(s1))];
plot(tempi,s1); xlabel('time (s)'); ylabel('s(t)');

```

8.2.0.0.1 Windowing La finestra temporale stabilisce la durata del singolo frame; la sua scelta è un compromesso tra tre fattori: (1) deve essere abbastanza breve in modo che le proprietà del suono non cambino significativamente al suo interno; (2) deve essere abbastanza lunga da poter calcolare il parametro che si vuole stimare (utile anche per ridurre l'effetto di un eventuale rumore sovrapposto al segnale); (3) il susseguirsi delle finestre dovrebbe coprire interamente il segnale (in questo caso il *frame rate* del parametro che andiamo a calcolare deve essere come minimo l'inverso della durata della finestra).

La finestra più semplice è quella rettangolare:

$$r(n) = \begin{cases} 1 & \text{per } 0 \leq n \leq N - 1 \\ 0 & \text{altrimenti} \end{cases} \quad (8.1)$$

Molte applicazioni usano finestre più lunghe del necessario a soddisfare le ipotesi di stazionarietà, cambiandone però la forma per enfatizzare i campioni centrali (figura 8.1); per esempio, se un segnale vocale è approssimativamente stazionario su 10ms, si può usare una finestra da 20ms nella quale i campioni dei 10ms centrali pesano maggiormente rispetto ai primi e ultimi 5ms. La ragione per pesare di più i campioni centrali è relativa all'effetto che la forma della finestra ha sui parametri di uscita. Quando la finestra viene spostata nel tempo per analizzare frames successivi di un segnale, ci possono essere delle grandi oscillazioni dei parametri calcolati se si usa una finestra rettangolare ($r(n)$); per esempio, una semplice misura dell'energia ottenuta sommando il quadrato dei campioni del segnale è soggetta a grandi fluttuazioni non appena la finestra si sposta per includere o escludere, all'inizio o alla fine, campioni con grandi ampiezza. Un'alternativa alla finestra rettangolare (8.1) è la finestra di Hamming:

$$h(n) = \begin{cases} 0.54 - 0.46 \cos\left(\frac{2\pi n}{N-1}\right) & \text{per } 0 \leq n \leq N - 1 \\ 0 & \text{altrimenti} \end{cases} \quad (8.2)$$

Affusolando gli estremi della finestra evito di avere grandi effetti sui parametri anche se ho repentini cambi nel segnale.

Alcuni dei parametri nel dominio del tempo possono essere rappresentati matematicamente nella forma:

$$Q(n) = \sum_{m=-\infty}^{\infty} T[s(m)]w(n-m) = T[s] * w(n) \quad (8.3)$$

dove $T[\cdot]$ è una trasformazione, anche non lineare, pesata da una finestra $w(n)$. Prima di essere elaborato, il segnale può venire eventualmente filtrato per isolare la banda di frequenze desiderata.

M-8.1

Write a MATLAB function for a generic time domain processing.

M-8.1 Solution

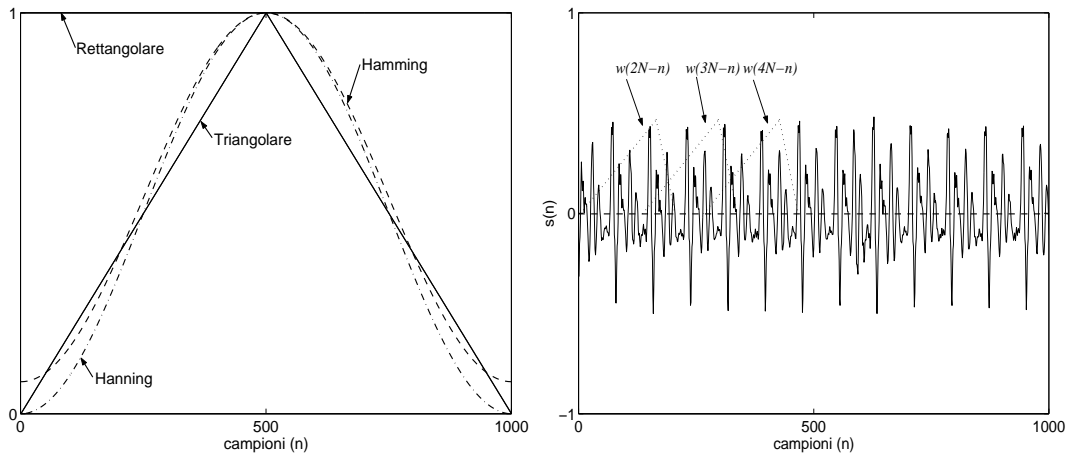


Figura 8.1: a sinistra vari tipi di finestre; a destra tre finestre sovrapposte al segnale $s(n)$, spostate rispetto all'origine di $2N$, $3N$ e $4N$ campioni

```
function [Q,tempi] = st_processing(s,frame,overlap,fs,finestra)
% Q      --> convoluzione di s con la finestra w
% tempi  --> scala dei tempi di Q
%
% s      --> segnale da elaborare
% frame  --> durata in secondi di un frame
% overlap --> percentuale di sovrapposizione dei frames
% fs     --> frequenza di campionamento di s
% finestra --> nome della finestra che si vuole usare

Ns = max(size(s))                % numero di campioni di s
Nframe = floor(fs * frame)       % numero di campioni per frame
Ndiff = floor(Nframe * (1 - overlap/100)) % numero di campioni tra frames
L = floor((Ns-Nframe)/Ndiff);    % numero di finestre

switch lower(finestra)
case 'hamming'
    window = hamming(Nframe);
case 'hanning'
    window = hanning(Nframe);
case 'bartlett'
    window = bartlett(Nframe);
case 'triangolare'
    window = triang(Nframe);
case 'rettangolare'
    window = ones(Nframe,1)/Nframe;
otherwise
    window = ones(Nframe,1)/Nframe;
end;

for n=1:L
    inizio = (n-1) * Ndiff + 1;    % inizio della finestra
```

```

    tempi(n,1) = n* Ndiff/fs;
    Q(n,1) = sum(s(inizio:inizio+Nframe-1,1) .* window);
end;

```

Nell'equazione 8.3, $w(n)$ può essere sia un filtro FIR a risposta finita (per es. la finestra rettangolare o quella di Hamming) che ci permette di ridurre il frame rate risparmiando quindi calcoli, sia un filtro IIR; un esempio di finestra a risposta infinita è

$$w(n) = \begin{cases} a^n & \text{per } n \geq 0 \\ 0 & \text{per } n < 0 \end{cases} \quad (8.4)$$

con $0 < a < 1$; un simile filtro può venire implementato utilizzando una semplice equazione alle differenze, infatti

$$Q(n) = aQ(n-1) + T[s(n)] \quad (8.5)$$

che deve essere calcolato per ogni campione del segnale di ingresso.

8.2.1 Short-Time Average Energy e Magnitude

Per un segnale discreto la *Short-Time Average Energy* è definita come:

$$E(n) = \frac{1}{N} \sum_{i=n-N+1}^n s(i)^2 \quad (8.6)$$

ovvero equivale a $Q(n)$ dell'equazione 8.3 ponendo $T[\cdot] = (\cdot)^2$. Nel caso particolare dell'analisi della voce le sue grandi variazioni temporali in ampiezza tra suoni vocalizzati e non, come pure tra fonemi diversi, permette la segmentazione del parlato nei sistemi automatici di riconoscimento vocale: aiuta per esempio a determinare l'inizio e la fine delle parole isolate (nei sistemi di trasmissione che multiplexano molte conversazioni, delimitare le parole significa evitare di trasmettere le pause).

Un inconveniente della *Short-Time Average Energy* così come l'abbiamo precedentemente definita è la sua sensibilità a grandi ampiezze di segnale (i campioni compaiono elevati al quadrato); un semplice modo per alleviare questo problema è quello di introdurre la *Short-Time Average Magnitude* così definita

$$M(n) = \frac{1}{N} \sum_{i=n-N+1}^n |s(i)| \quad (8.7)$$

equivalente a porre $T[\cdot] = |\cdot|$ nell'equazione 8.3

M-8.2

Write two MATLAB functions to compute Short-Time Average Energy e Magnitude.

M-8.2 Solution

```

Nframe=100;      % numero di campioni per frame
Ns=max(size(s)); % numero di campioni del segnale

for n=1:Ns;      % calcola la Short-Time Average Energy
    E(n,1)=sum(s(max(1,n-Nframe+1):n)).*...

```

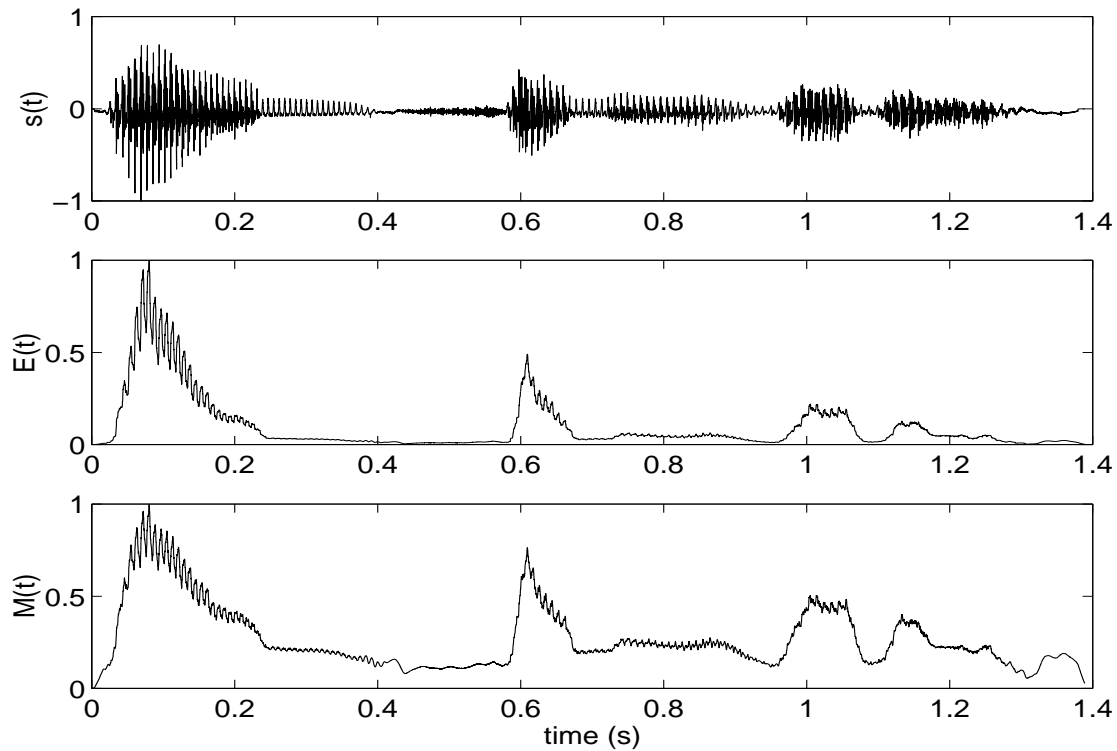


Figura 8.2: In alto l'espressione (/FINE SUNNY WEATHER/) con sotto le corrispondenti *Short-Time Average Energy* e *Short-Time Average Magnitude* normalizzate al valore massimo, calcolate usando finestre rettangolari da $N=100$ campioni e *frame rate* pari alla frequenza di campionamento del segnale (8kHz).

```

s(max(1,n-Nframe+1):n)/Nframe;
end;

for n=1:Ns;      % calcola la Short-Time Average Magnitude
    M(n,1)=sum(abs(s(max(1,n-Nframe+1):n)))/Nframe;
end;

% disegna E(t) e M(t)
E=E/max(E);    % normalizza E(t)
tempi = (1/fS)*[1:max(size(E))]; subplot(2,1,1);
plot(tempi,E); xlabel('time (s)'); ylabel('E(t)');

M=M/max(M);    % normalizza M(t)
tempi = (1/fS)*[1:max(size(M))]; subplot(2,1,2);
plot(tempi,M); xlabel('time (s)'); ylabel('M(t)');

```

La figura 8.2 mostra un esempio di segnale vocale con l'energia corrispondente, calcolata usando l'algoritmo appena definito, mentre la figura 8.3 mostra come la scelta della finestra influenzi la *Short-Time Average Energy* del segnale; nelle figure i grafici sono normalizzati al valore massimo ma

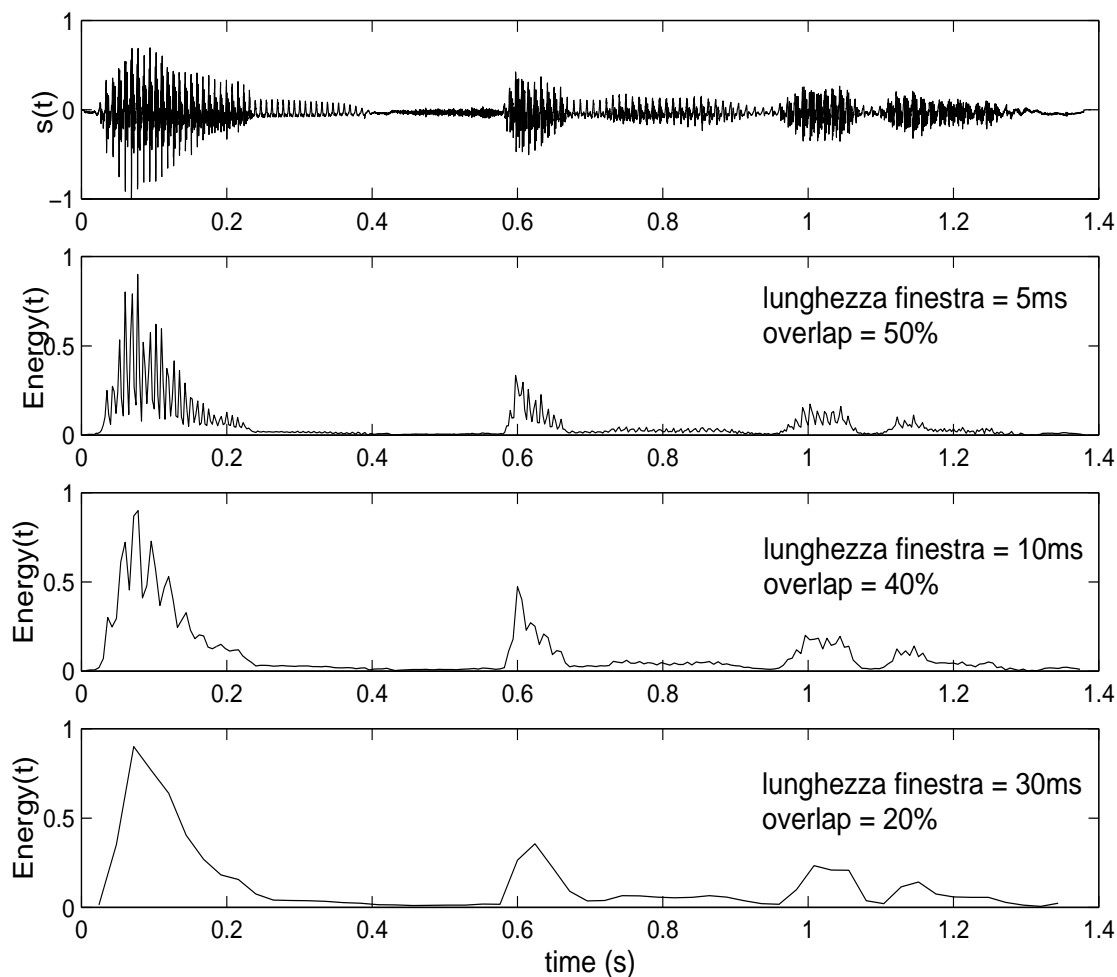


Figura 8.3: In alto l'espressione /FINE SUNNY WEATHER/ con sotto la *Short-Time Average Energy* calcolata con finestre di Hamming di diverse lunghezze, usando la funzione `st_processing(s,frame,overlap,fs,finestra)`; si noti come diventi smussata per finestre più lunghe.

bisogna fare attenzione se si vuole *confrontare* uno stesso parametro applicato a segnali diversi, nel qual caso l'eventuale normalizzazione va fatta rispetto ad un valore comune.

8.2.2 Short-Time Average Zero-Crossing Rate

Normalmente per ottenere informazioni sul contenuto spettrale della voce si ricorre alla trasformata di Fourier; per alcune applicazioni un semplice parametro come la *Zero-Crossing Rate (ZCR)* dà una adeguata informazione spettrale ad un basso costo elaborativo. La *ZCR* corrisponde al numero di passaggi per lo zero del segnale che matematicamente si esprime come il cambiamento di segno di due campioni successivi. Per segnali a banda stretta (es. sinusoidi o la singola uscita di un banco di filtri passa-banda), dalla *ZCR* si ricava la frequenza fondamentale (F_0) del segnale:

$$F_0 = \frac{ZCR * F_S}{2} \quad (8.8)$$

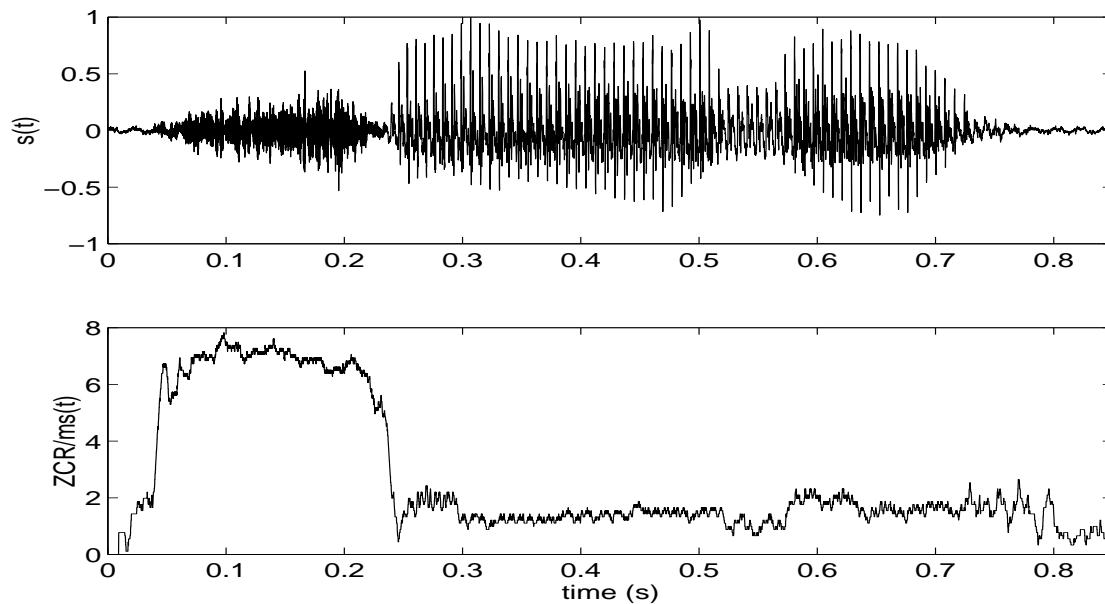


Figura 8.4: *Zero-Crossing Rate* (zero crossing al ms) dell'espressione /SONO/ calcolata con una finestra rettangolare di $N=100$ campioni e frame rate pari a quella del segnale (11kHz); si noti come si riesca a discriminare la /s/ iniziale (suono non vocalizzato) dal resto della parola (suono vocalizzato).

dove F_S è la frequenza di campionamento del segnale e ZCR è espressa in *zero crossing* per campione.

Matematicamente $ZCR = Q(n)$ se nell'equazione 8.3 pongo $T[s(n)] = |\text{sign}(s(n)) - \text{sign}(s(n-1))|/2$ e scalo la finestra $w(n)$ di un fattore $1/N$; ottengo così:

$$Z(n) = \frac{1}{N} \sum_{m=n-N+1}^n \frac{|\text{sign}[s(m)] - \text{sign}[s(m-1)]|}{2} w(n-m) \quad (8.9)$$

dove il segno di $s(n)$ è definito come:

$$\text{sign}(s(n)) = \begin{cases} 1 & \text{per } s(n) \geq 0 \\ -1 & \text{altrimenti} \end{cases} \quad (8.10)$$

M-8.3

Write a MATLAB function for Zero Crossing Rate computation.

M-8.3 Solution

```
Nframe = 100;           % numero di campioni per frame
Ns = max(size(s));

for n = 1+Nframe:Ns; % calcola la Short-Time Average ZCR
    Z(n,1) = sum(abs(sign(s(n-Nframe+1:n))- ...
        sign(s(n-Nframe:n-1))))/2)/Nframe;
end;
```

```

Z=Z*fS/1000;           % Zero-Crossing per ms

% disegna Z(t):
t = (1/fS)*[1:max(size(Z))];
plot(t,Z); xlabel('time (s)'); ylabel('ZCR/ms(t)');

```

Nell'analisi vocale la *ZCR* può aiutare a determinare se il suono è vocalizzato oppure no (vedi figura 8.4); infatti il modello della generazione della voce suggerisce che l'energia della componente vocalizzata è concentrata al di sotto dei 3 kHz mentre quella della componente non vocalizzata si trova a frequenze più alte. Poichè la *ZCR* è in stretto rapporto con la distribuzione frequenziale di energia, ad alte *ZCR* corrispondono suoni non vocalizzati (*unvoiced speech*) mentre a basse *ZCR* suoni vocalizzati (*voiced speech*). Affiancata alla *Short-Time Average Energy* permette di individuare con precisione l'inizio e la fine delle parole soprattutto nei casi di suoni quali /s/ (vedi l'inizio della parola di figura 8.4), /F/, /N/, /M/, /T/, /P/.

M-8.4

Implement a voiced-unvoiced detector as previously explained. Test it on real speech signals. Does it also work for voice - music detection? Why?

A differenza della *Short-Time Average Energy* la *ZCR* è molto sensibile al rumore (per es. quello dei sistemi digitali, degli ADC ma anche dei 60Hz della rete di alimentazione) per cui nel caso di conversione analogico-digitale diventa utile filtrare il segnale con un filtro passa-banda, invece del solo filtro anti-aliasing.

M-8.5

Come nel caso della *Short-Time Average Energy* e della *Short-Time Average Magnitude* anche la *Zero-Crossing Rate* può essere calcolata con una frequenza molto più bassa di quella di campionamento del segnale (vedi figura 8.5). Calcolare *Zero-Crossing Rate* ricorrendo alla funzione *st_processing* vista prima.

M-8.5 Solution

```

Ns = max(size(s));
finestra = 'hamming';
% calcola la Short-Time Average Energy
[E,tE]=st_processing(s.^2,0.012,50,fS,finestra);
% calcola la Short-Time Average Magnitude
[M,tM]=st_processing(abs(s),0.012,50,fS,finestra);
% calcola la Short-Time Average ZCR (ZC per campione)
[Z,tZ]=st_processing([0; 0.5*abs(sign(s(2:Ns))-sign(s(1:Ns-1)))],...
    0.012,50,fS,'rettangolare');

% disegna i segnali
E=E/max(E)*0.8; % normalizza
subplot(3,1,1); plot(tE,E); xlabel('time (s)'); ylabel('Energy(t)');
M=M/max(M)*0.8; % normalizza
subplot(3,1,2); plot(tM,M); xlabel('time (s)'); ylabel('Magnitude(t)');
Z=Z*fS/1000; % ZCR per ms
subplot(3,1,3); plot(tZ,Z); xlabel('time (s)'); ylabel('ZCR/ms(t)');

```

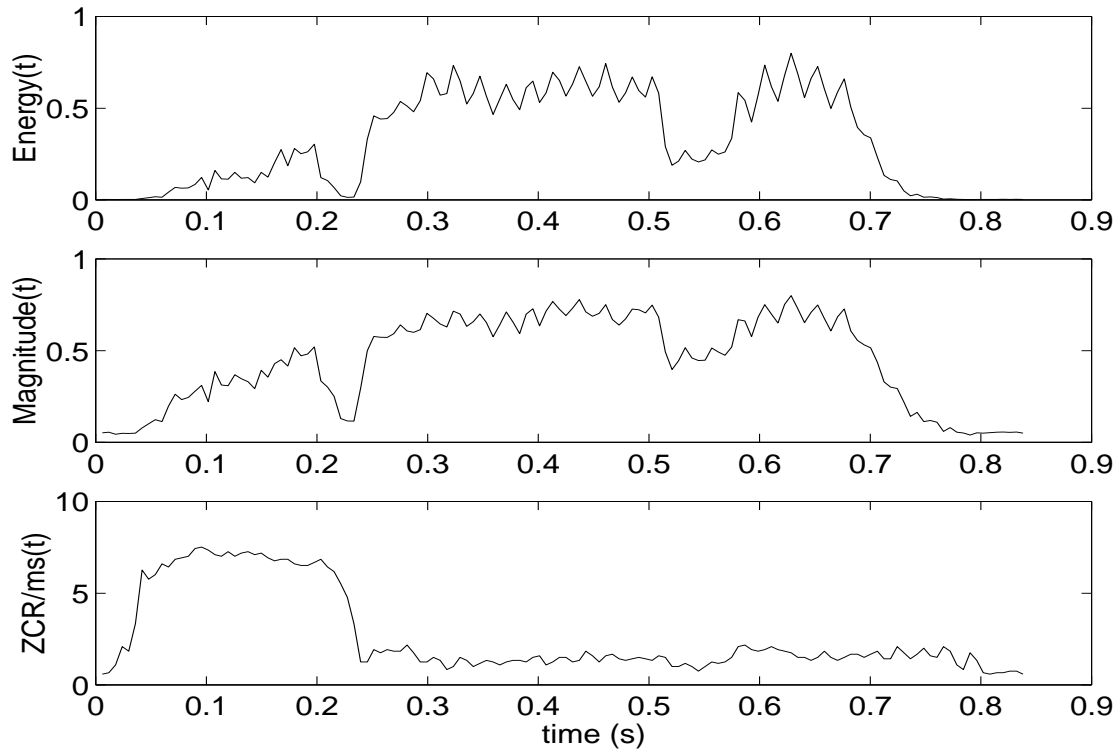


Figura 8.5: *Short-Time Average Energy*, *Short-Time Average Magnitude* e *Zero-Crossing Rate* dell'espressione /SONO/ campionata a 11kHz ed elaborata con frame di 12.5ms e overlap tra i frames del 50%; riducendo l'overlap e allungando le finestre temporali (frame) i parametri nel dominio del tempo perdono in risoluzione temporale ma conservano ugualmente le caratteristiche del segnale (vedere per un confronto la *Zero-Crossing Rate* della figura 8.4).

8.2.3 Short-Time Autocorrelation Function

Il segnale che corrisponde all'anti-trasformata di Fourier della densità spettrale di energia ($C_s(f)$) è l'*autocorrelazione* del segnale; in formule

$$\mathcal{F}[\phi(k)] = C_s(f) = |S(f)|^2 \quad (8.11)$$

Per un segnale discreto è definita come

$$\phi(k) = \sum_{m=-\infty}^{\infty} s(m)s(m+k) \quad (8.12)$$

L'autocorrelazione conserva le informazioni che riguardano le armoniche del segnale, l'ampiezza delle formanti e la loro frequenza. Dall'equazione 8.12 si vede che $\phi(k)$ misura in un certo senso la somiglianza del segnale con la sua versione traslata; avrà quindi valori più grandi in corrispondenza dei ritardi k per cui $s(m)$ e $s(m+k)$ hanno forme d'onda simili. Alcune importanti proprietà di $\phi(k)$ sono le seguenti:

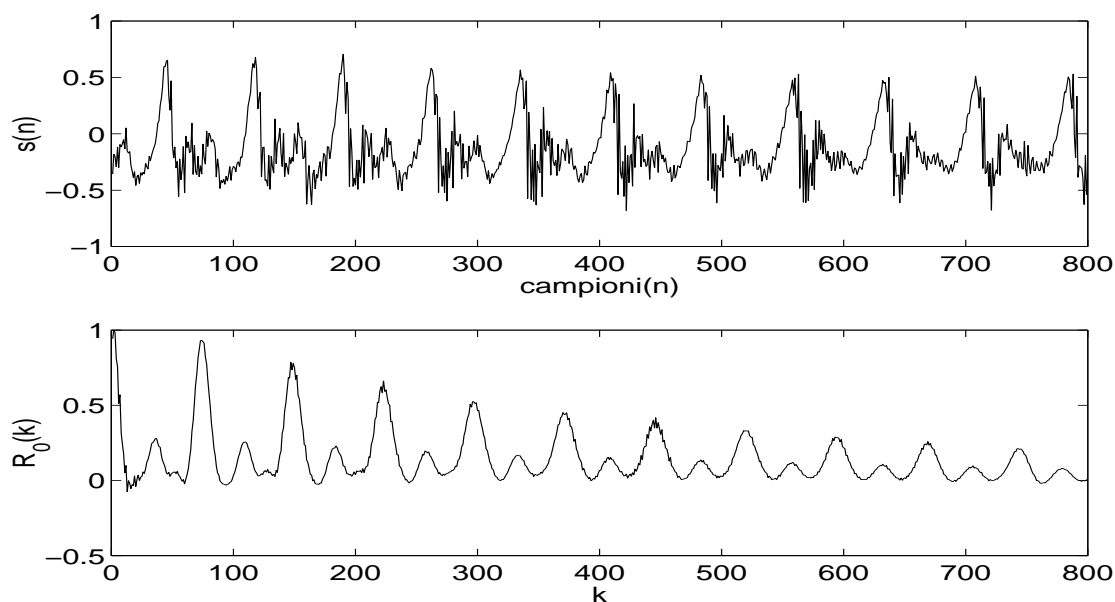


Figura 8.6: Frame da 800 campioni di suono vocalizzato campionato a 8kHz e sua *Short-Time Autocorrelation Function*

1. è una funzione pari: $\phi(k) = \phi(-k)$
2. per $k = 0$ assume il suo massimo valore: $\phi(0) \geq |\phi(k)| \forall k$
3. $\phi(0)$ corrisponde all'energia del segnale (o alla potenza media se i segnali sono periodici o non deterministici)
4. se il segnale è periodico con periodo P anche l'autocorrelazione è periodica con lo stesso periodo: $\phi(k) = \phi(k + P)$ (proprietà importante se si vuole stimare la periodicità del segnale)

La *Short-Time Autocorrelation Function* è ottenuta dall'equazione 8.12 filtrando il segnale con delle opportune finestre temporali $w(n)$:

$$R_n(k) = \sum_{m=-\infty}^{\infty} s(m)w(n-m)s(m+k)w(n-k-m) \quad (8.13)$$

Con un opportuno cambio di variabili l'equazione precedente può essere riscritta nella forma

$$R_n(k) = \sum_{m=-\infty}^{\infty} [s(n+m)w'(m)][s(n+m+k)w'(k+m)] \quad (8.14)$$

dove $w'(n) = w(-n)$; se ora $w'(n)$ ha durata finita N ottengo

$$R_n(k) = \sum_{m=0}^{N-1-k} [s(n+m)w'(m)][s(n+m+k)w'(k+m)] \quad (8.15)$$

M-8.6

Write a MATLAB function for computing the *Short-Time Autocorrelation Function* .

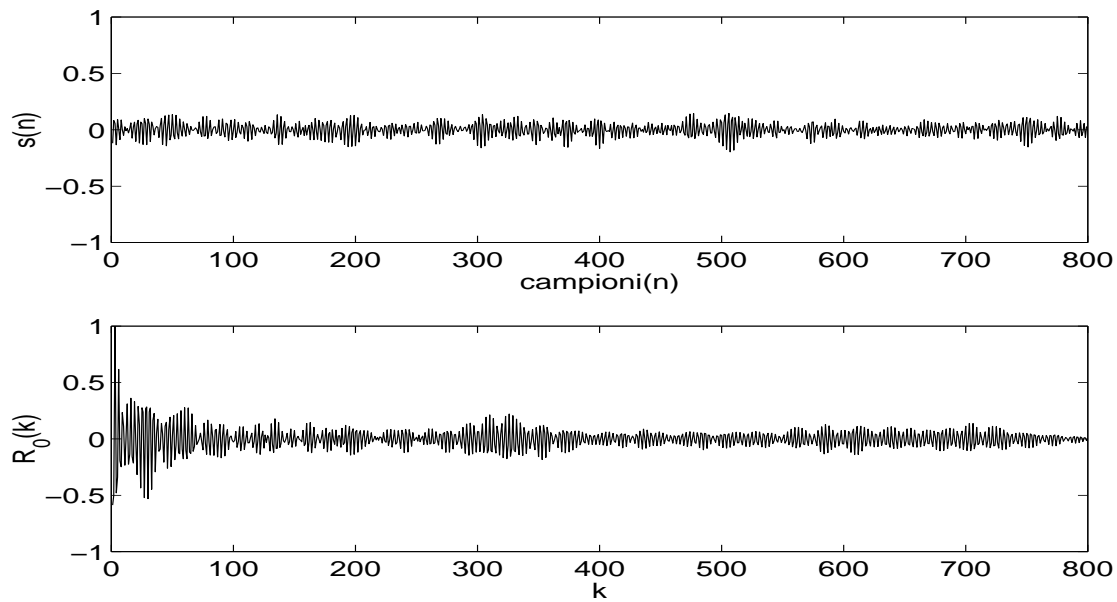


Figura 8.7: Frame di suono non vocalizzato /s/ campionato a 11kHz e sua *Short-Time Autocorrelation Function*; si noti come assomigli ad un rumore e come si differenzi da quella di figura 8.6.

M-8.6 Solution

```

Ns = max(size(s));      % numero di campioni
window = ones(Ns,1);   % finestra rettangolare

s_w = s.*window;
for k = 1:Ns-1;        % calcola la Short-Time Autocorrelation
    R0(k) = sum(s_w(1:Ns-k).* ...
                s_w(k+1:Ns));
end;

% disegna R0(k):
R0=R0/max(abs(R0));    % normalizza R0(k)
plot(1:max(size(R0)),R0); xlabel('k'); ylabel('R_0(k)');

```

La *Short-Time Autocorrelation Function* trova applicazione nell'estrazione del pitch e nella discriminazione tra suono vocalizzato (figura 8.6) e non vocalizzato (figura 8.7). Nella determinazione di F_0 , $R_n(k)$ deve essere calcolata per diversi valori di k prossimi al numero di campioni del periodo di pitch (che dura da un minimo di 3ms per la voce femminile a un massimo di 20ms per quella maschile); se per esempio desidero avere una risoluzione del periodo di pitch di 0.1ms con un segnale campionato a 10kHz devo calcolare $R_n(k)$ per 170 valori di k . In questi casi la finestra $w(n)$ deve avere una durata almeno doppia rispetto al periodo del segnale che si vuole stimare.

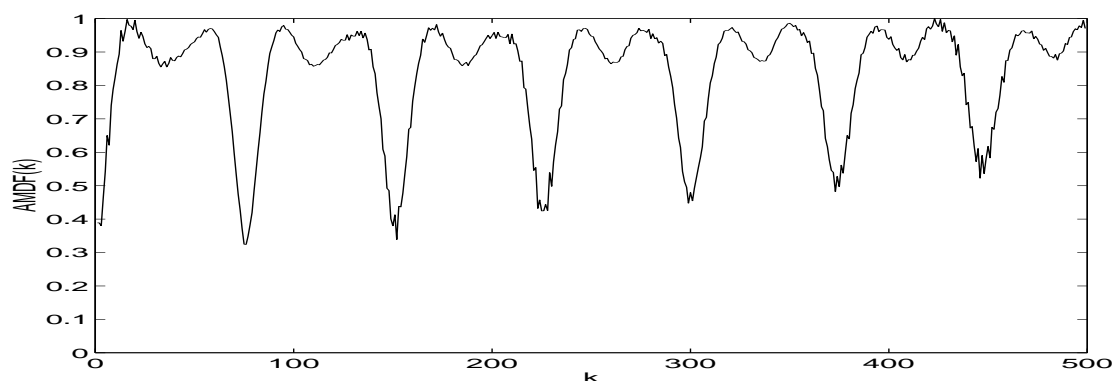


Figura 8.8: *Short-time Average Magnitude Difference Function* del frame di suono vocalizzato di figura 8.6.

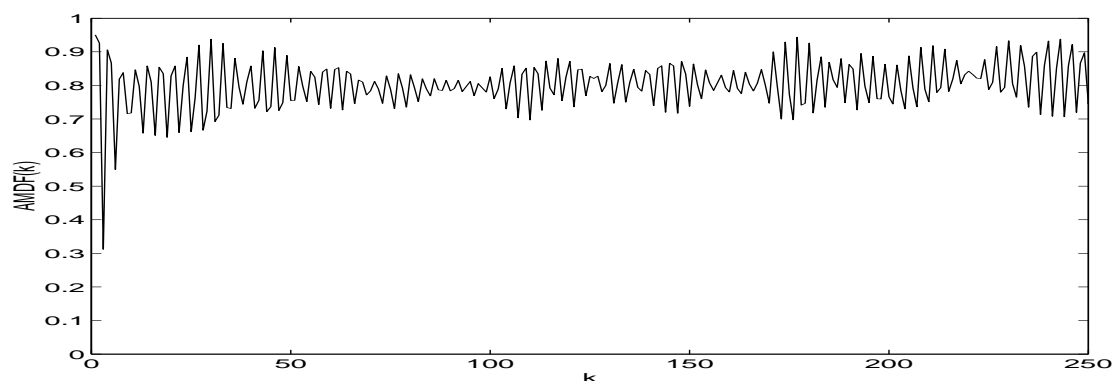


Figura 8.9: *Short-Time AMDF* del frame /S/ di figura 8.7.

8.2.4 Short-Time Average Magnitude Difference Function

Un'alternativa alla *Short-Time Autocorrelation Function* per la stima di F_0 è la *Short-time Average Magnitude Difference Function* (AMDF). Per un segnale periodico di periodo P ho che la successione

$$d(n) = s(n) - s(n - k) \quad (8.16)$$

è uguale a zero per $k = 0, \pm P, \pm 2P, \dots$, quindi invece di moltiplicare $s(m)$ per $s(m - k)$ posso considerare il valore assoluto della loro differenza:

$$\gamma_n(k) = \sum_{m=-\infty}^{\infty} |s(n+m)w(m) - s(n+m-k)w(m-k)| \quad (8.17)$$

dalla quale si può ricavarne una più semplice versione prendendo $w(n)$ rettangolare di durata N :

$$AMDF(k) = \sum_{m=k}^{N-1} |s(m) - s(m-k)| \quad (8.18)$$

M-8.7

Write a MATLAB function for *Short-time Average Magnitude Difference Function* computing.

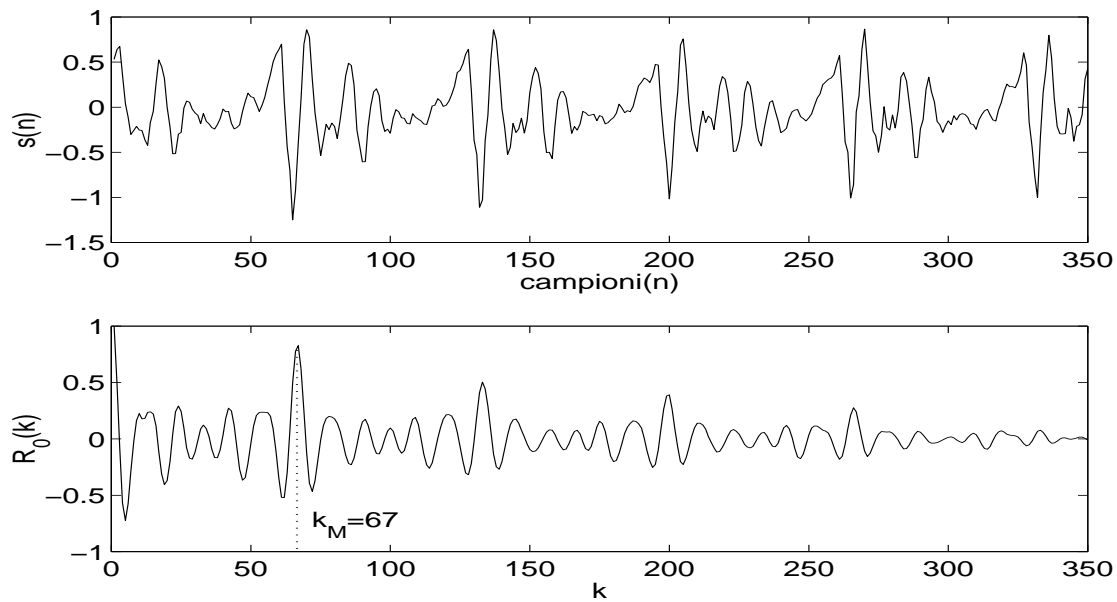


Figura 8.10: Frame del fonema /OH/ (350 valori campionati a 8kHz) e sua *Short-Time Autocorrelation Function*; il secondo massimo è posizionato a $k_M = 67$ da cui si può ricavare la periodicità della frequenza fondamentale del segnale ($F_0 \approx 120\text{Hz}$).

M-8.7 Solution

```

Ns=max(size(s));    % numero di campioni

window=ones(ceil(Ns/2)+1,1);    % finestra rettangolare

for k=1:floor(Ns/2)-1;    % calcola la Short-Time AMDF
    STAMDF(k) = sum(abs(s(floor(Ns/2):Ns).* window - ...
        s(floor(Ns/2)-k:Ns-k).* window));
end;

% disegna STAMDF(t):
STAMDF=STAMDF/max(STAMDF);    % normalizza STAMDF(t)
plot(1:max(size(STAMDF)),STAMDF); xlabel('k'); ylabel('AMDF(k)');

```

Le figure 8.8 e 8.9 mostrano l'andamento tipico della *Short-Time AMDF* per diversi frame: l'informazione che la *Short-Time Autocorrelation Function* dava sulla spaziatura temporale tra i massimi, corrispondente al reciproco della frequenza fondamentale, può essere ricavata ora considerando i minimi della *Short-Time AMDF*. Dal punto di vista computazionale se si usa hardware a virgola fissa il calcolo della *Short-time Average Magnitude Difference Function* è più veloce di quello della *Short-Time Autocorrelation Function*.

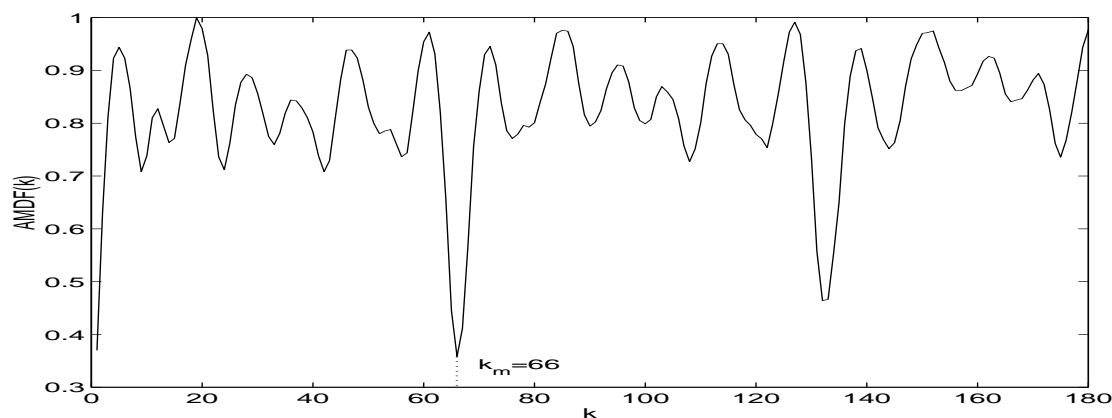


Figura 8.11: *Short-Time AMDF* del frame /OH/ di figura 8.10; qui la periodicità si ricava andando a cercare il secondo minimo ($k_m = 66$).

8.2.5 Stima del *pitch* (F0)

Determinare la frequenza fondamentale (F0) o il *pitch* di un segnale è un problema in molte applicazioni. Il suono vocalizzato viene generato dalla vibrazione delle corde vocali e il *pitch* si riferisce alla frequenza fondamentale di questa vibrazione. Dalla *Short-Time Autocorrelation Function* ricavo l'informazione sulla periodicità del segnale andando a trovare k_M , cioè il primo massimo dopo quello per $k = 0$:

$$F0 = \frac{F_S}{k_M} \quad (8.19)$$

dove F_S è la frequenza di campionamento del segnale (vedi figura 8.10). Utilizzando invece la *Short-Time AMDF* devo considerare il primo minimo dopo quello per $k = 0$ (k_m nella figura 8.11).

Tipicamente uno stimatore del *pitch* effettua tre operazioni:

- pre-processing: filtraggio e semplificazione del segnale attraverso la riduzione dei dati;
- estrazione del periodo;
- post-processing: correzione di eventuali errori.

L'estrazione del periodo mediante l'individuazione dei massimi dell'*autocorrelazione* fa uso di trasformazioni non lineari del segnale (es. center clipping).

M-8.8

Compute the *pitch* with *Short-Time Autocorrelation Function*.

M-8.8 Solution

```

inizio=floor(fs*0.001); % salta il primo massimo
[massimo,kM] = max(R0(inizio:max(size(R0))));
kM=kM + inizio -1;
F0=fs/kM;

```

M-8.9

Compute the pitch with *Short-time Average Magnitude Difference Function* .

M-8.9 Solution

```

inizio=floor(fs*0.001);    % salta il primo minimo
[minimo,km] = min(STAMDF(inizio:max(size(STAMDF))));
km=km + inizio -1;
F0=fs/km;

```

Bisogna tenere presente che talvolta il terzo massimo ha ampiezza maggiore del secondo nel qual caso, con le funzioni appena definite, sbagliremmo la stima (che sarebbe quella di un'armonica della frequenza fondamentale); per questo motivo spesso si affiancano altri metodi per evitare errori grossolani.

8.3 Stima dell'involuppo spettrale

L'involuppo spettrale è considerato un elemento molto significativo nella caratterizzazione dei suoni, specie nella voce. Esse infatti sono caratterizzate da uno spettro armonico, cui è sovrapposto un involuppo. Le zone in frequenza, in cui si concentra l'energia, sono in corrispondenza con le principali risonanze del tratto vocale (percorso del suono dalle corde vocali fino all'esterno della bocca). Queste risonanze, chiamate formanti, sono peculiari per la differenziazione e il riconoscimento delle vocali stesse (vedi fig. 8.12). Anche le varie famiglie di strumenti musicali sono spesso distinte tra loro da tipici involuppi spettrali. Spesso, nelle trasformazioni del suono, si parla, anche se impropriamente, di cambiamento dell'altezza (*pitch shifting*) con preservazione del timbro, quando si preserva l'involuppo spettrale.

8.3.1 Stima dell'involuppo spettrale mediante banco di filtri

Una prima maniera consiste nel fare una approssimazione, mediante segmenti, dello spettro in ampiezza. Si selezionano i massimi e si congiungono con linee rette, oppure si possono prendere punti equispaziati sull'asse delle frequenze e si congiungono con segmenti. Le ascisse (frequenze) dei punti possono anche essere scelte su una scala logaritmica o altra spaziatura percettivamente significativa. Questo metodo è abbastanza flessibile, ma non molto preciso. Questi punti si possono ottenere mediante l'uso di un banco di filtri passabanda equispaziati (a banda costante), o distribuiti logaritmicamente sull'asse delle frequenze (cosidetti filtri a Q costante, dove Q è il rapporto tra la larghezza di banda, e la frequenza centrale del filtro). Un esempio sono i cosidetti filtri di ottava o di terza. In alcuni casi essi sono progettati per riprodurre il comportamento della coclea. dal punto di vista computazionale i filtri possono essere realizzati mediante FFT, calcolando prima lo spettro (in modulo) e poi sommando i contributi di ciascun bin frequenziale pesato dalla risposta in frequenza del r -esimo filtro. Se i filtri sono passabanda rettangolari, basta sommare i contributi dei bin appartenenti alla banda r -esima. Risulta cioè che l'energia $E_r(j)$ per il canale r -esimo del j -esimo frame è data da

$$E_r(j) = \frac{1}{N} \sum_{k \in B_r} |X_j(k)|^2$$

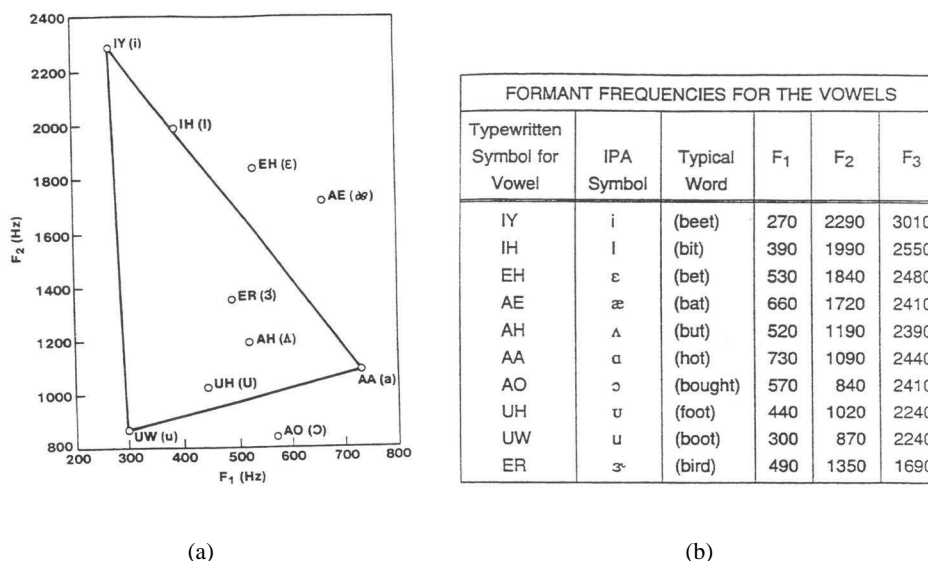


Figura 8.12: rappresentazione dei baricentri delle vocali nel piano descritto dalla frequenza F_1 e F_2 delle prime due formanti. Esse formano approssimativamente un triangolo (a). Frequenza delle prime formanti delle vocali inglesi (b).

dove B_r indica l'insieme delle componenti spettrali appartenenti al filtro r -esimo e N la dimensione della FFT. Talvolta il risultato è normalizzato (diviso) per la larghezza di banda; altre volte no. Questo dipende dall'uso che poi ne viene fatto.

M-8.10

Write a MATLAB function for the spectral envelope computing, with the filterbank approach. Try a filterbank of frequency linearly spaced filters and logarithmic spaced filters (e.g. third octave filters).

M-8.11

Write a MATLAB function for the spectral envelope computing, with the gamma tone filterbank approach. Look in the literature or on the web for gammatone filter definition. gamma tone filters simulate the behaviour of the cochlea.

8.3.2 Stima dell'inviluppo spettrale mediante predizione lineare (LPC)

Un'altro metodo consiste nel fare una approssimazione mediante predizione lineare (LPC) come visto nel capitolo sulla sintesi. In questo caso si stima un filtro a soli poli che approssima lo spettro. Quando l'ordine del filtro è basso, viene solo seguito l'inviluppo spettrale, trascurando la struttura fine dello spettro prodotta dalle periodicità del suono. Nella sezione 8.3.2.1 sono riportati alcuni esempi di analisi mediante predizione lineare (LPC).

Di seguito sono riportati due metodi non lineari particolarmente efficaci, consistenti nel cosiddetto cepstrum e nella sua variante mel-cepstrum. Quest'ultimo metodo fornisce la parametrizzazione dell'inviluppo spettrale più efficace nel riconoscimento del parlato e degli strumenti musicali.

M-8.12

Write a MATLAB function for the spectral envelope computing, with the LPC approach. Experiment different filter lengths p and compare with the original signal spectrum. Apply your function to different kinds of sounds: musicals, speech and environmental noises.

M-8.13

In LPC analysis, the position of formants (resonances) is related to the poles of the estimated transfer function. Factorize the denominator of the transfer function and estimate the frequency of the formants. Note that if θ_k is the argument of z_k complex conjugate zero of the denominator, then its corresponding resonant frequency f_k derives from $\theta_k = 2\pi f_k / F_s$; the formant bandwidth B_k is related to the zero modulus by $|z_k| = \exp(-\pi B / F_s)$.

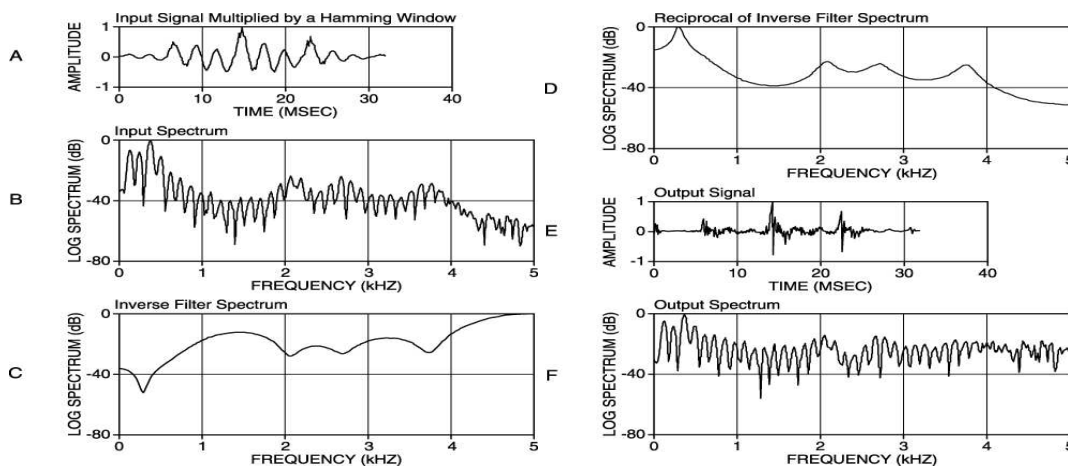
8.3.2.1 Esempi di analisi mediante predizione lineare (LPC)

Figura 8.13: Analisi LPC della vocale /i/

Nella fig. 8.13 è riportata l'analisi della vocale /i/ mediante predizione lineare (LPC), [tratto da J. D. Markel, Formant Trajectory Estimation from a Linear Least Squares Inverse Filter Formulation, Speech Communications Research Lab Monograph #7, Oct. 1971.] In fig. 8.13 (a) il frame del segnale cui è stata applicata la finestra di Hamming; (b) lo spettro del segnale; (c) lo spettro del filtro inverso; (d) l'involuppo spettrale stimato come reciproco dello spettro del filtro inverso; (e) il residuo (eccitazione); (f) spettro del residuo. La frequenza di campionamento è 10 kHz; lunghezza del frame $N = 320$ (32 ms); ordine del filtro LPC $p = 14$. Nella fig. 8.14 è riportata l'analisi mediante predizione lineare (LPC) di un suono non vocalizzato (non periodico) costituito dalla consonante fricativa /s/. Infine nella fig. 8.15 è riportata l'analisi della vocale /ae/ al variare dell'ordine del filtro LPC di predizione lineare: (a) il frame del segnale cui è stata applicata la finestra di Hamming; (b) lo spettro del segnale; (c-f) l'involuppo stimato con p che varia da 6 a 18. La frequenza di campionamento è 6 kHz.

8.3.3 Stima dell'involuppo spettrale mediante cepstrum

Il metodo del cepstrum consente la separazione di un segnale $y(n) = x(n) * h(n)$, basato sul modello sorgente-filtro, in cui la sorgente $x(n)$ passa attraverso un filtro descritto dalla risposta all'impulso

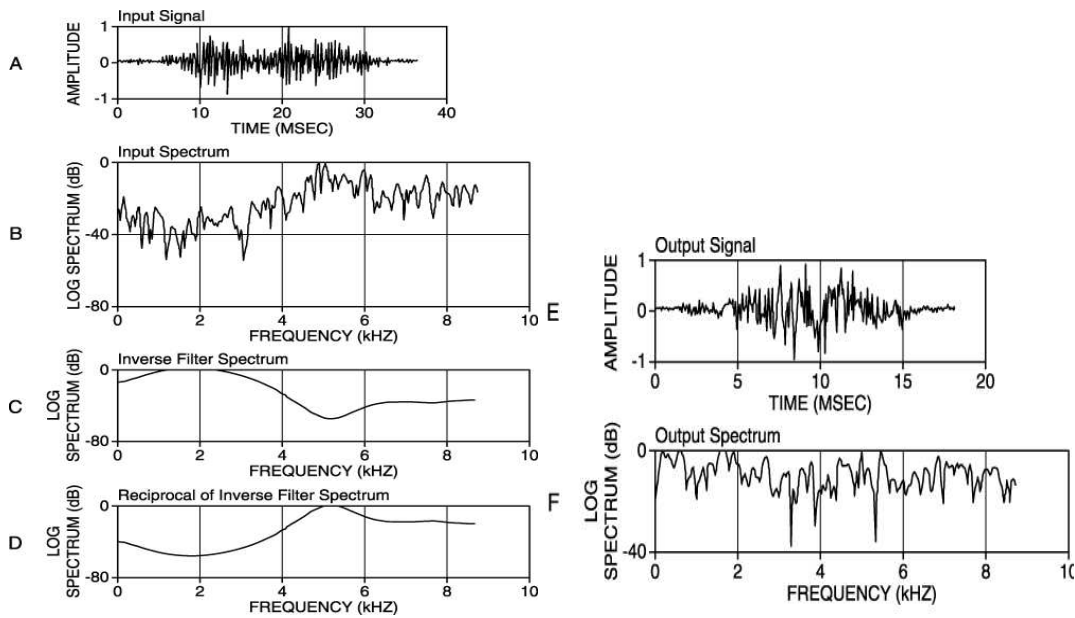


Figura 8.14: Analisi LPC della fricativa /s/

$h(n)$. Lo spettro del segnale $y(n)$ risulta $Y(k) = X(k) \cdot H(k)$, che è il prodotto di due spettri; k è l'indice per le frequenze discrete. Il primo corrisponde allo spettro della sorgente e il secondo a quello del filtro. E' difficile separare questi due spettri; più facile è separare l'involuppo (reale) del filtro dal resto dello spettro, attribuendo tutta la fase alla sorgente. L'idea del cespstrum si basa sulla proprietà del logaritmo $\log(a \cdot b) = \log(a) + \log(b)$.

Se prendiamo il logaritmo del modulo dello spettro $Y(k)$, otteniamo

$$\log |Y(k)| = \log(|X(k) \cdot H(k)|) = \log |X(k)| + \log |H(k)|$$

Se consideriamo il grafico di $\log |Y(k)|$ come un segnale nel tempo (dimenticandoci temporaneamente che in realtà è in frequenza), si possono distinguere due componenti: una oscillazione veloce, dovuta alla struttura armonica (righe) dell'eccitazione, e un andamento più lento corrispondente alle risonanze del filtro (involuppo spettrale). Si può quindi ottenere una separazione delle due componenti, cioè componente a variazione veloce e quella a variazione lenta, mediante rispettivamente un filtro passa alto e passa-basso, del segnale $\log |Y(k)|$ sempre interpretato come segnale nel tempo, vedi fig. 8.16 (sopra).

Un metodo per separare le due componenti, consiste nell'usare la trasformata (nel nostro caso inversa) di Fourier. Pertanto

$$\text{DFT}^{-1}(\log |Y(k)|) = \text{DFT}^{-1}(\log |X(k)|) + \text{DFT}^{-1}(\log |H(k)|)$$

La parte di $\text{DFT}^{-1}(\log |Y(k)|)$ verso l'origine descrive l'involuppo spettrale, quella distante l'eccitazione. In particolare si noterà una specie di riga in corrispondenza della periodicità del $\log |Y(k)|$ e quindi del periodo del suono, vedi fig. 8.16 (sotto). A questo punto si può capire il gioco di parole che sta alla base del nome cespstrum. Infatti la parola ceps-trum corrisponde a spec-trum con la prima parte letta all'inverso. Analogamente si chiamiamo quefrequency la ascissa di $\text{DFT}^{-1}(\log |Y(k)|)$ invece che frequency. Normalmente infatti la DFT^{-1} produce un segnale nel tempo, ma qui invece va

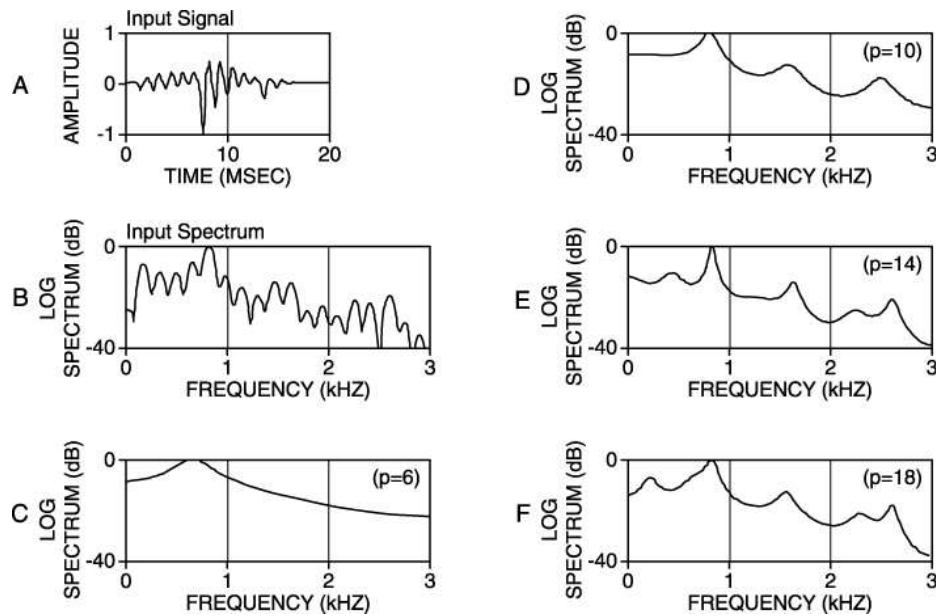


Figura 8.15: Analisi LPC della vocale /ae/ al variare dell'ordine p del filtro di predizione lineare.

interpretato come frequenza. In definitiva il cepstrum (reale) è dato da

$$c(n) = \text{DFT}^{-1}(\log |Y(k)|) \quad (8.20)$$

Si noti che il cepstrum è una funzione pari dell'indice n , in quanto $\log |Y(k)|$ è una funzione reale e pari. Questa proprietà consente di usare la trasformata coseno inversa per ottenere $c(n)$ da $\log |Y(k)|$. L'indice n di $c(n)$ è chiamato quefrequency, dove ad alta quefrequency (variazioni rapide nello spettro in dB) corrispondono valori di n grandi e viceversa. Pertanto si può assumere che i valori bassi della quefrequency descrivano l'involuppo spettrale, mentre quelli alti corrispondono all'eccitazione o sorgente.

La separazione è ottenuta moltiplicando il cepstrum per una finestra passa basso $w_{LP}(n)$ nel dominio del cepstrum. Ad es.

$$w_{LP}(n) = \begin{cases} 1 & |n| < n_c \\ 0.5 & |n| = n_c \\ 0 & |n| > n_c \end{cases}$$

dove n_c è una opportuna soglia. Ad esempio per i suoni vocalizzati possiamo considerare che il formante più basso F_1 di un maschio adulto sia circa $F_1 = 270$ Hz. Le oscillazioni dello spettro corrispondenti all'involuppo non devono avere componenti sopra la quefrequency $q_p = 3,7 \text{ ms} = 1/270$ Hz. In definitiva per suoni periodici, $n_c < n_p$, con n_p periodo in campioni. Per una frequenza di campionamento $f_S = 44.1$ kHz, risulta $n_p = f_S q_p = f_S / F_1 = 163$ campioni. In pratica verrà scelto come soglia un valore leggermente inferiore. Si noti che per la voce femminile la separazione è più difficile. Infatti l'altezza media della voce femminile è di circa 256 Hz, mentre il formante più basso è a 310 Hz. Questi valori sono piuttosto vicini, e quindi meno facilmente separabili.

L'involuppo spettrale, in una scala proporzionale ai decibel, è dato da

$$\log H(k) = \text{DFT}[w_{LP} \cdot c(n)] = \text{DFT}[w_{LP}(n) \cdot \text{DFT}^{-1}(\log |Y(k)|)] \quad (8.21)$$

In fig. 8.17 sono riportati esempi di analisi cepstrale per suoni vocalizzati e non vocalizzati, tratti da [Schafer and Rabiner, System for Automatic Formant Analysis of Voiced Speech, JASA, vol. 47, 1970, p. 634].

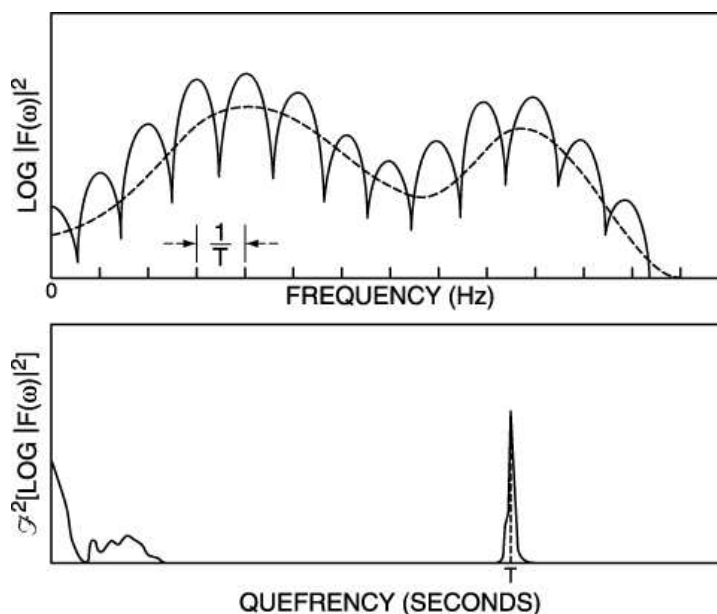


Figura 8.16: Esempio di cepstrum: sopra è rappresentato $\log |Y(k)|^2$; sotto il corrispondente cepstrum $c(n) = \text{DFT}^{-1}(\log |Y(k)|)$

M-8.14

Write a MATLAB function for the spectral envelope computing, with the cepstral approach and experiment it for different kinds of sounds. Develop a pitch estimate function based on cepstral approach.

Si noti inoltre che i massimi dell'inviluppo spettrale corrispondono alle risonanze (formanti) molto caratteristiche per differenziare le varie vocali. Essi possono quindi essere individuati dall'inviluppo spettrale, come mostrato in fig. 8.18.

M-8.15

Estimate the formants of a voice in a song and plot their position on the spectrogram.

8.3.4 Analisi mediante mel-cepstrum

Studi di psicoacustica hanno mostrato che la percezione umana del contenuto frequenziale del suono non segue una scala lineare, ma all'incirca logaritmica. Infatti per ogni tono di f , misurato in Hz, corrisponde una altezza soggettiva misurata su una scala chiamata scala *mel*. Come riferimento della scala *mel*, si ha che 1000 Hz corrispondono a 1000 *mel*. Si usa una trasformazione non lineare della scala della frequenza per ottenere il corrispondente valore in *mel* (fig. 8.19), data da

$$\text{mel}(f) = \begin{cases} f & \text{if } f \leq 1 \text{ kHz} \\ 2595 \log_{10} \left(1 + \frac{f}{700} \right) & \text{if } f > 1 \text{ kHz} \end{cases}$$

Per applicare la scala *mel* al cepstrum, si usa un banco di filtri triangolari passabanda con frequenza centrale in K valori equispaziati in *mel*, vedi fig. 8.20. La larghezza di banda di ciascun filtro è la distanza dalla frequenza centrale del filtro precedente, moltiplicata per due. Il primo filtro

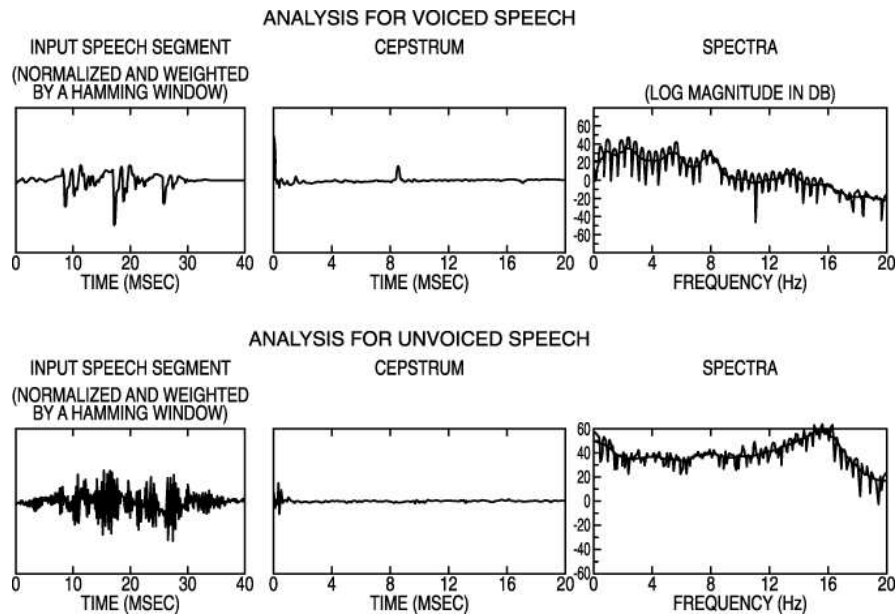


Figura 8.17: Esempio di analisi cepstrale per suoni vocalizzati e non vocalizzati

parte da 0. Pertanto la larghezza di banda dei filtri sotto 1000 Hz sarà di 200 Hz; poi essa crescerà esponenzialmente. Pertanto i filtri saranno a banda costante fino a 1000Hz, a Q costante sopra.

Il mel-cepstrum vuole stimare l'involuppo spettrale dell'uscita di questo banco di filtri. Sia quindi Y_n il logaritmo dell'energia in uscita dal canale n , attraverso la trasformata coseno discreta (DCT) ottengo i coefficienti mel-cepstrali MFCC (mel frequency cepstral coefficient) mediante l'equazione

$$c_k = \sum_{n=1}^N Y_n \cos \left[k \left(n - \frac{1}{2} \right) \frac{\pi}{N} \right] \quad k = 0, \dots, K$$

Si ricostruisce un involuppo spettrale semplificato usando i primi K_m coefficienti, con $K_m < K$, analogamente a quanto visto per la stima dell'involuppo con il cepstrum

$$\tilde{C}(mel) = \sum_{k=1}^{K_m} c_k \cos(2\pi k \frac{mel}{B_m})$$

dove B_m è la larghezza della banda analizzata, espressa in mel. Un tipico valore di K_m usato la caratterizzazione e classificazione della musica è $K_m = 20$. Si noti che il coefficiente c_0 è il valore medio dei valori (in dB) dell'energia dei canali del banco di filtri. Pertanto esso è in diretta relazione con l'energia del suono. Esso può servire per la stima dell'energia. Inoltre normalmente viene trascurato, quando si vuole fare un confronto della forma dell'involuppo, normalizzato in energia, di vari suoni, ad esempio nei problemi di riconoscimento.

M-8.16

Write a MATLAB function for the spectral envelope computing, with the mel-cepstral approach and experiment it for different kinds of sounds. Compare the results obtained with the different spectral envelope algorithms.

In fig. 8.21 è mostrato un esempio di analisi con mel-cepstrum. In essa sono confrontati spettri in decibel, rappresentati su una scala frequenziale logaritmica. Nel primo quadrante è rappresentato

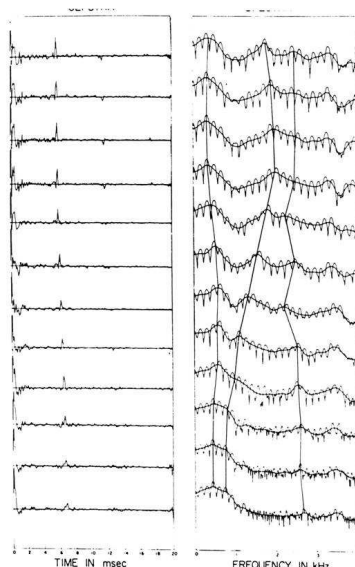


Fig. 7.15 Automatic formant estimation from cepstrally smoothed log spectra. (After Schafer and Rabiner [11].)

Figura 8.18: Automatic formant estimation from cepstrally smoothed log Spectra [from Schaefer Rabiner].

lo spettro del suono. Nel secondo (in alto a destra) l'involuppo ottenuto mediante mel-cesprum. Nel terzo (in basso a sinistra) l'involuppo ottenuto mediante predizione lineare. Infine nell'ultimo grafico, lo spettro ottenuto usando tutti i coefficienti spettrali

8.4 Attributi a medio livello ricavabili dall'analisi spettrale

Dall'analisi spettrale del suono vengono ricavati dei parametri che danno una descrizione delle caratteristiche fisiche a basso livello del suono. In particolare si ricava l'ampiezza, fase e frequenza istantanee di ogni parziale. Nel modello sinusoidale più rumore si può anche ricavare una descrizione delle caratteristiche spettrali del rumore. A partire da questi parametri a basso livello, è possibile ricavare una descrizione ad un livello di astrazione più alto, che possa servire sia per una eventuale trasformazione del suono stesso nella resintesi, sia per riconoscere la sorgente o per ricavare altre informazioni da essa trasmesse.

Nel caso questi parametri vengano poi usati per la trasformazione, è bene che siano di interpretazione intuitiva e che facciano riferimento, in qualche modo, agli attributi percettivi del suono. In altri casi si ricavano dallo spettro ad esempio il grado di armonicità, la rumorosità, la brillantezza; questi attributi descrivono le caratteristiche del suono e sono utili nei problemi di riconoscimento.

Oltre agli attributi istantanei del suono, spesso sono utili le loro derivate. La derivata prima descrive la tendenza dell'evoluzione temporale, in quell'istante; talvolta viene presa in considerazione anche la derivata seconda, che descrive l'accelerazione dell'evoluzione temporale, che risulta spesso più in relazione con le scelte volontarie. Nei segnali discreti, la derivata viene sostituita dal calcolo della differenza tra il valore corrente ed il precedente $d(n) = p(n) - p(n - 1)$. Spesso però la stima della derivata così ottenuta è abbastanza rumorosa ed è opportuno smussarla. A questo scopo per ogni istante si calcola la parabola che approssima, ai minimi quadrati, il valore del parametro in tre punti adiacenti. Si usa quindi la derivata (prima o seconda) della parabola nel punto centrale. Dall'analisi di

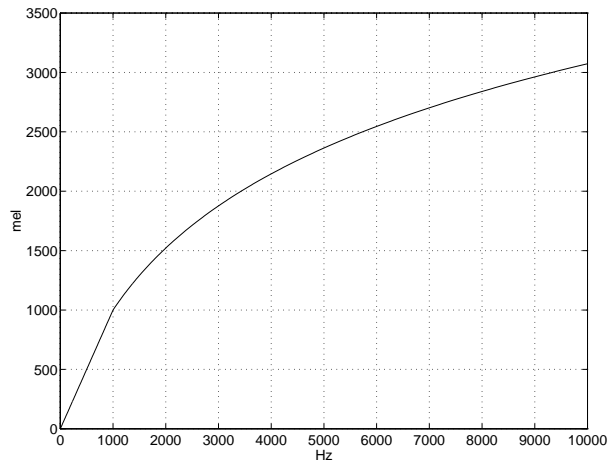


Figura 8.19: Trasformazione da Hz a mel

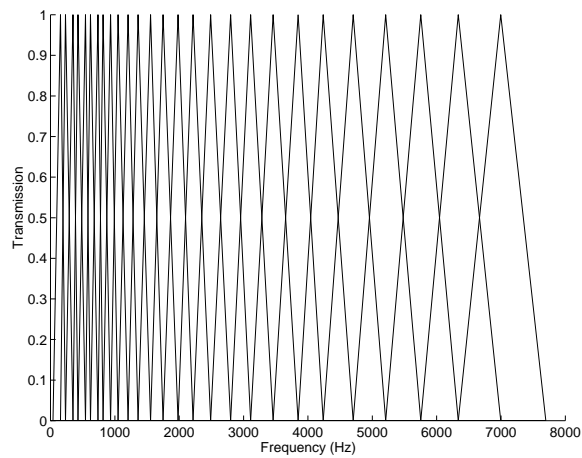


Figura 8.20: Banco di filtri su scala mel

questi parametri e della loro evoluzione, viene aiutata la segmentazione del suono, cioè la separazione di regioni temporali con andamento omogeneo.

M-8.17

Plot the time evolution of the first mel-cepstral coefficient of the analysis of a music fragment. Is it useful for detecting the transients and separating the notes?

M-8.18

Compute the first K_m mel cepstrum coefficients of a set of notes of different instruments of the same pitch. Are there similarities among instruments of the same kind? How they vary with the different dynamics (loudness) of the notes? Repeat the experiment with the sound of the same instrument played at different pitches.

8.4.1 Attributi a basso livello

Descriviamo ora più in dettaglio gli attributi più significativi che si ottengono dalla rappresentazione sinusoidale più rumore (o meglio residuo). Sia il suono $x(n)$ scomposto nelle due componenti $x_S(n)$

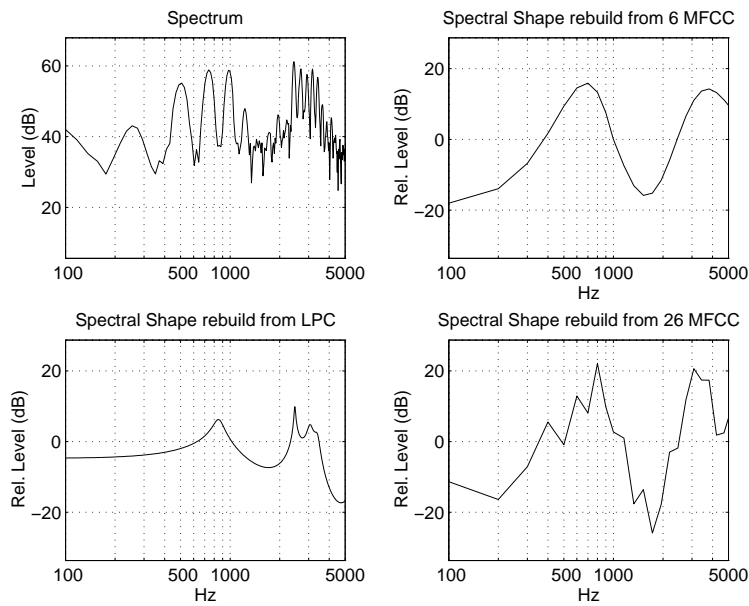


Figura 8.21: Esempio di analisi del suono di un clarinetto con mel-cespstrum.

sinusoidale e $x_R(n)$ residuo: $x(n) = x_S(n) + x_R(n)$, con $x_S(n) = \sum_{i=1}^I a_i \cos[n2\pi f_i(n)/F_S + \phi_i(n)]$. In questa espressione a_i rappresenta l'ampiezza, in scala lineare, della i -esima armonica e f_i la sua frequenza. Essi sono parametri che variano nel tempo. Gli attributi qui descritti si intendono ricavati dall'analisi di un frame e vengono in genere riferiti temporalmente al centro del frame stesso, oppure alla fine quando l'analisi viene fatta in tempo reale. Essi sono quindi attributi istantanei del suono in esame. I più usati sono:

- ampiezza totale della componente sinusoidale risultante dalla somma di tutte le parziali di un frame espresse in decibel

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

dove a_i è l'ampiezza della parziale i -esima;

- ampiezza della componente residuo, ottenuta dalla somma dei valori assoluti del residuo nel frame

$$AR_{tot} = 20 \log_{10} \left(\sum_{n=0}^{M-1} |x_R(n)| \right) = 20 \log_{10} \left(\sum_{k=0}^{N-1} |X_R(k)| \right)$$

- ampiezza totale del suono

$$\begin{aligned} A_{tot} &= 20 \log_{10} \left(\sum_{n=0}^{M-1} |x(n)| \right) = 20 \log_{10} \left(\sum_{k=0}^{N-1} |X(k)| \right) \\ &= 20 \log_{10} \left(\sum_{i=1}^I a_i + \sum_{k=0}^{N-1} |X_R(k)| \right) \end{aligned}$$

- peso dell'armonica i -esima rispetto al totale della componente sinusoidale

$$w_i = \frac{a_i}{\sum_{i=1}^I a_i}$$

- la frequenza fondamentale (pitch), che può essere ottenuta come media pesata delle frequenze normalizzate di tutte le armoniche,

$$F0 = \sum_{i=1}^I \frac{f_i}{i} \cdot w_i$$

In un suono esattamente periodico, tutte le parziali sono multiple della fondamentale. Cioè vale $f_i = iF0$. Nei suoni reali questo vale solo approssimativamente e pertanto la fondamentale deve essere stimata con espressioni del tipo di quella sopra indicata.

8.4.2 Attributi a livello superiore

Vengono ora presentati attributi a più alto livello che descrivono le caratteristiche spettrali del suono. Questi attributi sono anch'essi ricavati a livello di frame e sono quindi considerati istantanei, nel senso precisato sopra. I più usati sono:

- disarmonicità

$$HD = \sum_{i=1}^I |f_i - iF_0| \cdot w_i$$

- rumorosità (noisiness) è il rapporto tra l'energia della parte rumorosa e l'energia totale

$$Noisiness = \frac{\sum_{n=0}^{M-1} |x_R(n)|}{\sum_{n=0}^{M-1} |x(n)|}$$

In Mpeg7 si chiama AudioHarmonicity ed è un descrittore del segnale.

- brillantezza, determinata come il baricentro dello spettro

$$BR = \frac{\sum_{k=0}^{N-1} k |X(k)|}{\sum_{k=0}^{N-1} |X(k)|} \cdot \frac{F_S}{N}$$

Nel caso di suono armonici, dotati cioè di altezza, si definisce anche la brillantezza in relazione alla fondamentale $F0$ come

$$BR_{F0} = \frac{\sum_{i=1}^I i a_i}{\sum_{i=1}^I a_i} = \sum_{i=1}^I i w_i$$

- pendenza spettrale, ottenuta dalla regressione lineare sui punti (f_i, a_i)

$$Stilt = \frac{1}{\sum_{i=1}^I t_i^2} \cdot \sum_{i=1}^I \frac{t_i a_i}{w_i}$$

dove

$$t_i = \frac{1}{w_i} \left(f_i - \frac{\sum_{i=1}^I f_i / w_i^2}{\sum_{i=1}^I 1 / w_i^2} \right)$$

- deviazione spettrale delle armoniche (Harmonic Spectral Deviation)

$$HDEV = \frac{1}{I} \sum_{i=1}^I [a_i - spec_env(f_i)]$$

dove $spec_env(f_i)$ è l'involuppo spettrale stimato con un dei metodi visti sopra, valutato alla frequenza f_i della i -esima armonica. Questo parametro fa parte dei descrittori spettrali del timbro in Mpeg7.

- rapporto l'energia delle armoniche dispari e pari

$$OER = \frac{\sum_{i=pari} a_i^2}{\sum_{i=dispari} a_i^2}$$

Questo parametro è utile per distinguere i suoni tipo clarinetti, che hanno poca energia nelle armoniche pari, comportandosi come un tubo chiuso ad una estremità, da quelli tipo tromba, che hanno energia simile nei due tipi di armoniche.

- tristimulus. Questi parametri sono stati pensati per pesare differemente le armoniche nelle varie zone: fondamentale ($T1$), dalla seconda alla quarta ($T2$), le rimanenti ($T3$). E' definito da

$$T1 = \frac{a_1}{\sum_i a_i} \quad T2 = \frac{a_2 + a_3 + a_4}{\sum_i a_i} \quad T3 = \frac{\sum_{i=5}^I a_i}{\sum_i a_i} = 1 - T1 - T2$$

In fig. 8.22(a) è riportata la tipica rappresentazione del tristimulus dove nellasse x è indicato

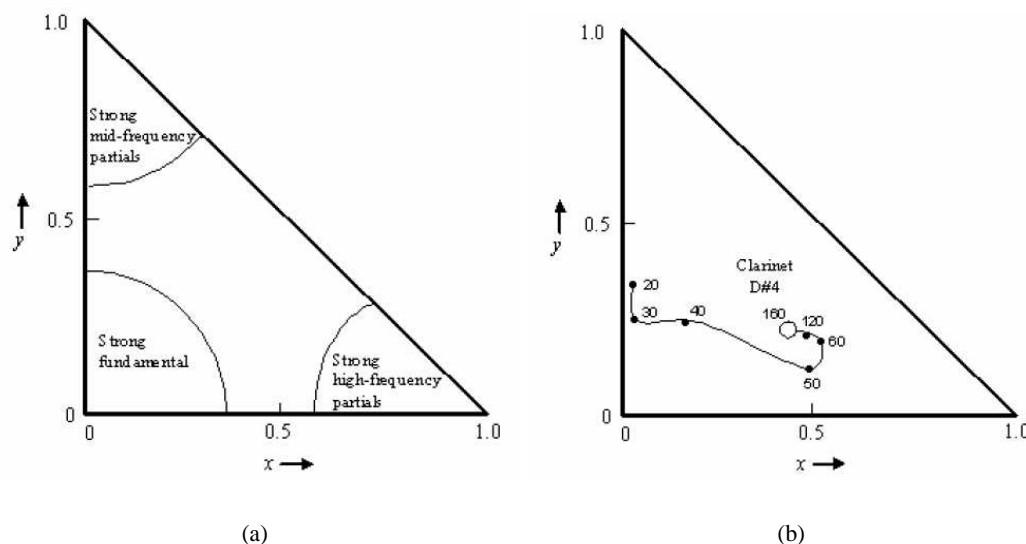


Figura 8.22: (a) Basic layout of the tristimulus diagram: $T3$ vs. $T2$. (b) Tristimulus diagram showing the timbral time course of a note played on a clarinet. The numbers alongside the plot represent time in milliseconds after the onset of the note and the white circle represents the steady state timbre, after an initial transient. (Pollard and Jansson, 1982)

$T3$ e nellasse y è indicato $T2$. Pertanto i suoni con alto $T1$ (fondamentale forte) sono vicino

allorigine, con alto T2 (alte armoniche medie) sono in alto, con alto T3 (alte armoniche superiori) sono a destra. Inoltre il fatto che la somma di $T1 + T2 + T3 = 1$ implica che i suoni siano rappresentati dentro il triangolo con vertici in (0,0), (0,1) e (1,0). In fig. 8.22(b) è rappresentata l'evoluzione del tristemulus di una nota di clarinetto.

- bandwidth

$$BW = \frac{\sum_{k=0}^{N-1} |X(k)| \cdot |f_k - BR|}{\sum_{k=0}^{N-1} |X(k)|}$$

- ampiezza (valore efficace del suono)

$$amp = \sqrt{\sum_k X(k)^2}$$

- spectral rolloff definito come la frequenza R_s sotto la quale è concentrata 85% della distribuzione di ampiezza

$$\sum_{k=1}^{R_s} |X(k)| = 0.85 \cdot \sum_{k=1}^{N-1} |X(k)|$$

Questa è una altra misura della forma dello spettro.

- flusso spettrale (Spectral Flux) è definito come la distanza euclidea tra due spettri in ampiezza di frames consecutivi

$$SF = \sum_{k=1}^{N-1} [N_t(k) - N_{t-1}(k)]^2$$

dove $N_t(k)$ e $N_{t-1}(k)$ sono rispettivamente le ampiezze spettrale della trasformata di Fourier del frame all'istante t e $t - 1$. Il flusso spettrale è una misura della quantità di variazione locale dello spettro ed è considerato, da alcuni esperimenti, essere correlato alla terza dimensione percettiva del timbro (vedi la percezione del timbro nel capitolo 2 delle dispense).

M-8.19

Implement the feature descriptors described in this section and compare their effectiveness on different kinds of sounds. Plot their evolution from the attack to the decay of a instrumental tone. Plot their evolution on a singing voice (e.g. Tom's Dinner by Susan Vega)

8.4.3 Attributi di segmento sonoro

Quando si considera la successione temporale di un attributo nei frames successivi, si ottiene una funzione del tempo definita negli istanti di riferimento dei vari frames. temporalmente i punti saranno separati del hop size usato nell'analisi. Essi sono quindi sottocampionati rispetto al segnale. Con forme di interpolazione o approssimazione, eventualmente smussando l'andamento, si possono ottenere gli andamenti a frequenza di campionamento.

Come detto sopra, è importante anche caratterizzare l'evoluzione temporale di questi parametri. Inoltre si possono ricavare degli attributi di segmento. Con il termine segmento si intende una porzione di suono di caratteristiche omogenee (ad esempio il regime sostenuto) o funzionalmente simili (ad esempio il transitorio di attacco). L'operazione di segmentazione del suono consiste nel dividerlo in segmenti e viene fatta esaminando gli attributi a livello di segnale. Le traiettoria degli attributi viene descritta calcolando per ogni parametro *par*

- la derivata al frame j

$$der(j) = \frac{par(j) - par(j-1)}{H/F_S}$$

dove H è l'hop size e F_S è la frequenza di campionamento;

- media pesata del parametro su tutti i frames j

$$media_{par} = \frac{\sum_j par(j) amp(j)}{\sum_j amp(j)}$$

- varianza pesata del parametro su tutti i frames j

$$var_{par} = \frac{\sum_j [par(j) - media_{par}]^2 \cdot amp(j)}{\sum_j amp(j)}$$

Ci sono alcuni parametri che descrivono le caratteristiche di una nota. Essi si ottengono una volta separata la nota dal contesto.

- logaritmo del tempo di attacco (Log-Attack Time)

$$LAT = \log_{10}(attack_time)$$

dove *attack_time* è la durata del tempo di attacco del suono. Questo parametro è molto importante per il riconoscimento dei timbri. Esso infatti è correlato ad una delle dimensioni percettive dello spazio timbrico ed è stato inserito tra i descrittori temporali del timbro in MPEG7. Si noti che la determinazione dell'inizio e fine dell'attacco, è una operazione difficile: spesso la nota è preceduta da rumore o altro, che rende incerta la determinazione dell' effettivo inizio. Inoltre dall'esame dell'involuppo temporale, non è neppure facile determinare il termine. Spesso si fa riferimento al massimo dell'involuppo; questo va bene per suoni percussivi, come il pianoforte. Per suoni ad eccitazione sostenuta, come gli archi o i fiati, il massimo può essere trovato ben in avanti, quando il transitorio di attacco è terminato; il musicista spesso tende a controllare l'espressività del suono, mediante il controllo dell'involuppo temporale. Si usa quindi prendere come soglie valori percentuali dell'ampiezza massima, ad esempio rispettivamente 10 % e 90 % percento del massimo.

- rapporto rumore - parte armonica (Noise to Harmonic Ratio - NHR) definito come il rapporto tra l'energia della parte rumorosa e l'energia della parte armonica per la parte del regime (sustain) si definisce la modulazione dell'energia e la modulazione della frequenza fondamentale. na volta separato il regime, si sottrae dall'involuppo d'energia in dB la sua tendenza (lineare in dB). Poi si calcola lo spettro dell'involuppo corretto e si seleziona il picco nell'estensione [1 - 10 Hz]. Con lo stesso procedimento si determina la modulazione del periodo, sottraendo la tendenza lineare della fondamentale durante il regime, e poi trovando il massimo nello spettro dell'involuppo dell'altezza corretto. La frequenza del massimo corrisponderà alla frequenza del tremolo, normalmente tra 4 e 8 Hz.
- baricentro dell'involuppo temporale (temporal centroid)

$$TC = \frac{\sum_t inv(t) \cdot t}{\sum_t inv(t)}$$

dove $inv(t)$ è l'inviluppo temporale. Questo parametro è molto utile per distinguere i suoni percussivi da quelli sostenuti. Anche questo parametro è inserito tra i descrittori temporali del timbro in Mpeg7.

M-8.20

Implement the feature descriptors described in these sections and compare their effectiveness on different kinds of sounds

8.4.4 Onset detection

A musical note is considered as composed of a initial attack transient phase, followed by a steady-state phase and a final decay. Attack transients are zones of short duration and fast variations of the signal spectral content (non-stationarity), where resonances are still building up. Their perception is caused by changes in the intensity, pitch or timbre of a sound. Because of the unpredictability of such changes, they are difficult to model. Attack transients precede the steady state of the signal, when the signal is stationary, thus easily predictable. Note onset is defined as the beginning of attack transient of a note.

The boundaries between notes and different types of events are often ill-defined. The performer can introduce variations and modulations in a given sound without implying the presence of new notes. This can also occur as a consequence of the processing of the acoustic signal, recording conditions or just as an expressivity feature in the musical performance (i.e. vibratos in woodwind, brass and string instruments). Detection of onsets in polyphonic mixtures is difficult even for human listeners. Attack transients present some typical behaviour: ¹

Energy burst: in a note's energy profile, the highest concentration of energy can be found during the attack (when a steep increase can be observed). After that, energy progressively decreases (Fig. 8.23 (a)). The more impulsive the components of the signal are (percussive sounds as opposed to tonal - more sinusoidal - sounds), the more sudden this increase-decrease energy characteristic becomes.

Duration: the attack part of a note is usually very short, introducing significant changes to the signal (Fig. 8.23). This abruptness is a trademark of transients. It is particularly acute for percussive sounds.

Surprise: this is also related to the abruptness of transients, but from the statistical point of view. New events are unconnected to previous events, thus cannot be predicted from these. The proliferation of elements whose values are completely unexpected is more likely during transients.

Chaotic nature: during transients, the signal includes unstable chaotic elements, which quickly stabilise when entering the steady state (see Fig. 8.23 (b)). These elements are not only highly uncorrelated with previous and future signal values, but also within different signal elements at a given time.

Steady-state: although obvious, an important characteristic of transients is that they are followed by the steady-state of the note. Chaotic components followed by chaotic components can account for noise, while a stable follow-up hints at the possible presence of a note.

¹from PhD dissertation of J. Bello 2003

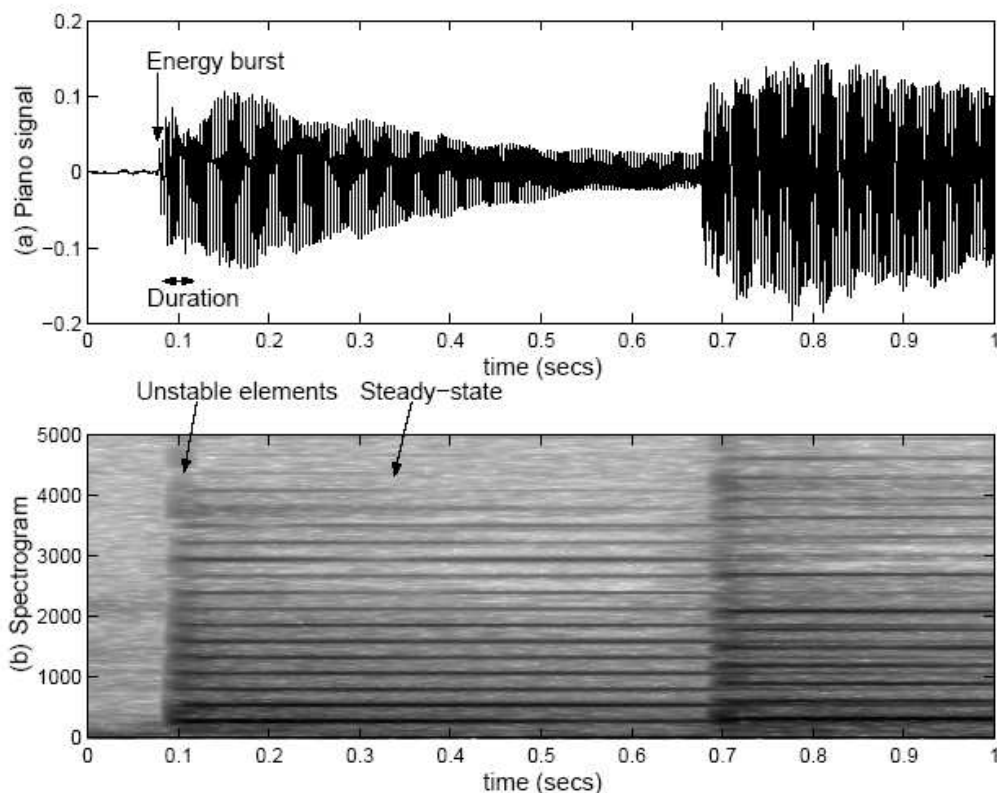


Figura 8.23: A sequence of two piano notes (a) and the corresponding spectrogram (b). The energy increase, short duration and instability related to transients can be observed as well as the stability of the steady-state part [from Bello 2003].

8.4.4.1 Onset detection by Local energy

Despite the number of variants, practically all of them are based on the calculation of a first order difference function of the signal amplitude envelopes and taking the maximum rising slope as an onset or an onset component. An example is the algorithm based on the surfboard method of Schloss (1985), which involves smoothing the signal to produce an amplitude envelope and finding peaks in its slope using linear regression. In fig. 8.24 the effect of a simple onset detector based on Local energy is shown. In fig. 8.24(a) the time-domain audio signal; in fig. 8.24(b) its smoothed amplitude envelope drawn in bold over it, computed by a 40ms windowed RMS smoothing with 75% overlap and in fig. 8.24(c) peaks in slope shown by dotted lines tangential to the envelope. This method is lossy, in that it fails to detect the onsets of many notes which are masked by simultaneously sounding notes. Occasional false onsets are detected, such as those caused by amplitude modulation in the signal.

The first order difference function reflects well the loudness of an onsetting sound, but its maximum values fail to precisely mark the time of an onset. This is due to two reasons. First, especially low sounds may take some time to come to the point where their amplitude is maximally rising, and thus that point is crucially late from the physical onset of a sound and leads to an incorrect cross-band association with the higher frequencies. Second, the onset track of a sound is most often not monotonically increasing, and thus we would have several local maxima in the first order difference function

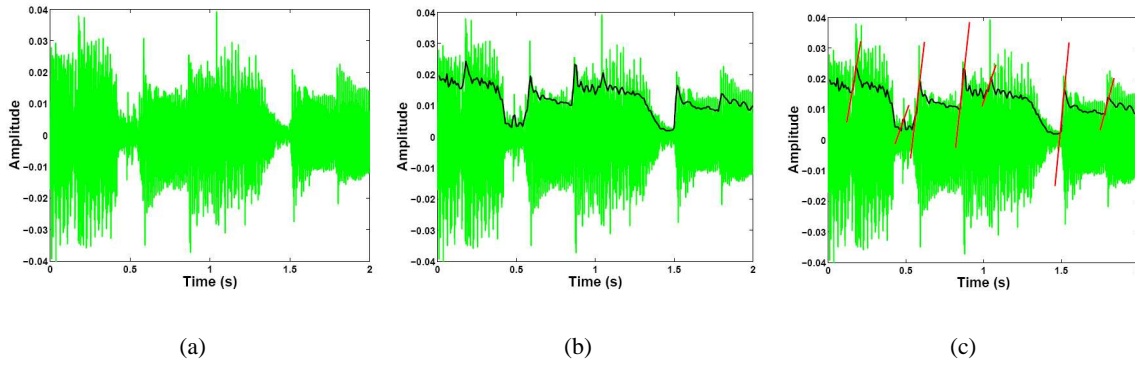


Figura 8.24: Example of onset detector based on local energy: time-domain audio signal (a), 40ms windowed RMS smoothing with 75% overlap (b), peaks in slope of envelope (c).

near the physical onset.

It is possible to handle both of these problems by using the relative difference function. Let $E(t)$ the Short time energy as defined in sec. 8.2.1. The relative difference function is defined as

$$D_r(t) = \frac{1}{E(t)} \frac{d}{dt} E(t) = \frac{d}{dt} \log E(t)$$

This function computes the amount of change in relation to the signal level. This is the same as differentiating the logarithm of the energy envelope. This is psycho-acoustically relevant, since perceived increase in signal amplitude is in relation to its level, the same amount of increase being more prominent in a quiet signal. Indeed the just detectable change in intensity is approximately proportional to the intensity of the signal, i.e. $\delta I/I$, the Weber fraction, is a constant. This relationship holds for intensities from about 20 dB to about 100 dB above the absolute threshold. Onset components are detected by a simple peak picking operation, which looks for peaks above a global threshold in the relative difference function $D_r(t)$.

The relative difference function effectively solves the above mentioned problems by detecting the onset times of low sounds earlier and, more importantly, by handling complicated onset tracks, since oscillations in the onset track of a sound do not matter in relative terms after its amplitude has started rising. In fig. 8.25 the absolute and relative difference functions of the onset of a piano sound, on six different frequency bands, are plotted. Both of the benefits discussed can be seen clearly. To improve the performance of the onset detector, first the overall loudness of the signal is normalized to a reference level using a psychoacoustics model of loudness. Then a filterbank divides the signal into many non-overlapping bands (often critical bands are used). At each band, we detect onset components and determine their time and intensity. In final phase, the onset components are combined to yield onsets.

Energy-based algorithms are fast and easy to implement, however their effectiveness decreases when dealing with non-percussive signals and when transient energy overlaps in complex mixtures. Energy bursts related to transient information are more noticeable at higher frequencies as the tonal energy is usually concentrated at lower frequencies, masking the effect of these variations on the signal content. More advanced models utilize band-wise processing and a psychoacoustic model of intensity coding to combine the results from the separate frequency bands.

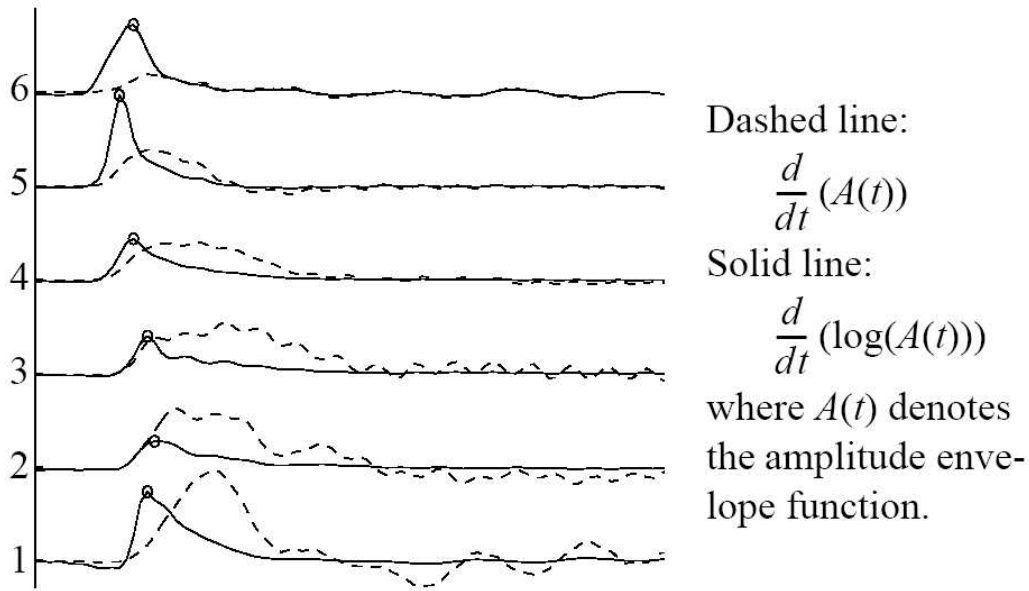


Figura 8.25: Onset of a piano sound. First order absolute (dashed) and relative (solid) difference functions of the amplitude envelopes of six different frequency bands [from Klapuri 1999].

8.4.4.2 Onset detection in frequency domain

This attack transient noise is particularly noticeable at high frequency locations, since at low frequencies, high concentrations of energy (in the bins corresponding to the first few harmonics of the played note) mask this effect.

The High Frequency Content (HFC) function, is defined, for the j th frame, as

$$D_H(j) = \sum_k k |X_j(k)|$$

where $|X_j(\cdot)|$ is the spectral magnitude of the j th frame. Aim of this function is to emphasize the high frequency content of the sound and it works well for identifying percussive sounds. If compared with energy, this HFC function has greater amplitude during the transient/attack time.

8.4.4.3 Onset detection by complex domain approach

The HFC precisely identifies percussive onsets, but is less responsive to non-percussive components. In the complex domain approach, to cope with harmonic changes of low transient timbres, a target STFT value \hat{X}_k is generated as

$$\begin{aligned} \hat{X}_k[n] &= |\hat{X}_k[n]| e^{j\hat{\phi}_k[n]} \\ \hat{\phi}_k[n] &= \text{princarg}(2\phi_k[n-1] - \phi_k[n-2]) \end{aligned}$$

where $\phi_k[n]$ is the estimated phase deviation. The measure of the Euclidean distance, in the complex domain, between the target STFT value \hat{X}_k and the observed STFT X_k allows the definition of a

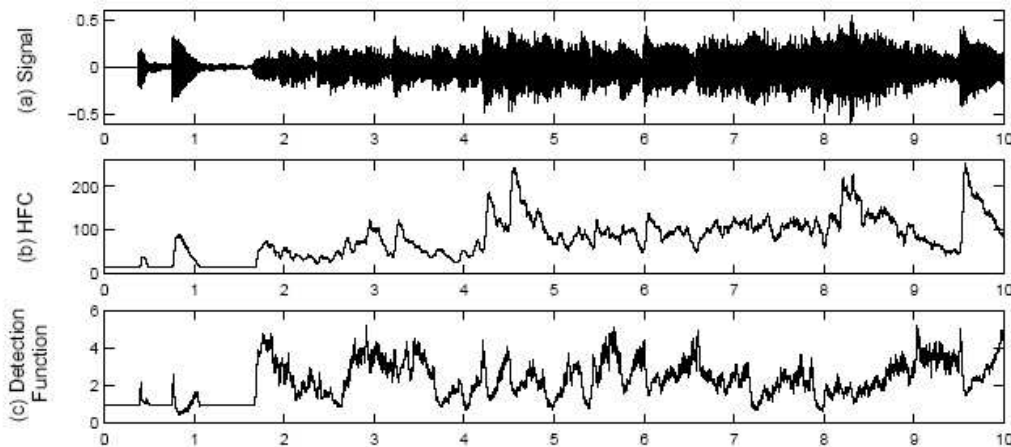


Figura 8.26: Piano signal (a), its high frequency content (b), the profile of High Frequency Content function [from Bello 2003].

detection function

$$D_C[n] = \frac{1}{N} \sum_k ||\hat{X}_k[n] - X_k[n]||^2$$

The detection function still contains spurious peaks and some pre-processing and low pass filtering is required before peak picking.

M-8.21

Implement the onset detectors described in these sections and compare their performance on different kinds of music

8.4.5 Meter estimation algorithms

As a part of a larger project of modeling the cognition of basic musical structures, Temperley and Sleator proposed a meter estimation algorithm for arbitrary MIDI files [Temperley99: D. Temperley and D. Sleator, Modeling Meter and Harmony: A Preference-Rule Approach, Computer Music Journal, 23(1), 1027, Spring 1999]. The algorithm was based on implementing the preference rules verbally described in [Lerdahl 1983], and produced the whole metrical hierarchy as output.

Dixon proposed a rule-based system to track the tactus pulse of expressive MIDI performances [S. Dixon, Automatic Extraction of Tempo and Beat from Expressive Performances, J. New Music Research 30 (1), 39-58, 2001]. The method works quite well for MIDI files of all types but has problems with audio files which do not contain sharp attacks. The source codes of both Temperley's and Dixon's systems are publicly available for testing. See

<http://www.link.cs.cmu.edu/music-analysis>

and

<http://www.oefai.at/~simon/beatroot>

M-8.22

The source codes of both Temperley's and Dixon's systems are publicly available for testing. Test and compare their performance on different kinds of music.

Indice

8	Analisi dei suoni	8.1
8.1	Introduzione	8.1
8.2	Parametri nel dominio del tempo	8.2
8.2.1	Short-Time Average Energy e Magnitude	8.5
8.2.2	Short-Time Average Zero-Crossing Rate	8.7
8.2.3	Short-Time Autocorrelation Function	8.10
8.2.4	Short-Time Average Magnitude Difference Function	8.13
8.2.5	Stima del <i>pitch</i> (F0)	8.15
8.3	Stima dell'involuppo spettrale	8.16
8.3.1	Stima dell'involuppo spettrale mediante banco di filtri	8.16
8.3.2	Stima dell'involuppo spettrale mediante predizione lineare (LPC)	8.17
8.3.2.1	Esempi di analisi mediante predizione lineare (LPC)	8.18
8.3.3	Stima dell'involuppo spettrale mediante cepstrum	8.18
8.3.4	Analisi mediante mel-cepstrum	8.21
8.4	Attributi a medio livello ricavabili dall'analisi spettrale	8.23
8.4.1	Attributi a basso livello	8.24
8.4.2	Attributi a livello superiore	8.26
8.4.3	Attributi di segmento sonoro	8.28
8.4.4	Onset detection	8.30
8.4.4.1	Onset detection by Local energy	8.31
8.4.4.2	Onset detection in frequency domain	8.33
8.4.4.3	Onset detection by complex domain approach	8.33
8.4.5	Meter estimation algorithms	8.34