UNIVERSITÁ DEGLI STUDI DI MILANO

Corso di Laurea in Informatica Musicale

Dipartimento di Informatica

Riconoscimento automatico del profilo armonico di segnali audio: implementazione ed ottimizzazione

Relatore: Prof. Goffredo Haus
Correlatore: Dott. Giorgio Presti

Elaborato di:
Alessandro D'AMELIO, matricola 792875

Anno Accademico 2013/2014

Indice

Capitolo 1

Introduzione

In questo elaborato viene descritto il problema dell'estrazione automatica di descrittori musicali di alto livello dai segnali audio, con particolare riferimento all'armonia quale elemento fondamentale della musica tonale.
Viene inoltre proposta una implementazione delle tecniche esposte utilizzando il linguaggio di scripting offerto dall'ambiente di sviluppo MATLAB

L'elaborato è organizzato come segue:
viene fornita nel secondo capitolo una breve panoramica sui principi teorici utili alla comprensione degli argomenti esposti.
Nel terzo capitolo vengono trattati nel dettaglio i problemi relativi all'estrazione delle features finalizzata al riconoscimento della tonalità di un brano musicale. Vengono quindi presentati due differenti approcci (pitch tracking e chord recognition) rivolti rispettivamente ai segnali monofonici e polifonici.
Nel quarto capitolo viene descritto il problema dell'estrazione della tonalità.
Nel quinto vengono presentati i test e i risultati ottenuti.

1.1 Motivazioni

Negli ultimi anni si è assistito ad una enorme crescita delle collezioni di musica in formato digitale all'interno di database di grandi dimensioni. La musica è ad oggi uno dei tipi di informazione più diffusa online; basti pensare a tutti i servizi di streaming e downloading di musica sul World Wide Web.

La mole di informazione contenuta all'interno di questi database richiede nuove tecniche di catalogazione, ricerca e organizzazione dell'informazione musicale. L'insieme di queste tecniche va sotto il nome di Music Information Retrival (MIR).
Il Music Information Retrival fornisce un supporto al tradizionale approccio di indicizzazione dell'informazione musicale basata sui metadata, che sebbene rappresenti un approccio valido se si tratta di collezioni di piccole e

medie dimensioni, diventa estremamente dispendioso per collezioni molto numerose. É stato infatti stimato un tempo di 20-30 minuti per traccia per l'inserimento dei metadati finalizzato alla costruzione di un database predisposto per una similarity-based search [1].

In aggiunta ai sistemi basati sui metadati, informazioni sul contenuto musicale possono essere utilizzate per la ricerca. La descrizione basata sui contenuti, permette di identificare quello che l'utente sta cercando, anche quando questi non sa esattamente quello che cerca; ad esempio *Shazam* può riconoscere un brano a partire da un sample registrato con un cellulare e fornire informazioni circa l'artista, l'album e il titolo, oltre a un link per il downloading del brano originale.

L'obiettivo di questo lavoro è quello di fornire una implemetazione di alcune delle tecniche di estrazione di descrittori musicali di alto livello da segnali audio digitali, in particolar modo note, accordi e tonalità.

Capitolo 2

Fondamenti Teorici

In questo capitolo vengono fornite alcune informazioni di base sull'analisi in frequenza dei segnali. Vengono inoltre considerati alcuni aspetti fondamentali della teoria musicale, utili alla comprensione dell'armonia.

2.1 Analisi in frequenza dei segnali

2.1.1 Segnali periodici

Una delle classi di segnali cui si farà riferimento all'interno di questo elaborato é quella dei segnali che si ripetono periodicamente nel tempo, detti segnali periodici. Piú precisamente diremo che un segnale $f(t)$ é periodico di periodo T se si verifica che

$$f(t) = f(t + kT), \qquad \forall t \in \mathbb{R}$$

Il minimo valore del periodo T > 0 che soddisfa la definizione di periodicitá è chiamato periodo fondamentale ed è denotato da T0. L'inverso di questo valore rappresenta la frequenza fondamentale del segnale.

2.1.2 Serie di Fourier

I segnali periodici possono essere rappresentati come combinazione lineare di segnali esponenziali complessi (fasori di modulo unitario) impiegando oggetti matematici che vanno sotto il nome serie e trasformate di Fourier (quest'ultima è una generalizzazione ai segnali non periodici):

$$f(t) = \sum_{n=-\infty}^{\infty} c_{\mathrm{n}} e^{\mathrm{in}\omega_0 \mathrm{t}} \qquad con \qquad c_{\mathrm{n}} = \frac{1}{T} \int_{-\frac{T}{2}}^{\frac{T}{2}} f(t) e^{-\mathrm{in}\omega_0 \mathrm{t}} dt$$

Queste due equazioni prendono rispettivamente il nome di equazionione di sintesi ed equazione di analisi della serie di Fourier. La prima consente di

sintetizzare il segnale $f(t)$ sovrapponendo i singoli fasori, la seconda consente di decomporlo calcolandone i coefficienti complessi.

Attraverso l'identità di Eulero

$$cos(\Theta) + isen(\Theta) = e^{\mathrm{i}\Theta}$$

è possibile riscrivere in modo equivalente queste equazioni utilizzando le funzioni trigonometriche; si dimostra infatti che

$$f(t) = \sum_{n=-\infty}^{\infty} c_{\mathrm{n}} e^{\mathrm{in}\omega_0 \mathrm{t}} = \frac{a_0}{2} + \sum_{n=1}^{\infty} [a_{\mathrm{n}} cos(n\omega_0 t) + b_{\mathrm{n}} sin(n\omega_0 t)]$$

Da questa rappresentazione si nota come tutti i segnali periodici possano essere rappresentati come sovrapposizione di funzioni trigonometriche con frequenza multipla di una frequenza data. A partire da questa caratteristica dei segnali periodici sono state messe a punto tecniche per determinarne la frequenza fondamentale che saranno esposte nel capitolo successivo.

2.1.3 Trasformata di Fourier

La trasformata di Fourier, come detto nella sezione precedente, è una generalizzazione ai segnali non periodici. Non potendo rappresentare questi ultimi come somma di armoniche con frequenza multipla (da cui si ottengono solo segnali periodici), si utilizza un procedimento più sofisticato che va sotto il nome di *Trasformata di Fourier*, la cui giustificazione matematica consiste nell'immaginare un qualunque segnale come periodico di periodo T con $T \to \infty$.
Dato $f(t)$ segnale non periodico, chiamiamo $f_{\mathrm{T}}(t)$ la funzione periodica di periodo T che coincide con $f(t)$ nell'intervallo $[-\frac{T}{2}, \frac{T}{2}]$; possiamo scrivere:

$$f(t) = \lim_{T \to \infty} f_{\mathrm{T}}(t).$$

Essendo $f_{\mathrm{T}}(t)$ una funzione periodica, puo essere sviluppata in serie di Fourier:

$$f_{\mathrm{T}}(t) = \sum_{n=-\infty}^{\infty} c_{\mathrm{n}} e^{\mathrm{in}\omega_0 \mathrm{t}} \qquad con \qquad c_{\mathrm{n}} = \frac{1}{T} \int_{-\frac{T}{2}}^{\frac{T}{2}} f_{\mathrm{T}}(t) e^{-\mathrm{in}\omega_0 \mathrm{t}} dt$$

Sapendo che $\omega_0 = 2\pi\nu$, chiamiamo $\nu_{\mathrm{n}} = n\nu$ la frequenza dell'ennesima armomnica e $\Delta\nu = \nu_{\mathrm{n+1}} - \nu_{\mathrm{n}} = \frac{1}{T}$ la distanza tra una frequenza e la successiva; ponendo $F(\nu) = c_{\mathrm{n}} T$ possiamo scrivere:

$$f_{\mathrm{T}}(t) = \sum_{n=-\infty}^{\infty} F(\nu_{\mathrm{n}}) e^{\mathrm{i}2\pi\nu_{\mathrm{n}}\mathrm{t}} \Delta\nu$$

$$F(\nu_{\mathrm{n}}) = \int_{-\frac{T}{2}}^{\frac{T}{2}} f_{\mathrm{T}}(t) e^{-\mathrm{i}2\pi\nu_{\mathrm{n}}\mathrm{t}}$$

Per $T \to \infty$, $\Delta\nu \to 0$, la serie precedente si riduce ad un integrale:

$$f(t) = \int_{-\infty}^{\infty} F(\nu) e^{\mathrm{i}2\pi\nu\mathrm{t}} d\nu \qquad F(\nu) = \int_{-\infty}^{\infty} f(t) e^{-\mathrm{i}2\pi\nu\mathrm{t}} dt$$

$F(\nu)$ è detta *Trasformata di Fourier* del segnale $f(t)$ e ne rappresenta il contenuto frequenziale(spettro).
$f(t)$ è detta *Anti-trasformata di Fourier* e permette di ri-ottenere il segnale nel dominio del tempo senza perdita di informazione.

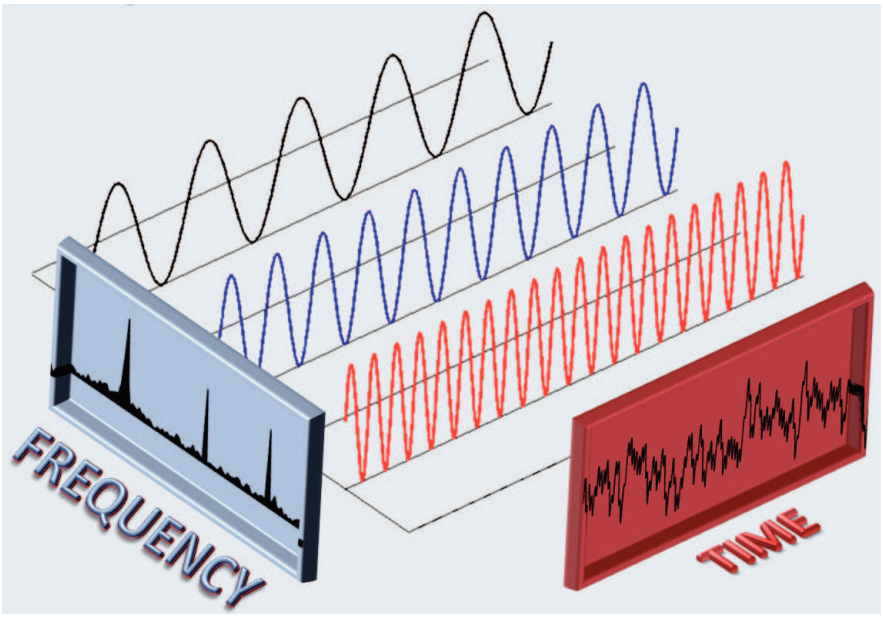

Figura 2.1: Dominio del tempo - frequenza

2.1.4 Segnali discreti e non stazionari

La trasformata e l'anti-trasformata di Fourier, operano su segnali continui nel dominio del tempo e delle frequenze.
Per poterle implementare su un calcolatore digitale, è necessario operare su segnali discreti, va quindi introdotta una versione discreta della trasformata, che prende il nome di trasformata discreta di Fourier (DFT).

DFT

Fissato un passo di campionamento τ, per ogni segnale $f(t)$ consideriamo il segnale $f_{\mathrm{s}}(t)$ ottenuto campionando $f(t)$ ai tempi $n\tau(-\infty < n < \infty)$ mediante una funzione impulsiva $\delta(t)$; otteniamo:

$$f_{\mathrm{s}}(t) = \sum_{n=-\infty}^{+\infty} f(n\tau)\delta(t - n\tau)$$

L'informazione contenuta nel segnale $f(t)$ viene approssimata con quella contenuta nel vettore discreto x formato da N campioni del segnale campionato con passo τ.

$$x(n) = f(n\tau) \qquad con \qquad n = 0, ..., N-1$$

Lo spettro di questa funzione si ottiene facendo la Trasformata di Fourier dell'espressione precedente; dato che il segnale considerato è un segnale campionato e quindi a tempo discreto, questa operazione prende il nome di *Trasformata di Fourier a tempo Discreto (DTFT)*; sapendo che $\mathcal{F}(\delta(t - n\tau)) = e^{-\mathrm{i}2\pi\nu\mathrm{n}\tau}$ otteniamo:

$$F_{\mathrm{s}}(\nu) = \sum_{n=-\infty}^{+\infty} x(n)e^{-\mathrm{i}2\pi\nu\mathrm{n}\tau}$$

La funzione $F_{\mathrm{s}}(\nu)$ rappresenta lo spettro *continuo* del segnale $x(n)$; per poter trattare questo segnale digitalmente è necessario operare un campionamento anche nel dominio della frequenza con intervalli di ampiezza pari a $F_0 = \frac{f_{\mathrm{s}}}{N}$ con $f_{\mathrm{s}} = \frac{1}{\tau}$ = *frequenza di campionamento*. La F_0 è detta *risoluzione frequenziale della DFT*:

$$X(k) = F_{\mathrm{s}}(kF_0) \qquad con \qquad k = 0, ..., N-1$$

otteniamo:

$$X(k) = \sum_{n=0}^{N-1} x(n)e^{-\mathrm{i}2\pi\mathrm{kF0n}\tau} = \sum_{n=0}^{N-1} x(n)e^{-\mathrm{i}2\pi\mathrm{k}\frac{1}{\mathrm{N}\tau}\mathrm{n}\tau} = \sum_{n=0}^{N-1} x(n)e^{-\mathrm{ik}\frac{2\pi}{\mathrm{N}}\mathrm{n}}$$

Quindi:

$$X(k) = \sum_{n=0}^{N-1} x(n)e^{-\mathrm{ik}\frac{2\pi}{\mathrm{N}}\mathrm{n}}, \qquad x(n) = \frac{1}{N}\sum_{n=0}^{N-1} X(k)e^{\mathrm{ik}\frac{2\pi}{\mathrm{N}}\mathrm{n}}$$

prendono il nome rispettivamente di trasformata e antitrasformata discrete di Fourier.

STFT

La trasformata di Fourier permette di evidenziare la prensenza delle componenti armoniche nel segnale, ma non permette di ricavare informazioni su *quando* tali frequenze siano effettivamente presenti.

Per segnali non stazionari (come la maggior parte dei segnali di nostro interesse) occorre inserire nella trasformazione una dipendenza dal tempo.

Il modo più immediato pre farlo consiste nel rendere *locale* la trasformata, operando solo su porzioni del segnale ottenute moltiplicandolo per una *finestra* che trasla nel tempo:

$$STFT_{\mathrm{x}}(\tau,\nu) = \int_{-\infty}^{+\infty} x(t)g(t-\tau)e^{-\mathrm{i}2\pi\nu t}dt$$

Nasce così la trasformata di Fourier a breve termine o Short Time Fourier Transform (STFT).
La funzione $g(t)$ è detta funzione finestra.

La moltiplicazione di due segnali nel dominio del tempo corrisponde alla convoluzione degli spettri nel dominio della frequenza per il *teorema di convoluzione*. La STFT fornisce quindi lo spettro del segnale alterato dalla presenza della finestra.

In questo senso la finestra rettangolare risulta molto invasiva provocando fenomeni di *leakage* molto marcati a causa delle rapide transizioni che la caratterizzano nel dominio del tempo e che, nel dominio della frequenza si traducono in ondulazioni che si replicano su tutto lo spettro con attenuazione decrescente (Fig. 2.2)

.

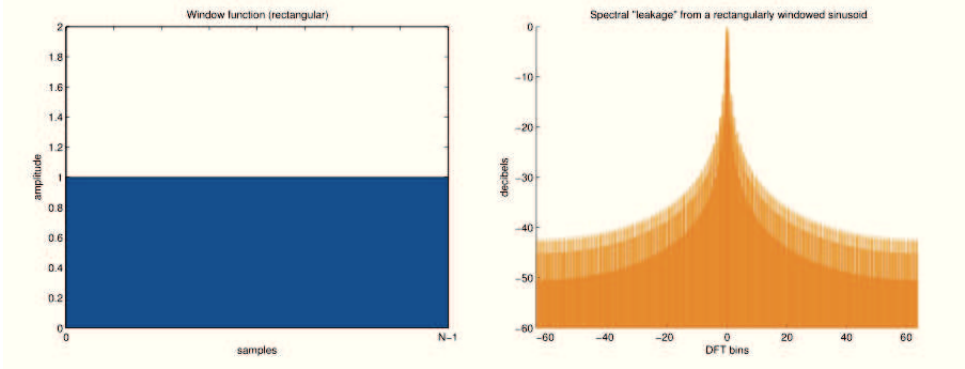

Figura 2.2: Finestra rettangolare

Si possono allora usare, per rimediare a questo inconveniente, delle finestre con dei contorni smussati e con uno spettro meno oscillante, come

la finestra di Hanning, che sono caratterizzate da una reiezione delle bande laterali maggiore rispetto alla finestra rettangolare (Fig. 2.3)
.

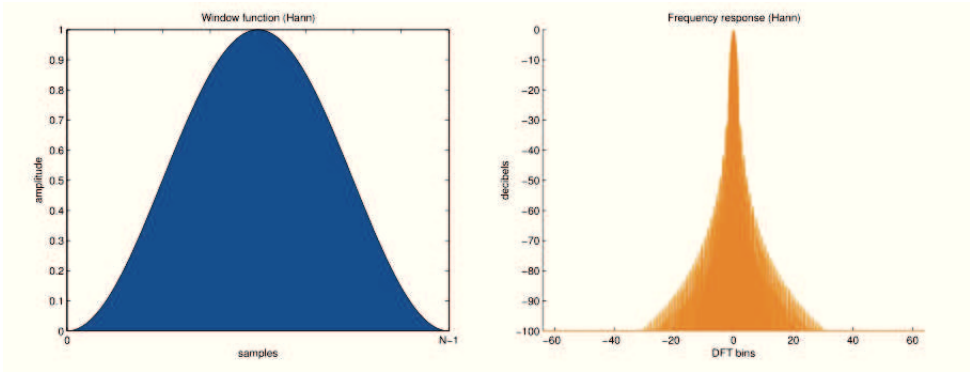

Figura 2.3: Finestra di Hanning

Sarà inoltre conveniente procedere per segmenti parzialmente sovrapposti con un *overlap* pari ad esempio al 75% della durata dell'intervallo elementare, al fine di ottenere una risoluzione temporale maggiore ed una descrizione più graduale e meglio interpolata dello spettro.

2.2 Fondamenti musicali

2.2.1 Rapporto tra frequenza e pitch

La nota è il blocco fondamentale di tutta la musica tonale. Le caratteristiche principali di una nota sono altezza, intensità, timbro e durata. In questo elaborato verrà considerata principalmente solo una di queste: l'altezza o *pitch*.

Il pitch di una nota dipende principalmente dalla sua frequenza, ma anche dall'intensità e dal contenuto armonico.
In questo lavoro ci si concentrerà essenzialmente sul rapporto tra pitch e frequenza di una nota:

A causa della fisiologia della percezione del suono, la distanza tra due suoni (intervallo) non viene misusrata come la differenza tra le frequenze, ma come rapporto tra le frequenze cioè come la differenza tra i loro logaritmi.

$$R = f_2/f_1$$

Tra queste due grandezze, quindi, sussiste una dipendenza logaritmica tale per cui ad un raddoppio della frequenza, corrisponde un incremento di ottava del pitch, qualunque sia la frequenza di partenza.

Ad esempio, il LA centrale ha una frequenza di 440 Hz ed il LA posto un'ottava sopra ha una frequenza di 880 Hz, mentre quello un'ottava sotto ha una frequenza di 220 Hz. Il rapporto tra le frequenze di due note separate da un'ottava è perciò di 2:1.

2.2.2 Le scale - la scala temperata

Una scala si può definire come una successione ascendente o discendente di suoni (indicati da note aventi un determinato pitch) compresi nell'ambito di una o più ottave.

Le scale maggiori e minori sono formate da 7 suoni(7 classi di pitch, o pitch class), ognuno con una diversa funzione armonica che prende il nome di grado. Una scala viene definita maggiore o minore, a seconda dello schema di intervalli (siano essi toni T o semitoni S) utilizzato nella sua costruzione:

T T S T T T S

per le scale maggiori;

T S T T S T T

per quelle minori.

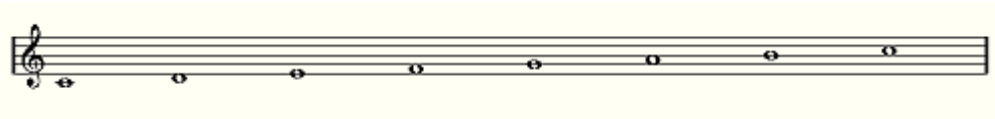

Figura 2.4: Scala di DO Maggiore

Nel corso della storia sono stati messi a punto diversi sistemi di suddivisione dell'ottava, ad esempio scala pitagorica o scala naturale; in questo elaborato verrà preso in cosiderazione il sistema di suddivisione che va sotto il nome di *temperamento equabile*.

Il temperamento equabile è basato sulla suddivisione dell'ottava in intervalli tra di loro uguali indipendentemente dalla tonalità utilizzata. Nell'uso più frequente, l'ottava è suddivisa in 12 parti (semitoni).

La scala prodotta secondo il temperamento equabile si ottiene, quindi, dividendo l'ottava in dodici parti uguali;

Poiché l'ottava è rappresentata dal rapporto 2:1, e per passare dalla frequenza di una nota a quella successiva è necessario operare un prodotto (non una somma), l'intervallo più piccolo è quello che, moltiplicato per se stesso dodici volte (cioé elevato alla 12) dà 2. Esso corrisponde al semitono temperato ed è il numero irrazionale $\sqrt[12]{2}$. In questo modo dodici semitoni coprono esattamente l'intervallo di un'ottava (Fig. 2.5).

Nota	Rapporto	Frequenza (Hz)
Do	1 : 1	261.6
Do# o Reb	$\sqrt[12]{2}$	277.2
Re	$\sqrt[12]{2^2}$	293.7
Re# o Mib	$\sqrt[12]{2^3}$	311.1
Mi	$\sqrt[12]{2^4}$	329.6
Fa	$\sqrt[12]{2^5}$	349.2
Fa# o Solb	$\sqrt[12]{2^6}$	370.0
Sol	$\sqrt[12]{2^7}$	390.0
Sol# o Lab	$\sqrt[12]{2^8}$	415.3
La	$\sqrt[12]{2^9}$	440.0
La# o Sib	$\sqrt[12]{2^{10}}$	466.2
Si	$\sqrt[12]{2^{11}}$	493.9
Do	$\sqrt[12]{2^{12}}$	523.3

Figura 2.5: Temperamento Equabile

2.2.3 Accordi (le triadi)

Un accordo è una collezione di note suonate simultaneamente. Gli accordi possono essere costruiti per sovrapposizione di terze a partire dalle scale maggiori o minori, permettendo un'armonizzazione della scala:

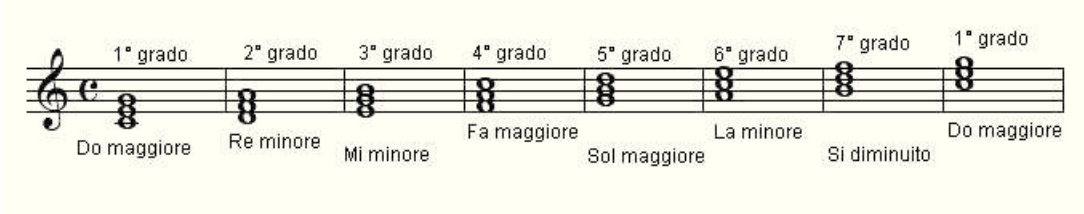

Figura 2.6: Armonizzazione scala di DO Maggiore

Mettendo insieme 3 note per sovrapposizione di terze a partire da una scala maggiore o minore si ottiene una *triade*; le triadi possono essere maggiori, minori, eccedenti o diminuite.

2.2.4 Concetto di tonalità

La tonalità costituisce l'insieme dei principi armonici e melodici che regolano i relativi legami tra accordi e/o note in un brano musicale. Le note della scala, e gli accordi costruiti su di essa, obbediscono a delle leggi che li pongono necessariamente in relazione rispetto alla tonica (nota che dà il nome alla

tonalità).

A partire da questo presupposto è possibile, sulla base di una successione di note (melodia) o di accordi (successione armonica), risalire alla tonalità del brano.

In questo lavoro la melodia e la successione armonica di un certo numero di accordi saranno le features sulle quali si baserà l'estrazione della tonalità.

Capitolo 3

Feature Extraction per il riconoscimento della tonalità nei segnali audio

Come già accennato nell'introduzione, questo lavoro utilizza due differenti approcci per l'estrazione delle features finalizzata al riconoscimento della tonalità, a seconda che si tratti di segnali audio monofonici o polifonici.
I due approcci vengono rispettivamente presentati nella prima e nella seconda sezione di questo capitolo.

3.1 Segnali monofonici (Pitch Tracking)

Per quel che riguarda i segnali audio monofonici è facile immaginare come la feature da estrarre sia la melodia. Una volta ricavate le note che compongono la melodia è possibile estrarne la tonalità d'impianto.

In questa sezione viene esposto il problema del pitch tracking, ovvero la ricerca della frequenza fondamentale (F_0) di un segnale periodico e non stazionario; vengonono presentate alcune tecniche note in letteratura ed una loro implementazione in MATLAB.

3.1.1 Harmonic Product Spectrum (HPS)

Precedentemente è stato spiegato come un segnale periodico, attraverso Fourier, sia rappresentabile da una serie di armonici multipli di una frequenza fondamentale.(Fig. 3.1).

L'Harmonic Product Spectrum sfrutta questa caratterisica dello spettro dei segnali periodici: l'asse delle frequenze viene *ricampionato* ad intervalli interi e ad ogni stadio di ricampionamento, la nuova versione viene mol-

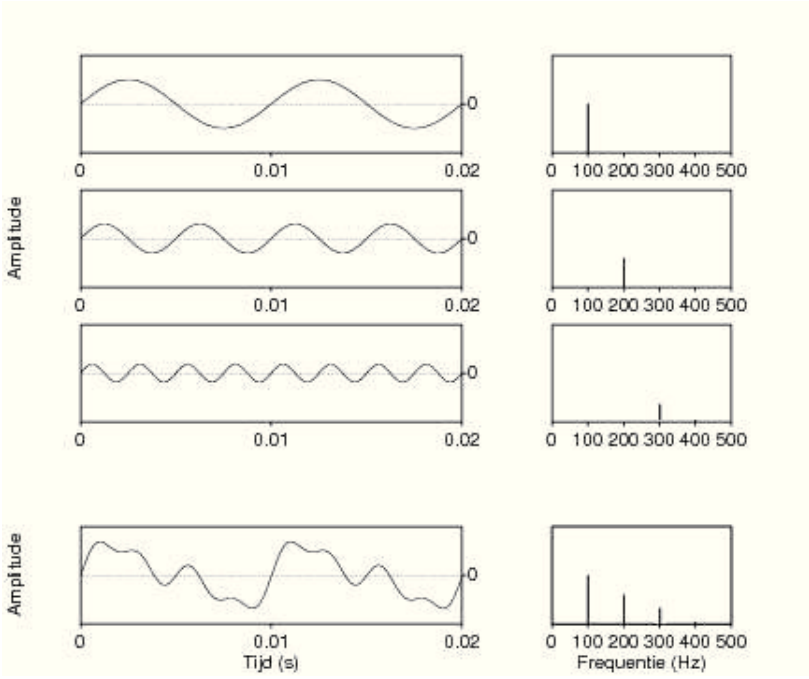

Figura 3.1: Struttura armonica di un segnale periodico

tiplicata per la precedente; questo procedimento porterà ad un picco molto marcato nel punto in cui le armoniche coincidono (vedi fig 3.2). La frequenza di questo picco sarà la frequenza fondamentale. Matematicamente:

$$HPS(\nu) = \prod_{r=1}^{R} |X(\nu r)| \,.$$

Dove R è il numero di armoniche da considerare (numero di stadi di ricampionamento) e X è la trasformata di Fourier del segnale.
La frequenza fondamentale corrisponderà al valore massimo di $HPS(\nu)$:

$$F_0 = max\left[HPS(\nu_{\mathrm{i}})\right]$$

Ipotizzando di avere a che fare con segnali non stazionari, per le motivazioni esposte nel capitolo precendente, viene utilizzata una Short Time Fourier Transform.
Il segnale viene quindi suddiviso in frames e la DFT e l'HPS vengono calcolati per ognuno di essi come riassunto nella Fig. 3.2.

Implementazione in MATLAB

Per l'implementazione in MATLAB è stata utilizzata una STFT con un framesize di 8192 campioni e un overlap del 75%. I risultati migliori sono stati

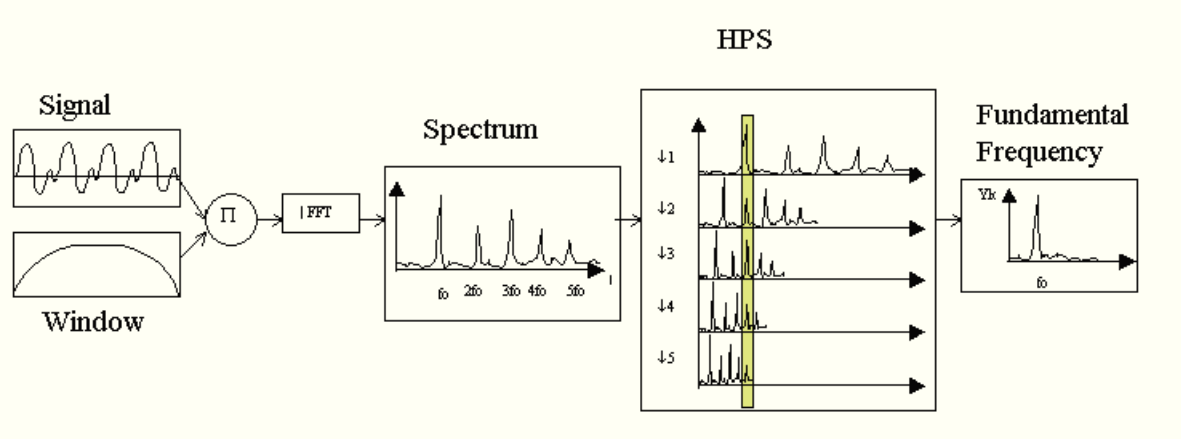

Figura 3.2: Harmonic Product Spectrum

raggiunti considerando un numero di armoniche (stadi di compressione) pari a 5.

La funzione HPS prende in ingresso 4 parametri:

nharm : Numero di armoniche da considerare.

signal : Matrice contenente il segnale nel dominio della frequenza, suddiviso in frame. L'i-esima riga rappresenta l'i-esimo campione del frame; la j-esima colonna rappresenta il j-esimo frame.

nbins : numero di punti della FFT.

mode : se *mode = 'exp'* la decimazione avviene per potenze di due, altrimenti avviene per numeri interi. Con le potenze di due vengono considerati solo gli armonici che corrispondono alle ottave, più efficace nel caso dei segnali polifonici; per il segnale monofonico, tutti gli armonici contribuiscono positivamente allo scopo.

e restituisce una matrice delle stesse dimesioni di *signal* contenente l'HPS del segnale calcolato frame per frame.

```
1              function hps = HPS(nharm,signal,nbins,mode)
2
3                  hps_step_n(:,:,1) = signal(:,:);
4
5                  if(strcmp(mode,'exp') == 1)
6                      factor = '2^n';
7                  else
8                      factor = 'n';
9                  end
```

```
10
11                  %downsample and store
12                  for n=1:nharm
13                      comodo = downsample(signal,eval(factor));
14
15                      len=ceil(nbins/eval(factor));
16                      for i=1:len
17                          hps_step_n(i,:,n+1) = comodo(i,:);
18                      end
19                  end
20
21                  hps = prod(hps_step_n,3);
22                  hps = hps(1:len,:);
```

Il tracciamento del pitch viene ottenuto cercando per ogni frame la posizione (l'indice) del valore massimo.
Questo valore viene quindi moltiplicato per la risoluzione spettrale della DFT per ottenere il pitch istantaneo (frequenza fondamentale al tempo τ).
I risultati ottenuti su una voce maschile sono rappresentati nella Fig. 3.3

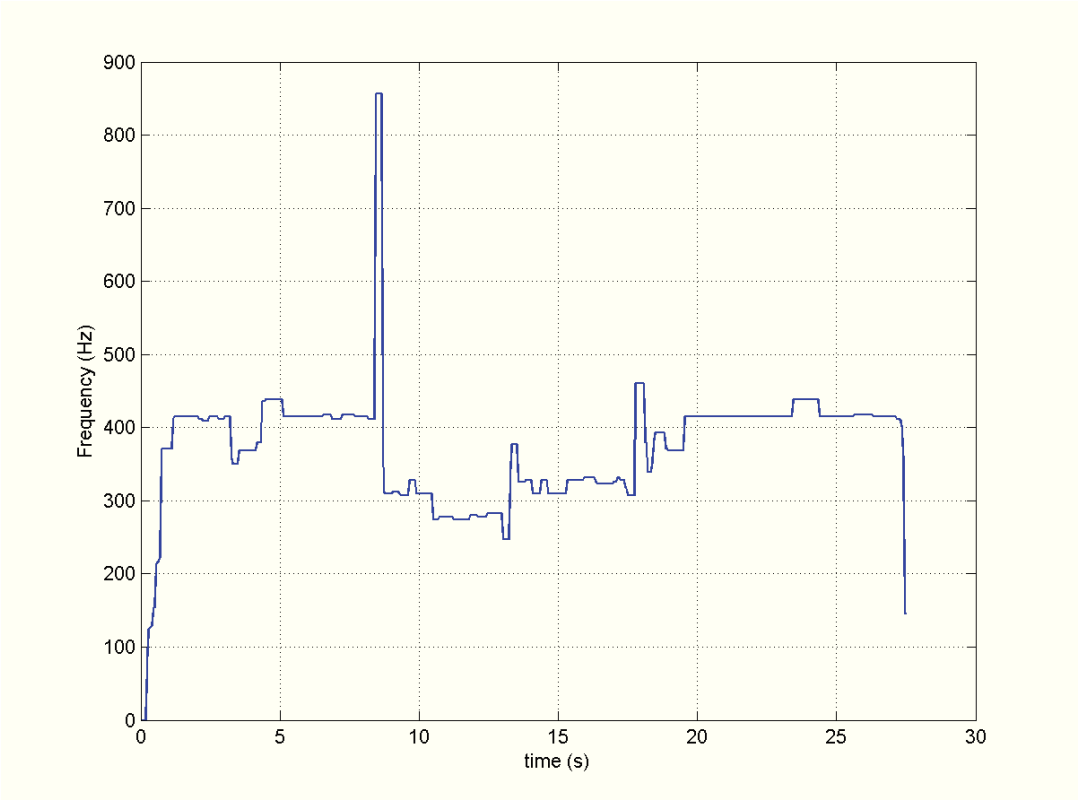

Figura 3.3: Pitch Tracking con HPS

3.1.2 Cepstrum

Se si considera lo spettro di un segnale periodico, (Fig. 3.1) si nota come questo contenga armoniche egualmente spaziate la cui ampiezza decresce all'aumentare della frequenza.

Per questo motivo, il modulo della trasformata di Fourier di un segnale periodico può a sua volta essere considerato un segnale (quasi) periodico con una qualche forma di modulazione d'ampiezza.

La tecnica del Cepstrum sfrutta questa caratteristica dei segnali periodici ed è definito come lo spettro di potenza del logaritmo dello spettro di potenza del segnale.

$$C(\tau) = |\mathcal{F}[log|\mathcal{F}[x(n)]|^2]|^2$$

Calcolando il logaritmo dello spettro di fatto non si sta facendo altro che comprimere l'asse delle ordinate in modo tale da ottenere una periodicità più marcata e ridurre le differenze di ampiezza tra le armoniche (Fig. 3.4).

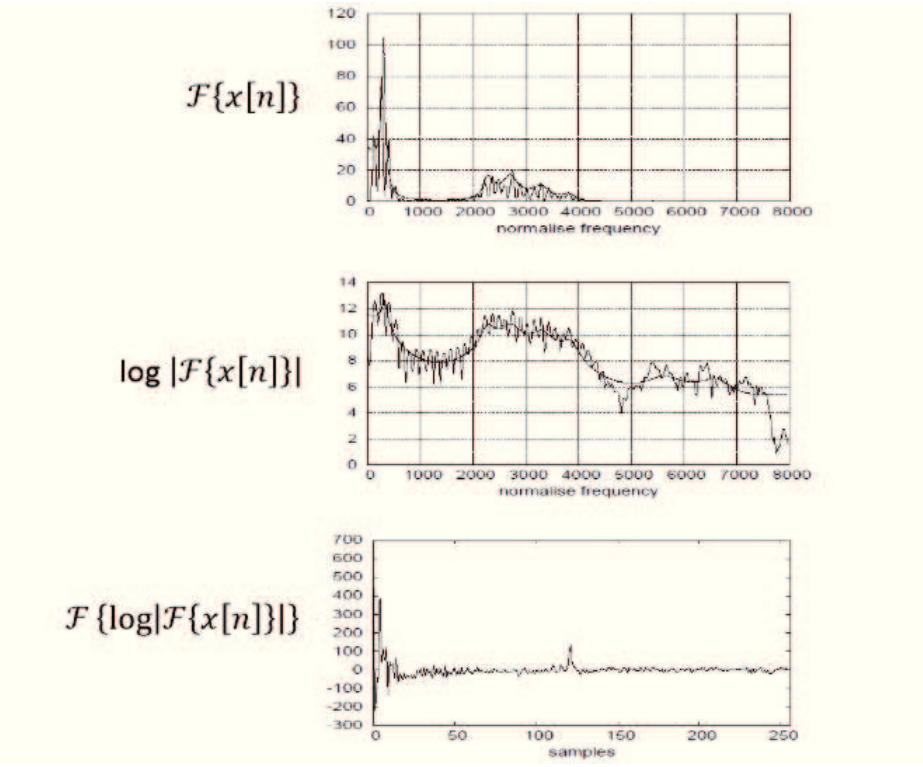

Figura 3.4: Cepstrum

Il segnale che si ottiene trasformando nuovamente secondo Fourier è un segnale nel dominio del tempo che presenterà uno *spike* in corrispondenza

del periodo del segnale originale e delle componenti a bassa frequenza che descrivono l'inviluppo di ampiezza.

Dato che le due componenti non si sovrappongono è possibile separale utilizzando un sempliche filtro.
Il pitch viene quindi riconosciuto eliminando le componenti a bassa frequenza nel dominio del cepstrum e cercando la posizione del valore massimo nella restante parte del vettore.
La frequenza istantanea viene ottenuta con la seguente formula:

$$F0 = Fs/k$$

Dove Fs è la frequenza di campionamento e k è l'indice del valore massimo al tempo τ.

Le varie fasi del Cepstrum sono riassunte nella Fig. 3.5.

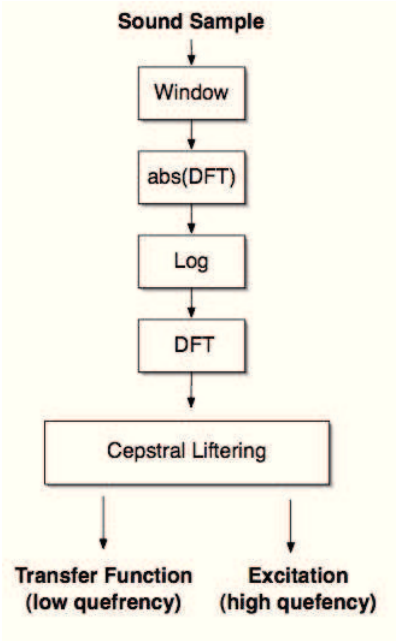

Figura 3.5: Cepstrum

Implementazione in MATLAB

La funzione cepstr prende in ingresso un unico parametro, un segnale nel dominio della frequenza, suddiviso in frame *(signal)* e ne calcola il cepstrum frame per frame *(C)*.
(signal) e *(C)* sono due matrici la cui i-esima riga rappresenta l'i-esimo campione del frame e la j-esima colonna rappresenta il j-esimo frame.

```
function C = cepstr(signal)

C = abs(fft(log(signal.^2)));

C = C(1:end/2);
C = C.^2;
```

Il tracciamento del pitch viene ottenuto cercando per ogni frame la posizione del valore massimo in un determinato range che esclude la zona piu bassa del dominio del Cepstrum (i primi 60 campioni).
I risultati ottenuti con la stessa voce utilizzata in precedenza sono rappresentati nella Fig. 3.6

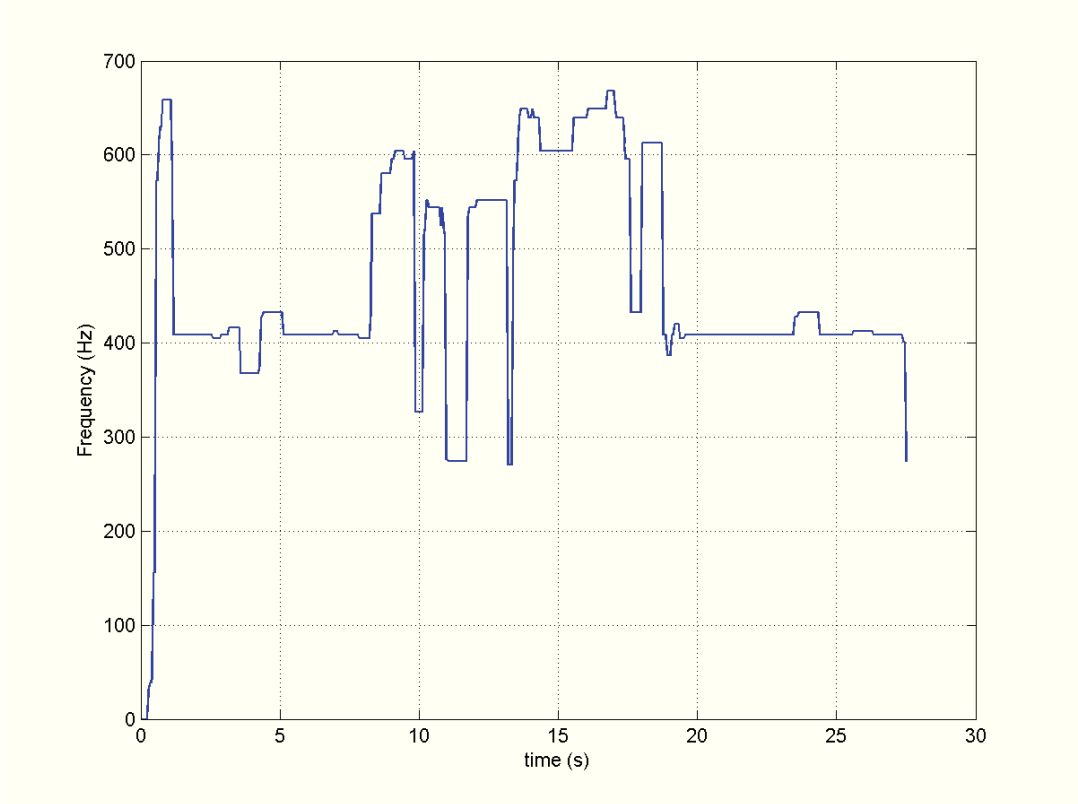

Figura 3.6: Cepstrum

3.1.3 Cepstrum-Biased HPS

I risultati ottenuti utilizzando le due tecniche, si presentano in alcuni punti molto differenti.
É possibile ottenere un pitch tracker più robusto combinando i due approcci, *polarizzando* cioè il Cepstrum con l'HPS (Cepstrum-Biased HPS).

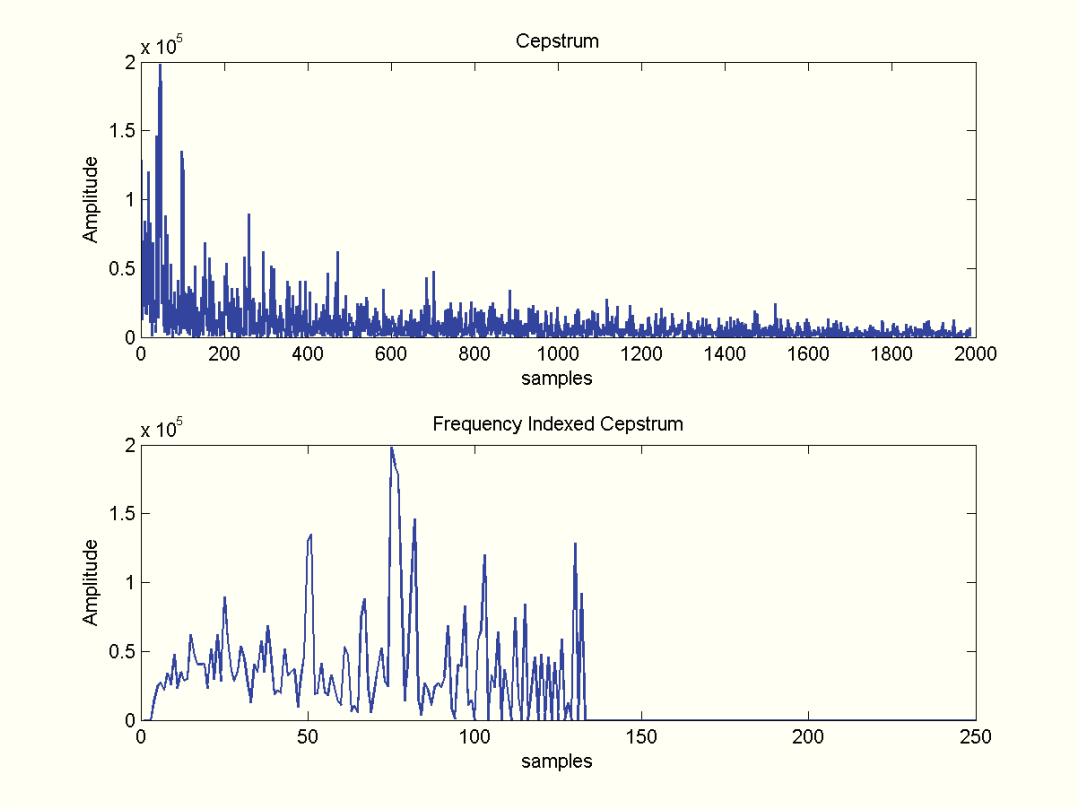

Figura 3.7: Passaggio dal Cepstrum al FIC

Dato che le due funzioni esistono in domini differenti, (Cepstrum nel dominio del tempo e HPS in quello della frequenza) il primo passo per poterli combinare è convertire il Cepstrum affinchè sia indicizzato nel dominio della frequenza. Per fare questo, i valori all'*Indice* $= n$ sono riscritti nel *Frequency Indexed Cepstrum (FIC)* con *Indice* $=$ *floor(N/n)*, dove N è il numero di punti della DFT e $''floor''$ è una funzione che specifica l'intero più grande, minore dell'argomento.

Ovviamente, alcuni valori di n punteranno allo stesso *bin* nel dominio della frequenza; in questo caso si sceglierà il valore maggiore.
Il Frequency Indexed Cepstrum può quindi essere definito come segue:

$$FIC(k) = max[cepstr(n)|0 \leq n \leq N, k = floor(\frac{N}{n})]$$

Una volta calcolato il FIC, il Cepstrum-Biased HPS (CBHPS) può essere definito come la moltiplicazione punto a punto del FIC con il logaritmo dell'HPS:

$$CBHPS(k) = FIC(k) * log(HPS(k))$$

Implementazione in MATLAB

La funzione *cbhps* prende in ingresso 3 parametri:

cepstrum : Matrice contenente il segnale nel dominio del cepstrum, suddiviso in frame. L'i-esima riga rappresenta l'i-esimo campione del frame; la j-esima colonna rappresenta il j-esimo frame.

hps : Matrice contenente il segnale nel dominio della frequenza restituita dalla funzione HPS. L'i-esima riga rappresenta l'i-esimo campione del frame; la j-esima colonna rappresenta il j-esimo frame.

N : numero di punti della FFT.

e restituisce una matrice delle stesse dimensioni delle matrici *cepstrum* e *hps* contenente il CBHPS del segnale originale.

```
function cbhps = cbhps(cepstrum,hps,N)

    winNumb = size(cepstrum,2);
    hop = 60;               %High Pass Liftering
    FIC = zeros(N,winNumb); %Frequency Indexed Cepstrum
    for i=1:winNumb
        for n=hop:N/4
            k = floor(N/n);
            if(FIC(k,i) < cepstrum(n,i))
                FIC(k,i) = cepstrum(n,i);
            end
        end
    end

FIC = FIC(1:end/2+1,:);

cbhps = bsxfun(@times,log(hps),FIC);    %Cepstrum-Biased HPS
```

La frequenza fondamentale viene ottenuta cercando l'indice k che massimizza il CBHPS per ogni frame, moltiplicato per la risoluzione spettrale della DFT.
I risultati ottenuti sono rappresentati in 3.8

3.1.4 Post-processing

Nella Fig. 3.8 si nota come il CBHPS sia un pitch tracker più robusto rispetto ai soli Cepstrum e HPS.
Si nota anche come nonostante questo, il grafico evidenzi degli errori sotto forma di spike; questi errori possono essere eliminati utilizzando a valle del procedimento un *Filtro Mediano*

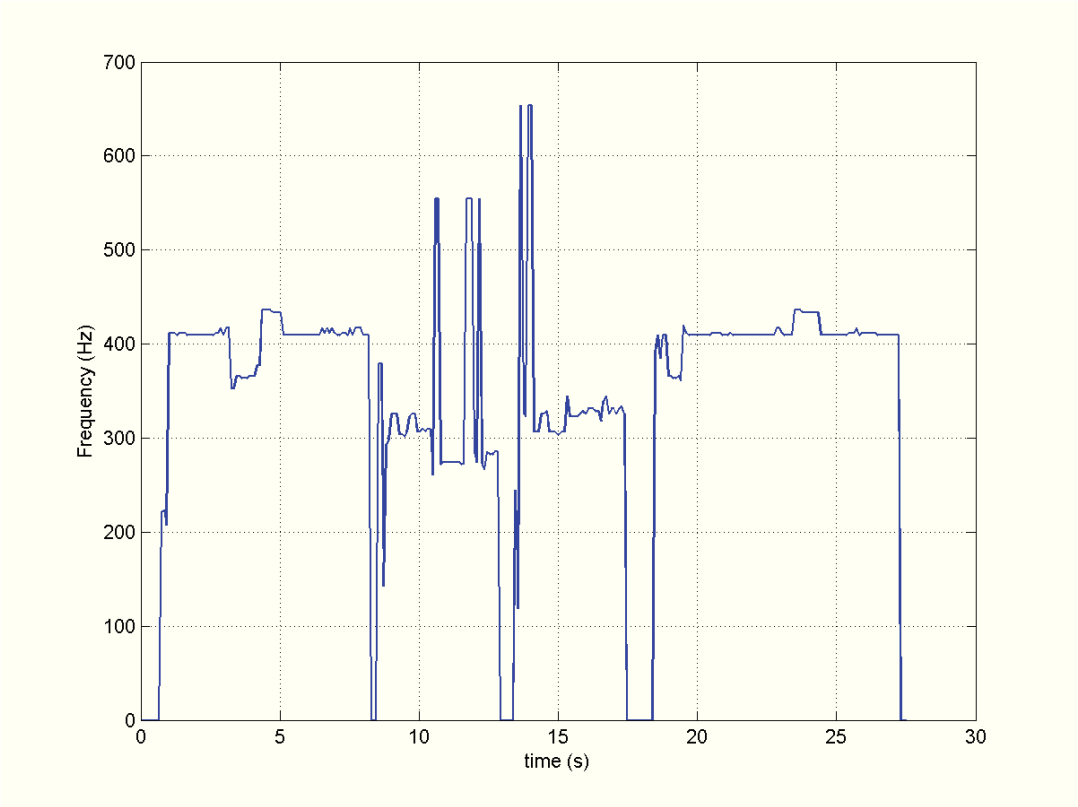

Figura 3.8: Cepstrum-Biased HPS

Filtro Mediano

Un filtro mediano è un filtro digitale che viene utilizzato per eliminare il rumore da immagini o da segnali digitali.

Il principio di funzionamento di questo filtro è di lavorare su singolo campione, andandolo a sostituire con il *valore mediano* dei suoi vicini, rappresentati da una cosiddetta finestra, con al centro il valore da sostituire.

La mediana M di un set di valori, divide il set in due metà della stessa dimensione in modo tale che esistano k valori più piccoli di M e k valori più grandi di M:
Se si considera una lista ordinata di N valori $x(n)$ con N dispari, la mediana M è semplicemente l'elemento centrale $x(\frac{N-1}{2})$.

Per mostrare, utilizzando una finestra di 3 valori, come funziona il filtro mediano, vediamo come si applica a un semplice segnale discreto: $x = [2, 80, 6, 3]$:

Il filtro mediano produce in uscita il segnale y:

$$\text{y[1]} = \text{Median[2 2 80]} = 2$$

$$\begin{aligned} \text{y}[2] &= \text{Median}[2\ 80\ 6] = \text{Median}[2\ 6\ 80] = 6 \\ \text{y}[3] &= \text{Median}[80\ 6\ 3] = \text{Median}[3\ 6\ 80] = 6 \\ \text{y}[4] &= \text{Median}[6\ 3\ 3] = \text{Median}[3\ 3\ 6] = 3 \end{aligned}$$

Quindi:

$$y = [2, 6, 6, 3]$$

Da questo esempio si nota come l'utilizzo del filtro mediano è più indicato nelle applicazioni che implicano la sopplressione di rumore impulsivo (come nel caso della Fig. 3.8) rispetto ad un classico *moving-average filter*, che calcola la media dei valori all'interno della finestra; la mediana infatti è più robusta riguardo ai così detti *outliers*, rispetto alla media. Questo significa che un singolo valore all'interno della lista con un'ampiezza molto grande (outlier) non influenza la mediana. Come risultato di questa caratteristica, dopo l'applicazione di un Median Filter, i contorni del segnale vengono mantenuti e la soppressione del rumore è più efficace. Si veda, per un confronto la Fig. 3.9.

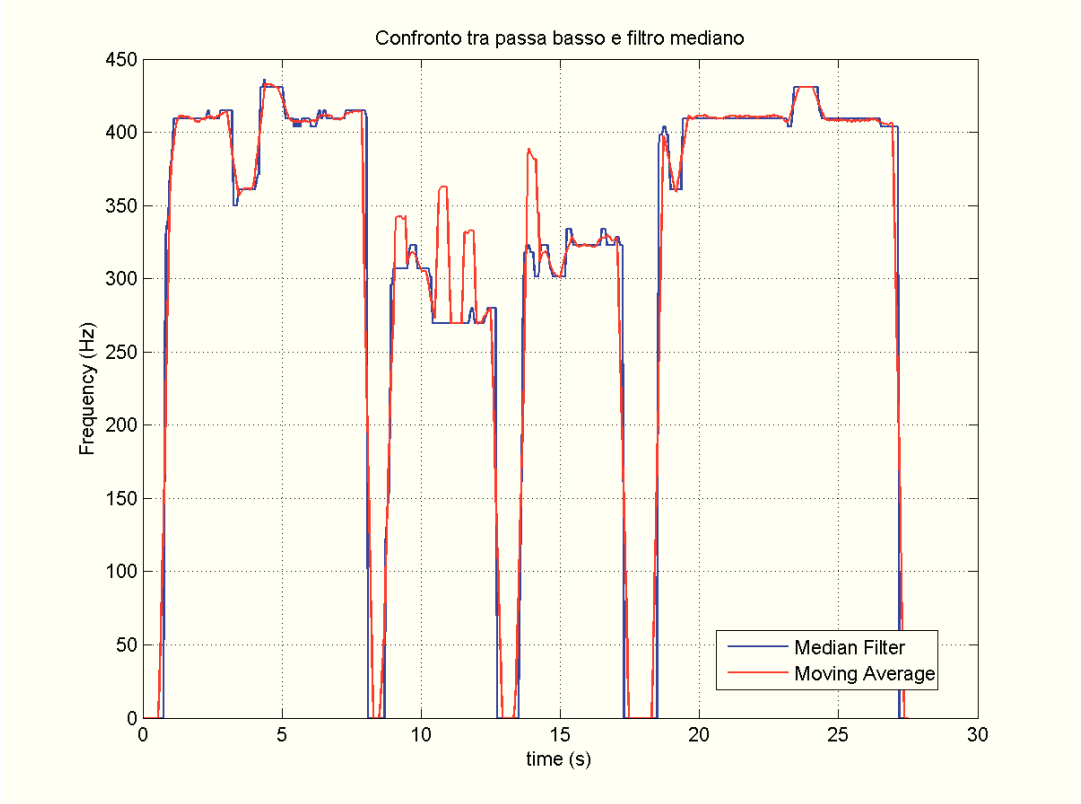

Figura 3.9: Cepstrum-Biased HPS a confronto con Filtro Passa Basso (Moving Average) e filtro mediano

3.1.5 Altre tecniche

Gli algoritmi presentati fin ora operano nel dominio della frequenza; è possibile effettuare un tracciamento del pitch anche utilizzando approcci nel dominio del tempo.
In questa sezione viene presentato il più popolare tra questi, la funzione di Autocorrelazione.

Autocorrelazione

La funzione di autocorrelazione per un segnale discreto è definita come segue:

$$\phi(\tau) = \frac{1}{N} \sum_{n=0}^{N-1} x(n)x(n+\tau)$$

e definisce la misura in cui un segnale è correlato con una versione traslata nel tempo (τ) di se stesso.
Se si fa riferimento alla definizione di segnale periodico data nel capitolo precedente, è facile aspettarsi una forte correlazione quando l'offset τ è pari al periodo fondametale del segnale.

La frequenza fondamentale di un dato segnale può quindi essere stimata cercando il picco con ampiezza maggiore all'interno della funzione di autocorrelazione e facendone l'inverso.

3.2 Segnali Polifonici (Chord Recognition)

In questa sezione verrà esposto il problema del riconoscimento degli accordi all'interno di segnali audio polifonici.
L'algoritmo, implementato in MATLAB, produrrà in uscita una *sequenza di accordi* presentata sotto forma di segmenti non sovrapposti nel tempo, ognuno con un'etichetta che rappresenta il nome dell'accordo.

Il riconoscimento automatico degli accordi ha diversi, impieghi nel campo del Music Information Retrival, dall'identificazione di un brano fino al riconoscimento della sua struttura; in questo elaborato, la sequenza di accordi verrà utilizzata come feature per l'estrazione della tonalità.

3.2.1 Pre-processing del segnale audio

Mid-Side

I segnali audio stereo, sono formati da un canale destro e uno sinistro; quando l'ascoltatore è posizionato esattamente al centro tra due speaker, può distinguere i suoni provenienti dai due canali distinti o quelli provenienti dal

centro. Quando entrambi i canali contengono le stesse informazioni, il segnale viene percepito come centrale (phantom center).

Il phantom center è la parte *Mid* del *Mid-Side*.

Tutto quello che invece è contenuto più o meno ai margini del panorama stereofonico e che viene percepito come *non* centrale è il *Side*. Mid e Side vengono ottenuti a partire dal segnale stereo nel seguente modo:

$$Mid = Canale_L + Canale_R \qquad Side = Canale_L - Canale_R$$

Il Mid in un brano musicale, in genere contiene principalmente informazioni come la melodia, (voce principale) o suoni percussivi (cassa, rullante), che per l'implementazione di un chord recognizer risultano essere superflue se non deleterie.
Tutte le operazioni di seguito esposte, saranno quindi effettuate esclusivamente a partire dal *Side* del brano, che in genere contiene l'accompagnamento.

3.2.2 L'Algoritmo

La Fig. 3.10 fornisce una panoramica dell'algoritmo:

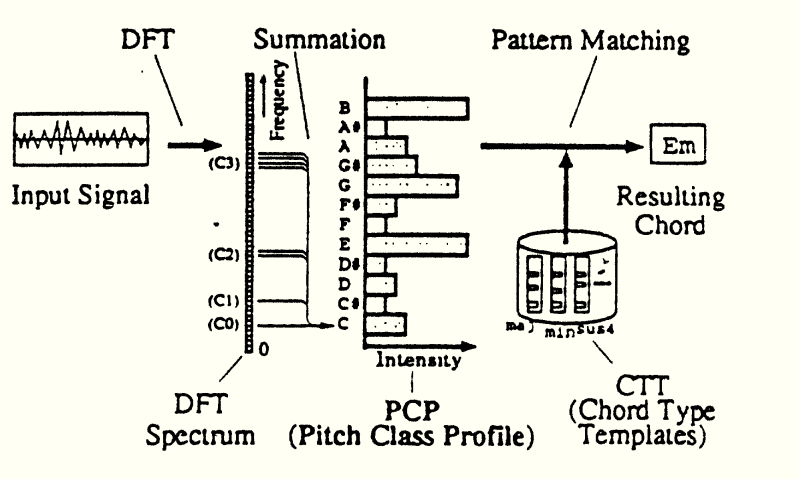

Figura 3.10: Algoritmo di Chord Recognition

Inizialmente il segnale viene trasformato nel dominio della frequenza utilizzando una Short Time Fourier Transform. Sono stati utilizzati un frame-size di 2^{14} campioni con un overlap del 75% e una finestra di Hanning.
Per ogni frame, viene quindi calcolato un *Enhanced Pitch Class Profile (EPCP)*, il quale viene successivamente utilizzato per effettuare un *Template Matching* con una serie di vettori binari (Chord Type Templates [CTT]) che

rappresentano i diversi accordi. L'accordo rilevato per ogni frame è quello che minimizza la *Distanza Euclidea* tra l'EPCP e i CTT.

3.2.3 Chord Type Templates (CTT)

I Chord Type Templates sono dei vettori binari di 12 punti ognuno dei quali corrisponde ad un pitch class nella scala cromatica. Ogni bin sarà posto a 0 o a 1 a seconda che la nota sia presente o meno nell'accordo.
Ad esempio dato che la triade di Do Maggiore è formata dalle note DO(tonica), MI(terza), SOL(quinta), il CTT per questo accordo sarà $[0,0,0,1,0,0,0,1,0,0,1,0]$ con il vettore etichettato nel seguente modo: [A,A#,B,C,C#,D,D#,E,F,F#,G,G#] (Fig. 3.11).

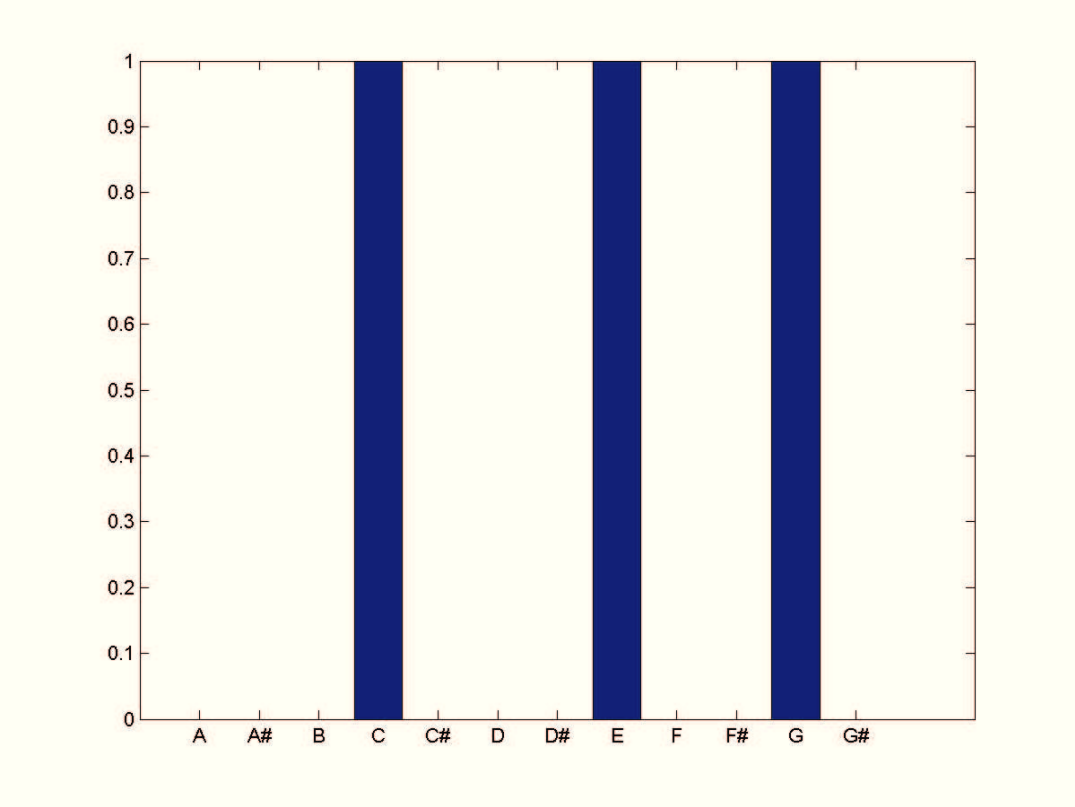

Figura 3.11: CTT di Do Maggiore

Ai fini di ottenere risultati migliori nel template matching, si sceglie di utilizzare templates non strettamente binari, considerando anche gli *overtones* che compongono le note singole dell'accordo, inseriti nel template con un'ampiezza minore. Nella Fig. 3.12 è rappresentato un CTT di Do Maggiore arricchito con la terza armonica di ogni nota che compone l'accordo (la terza armonica si trova una 5^a sopra la fondamentale).

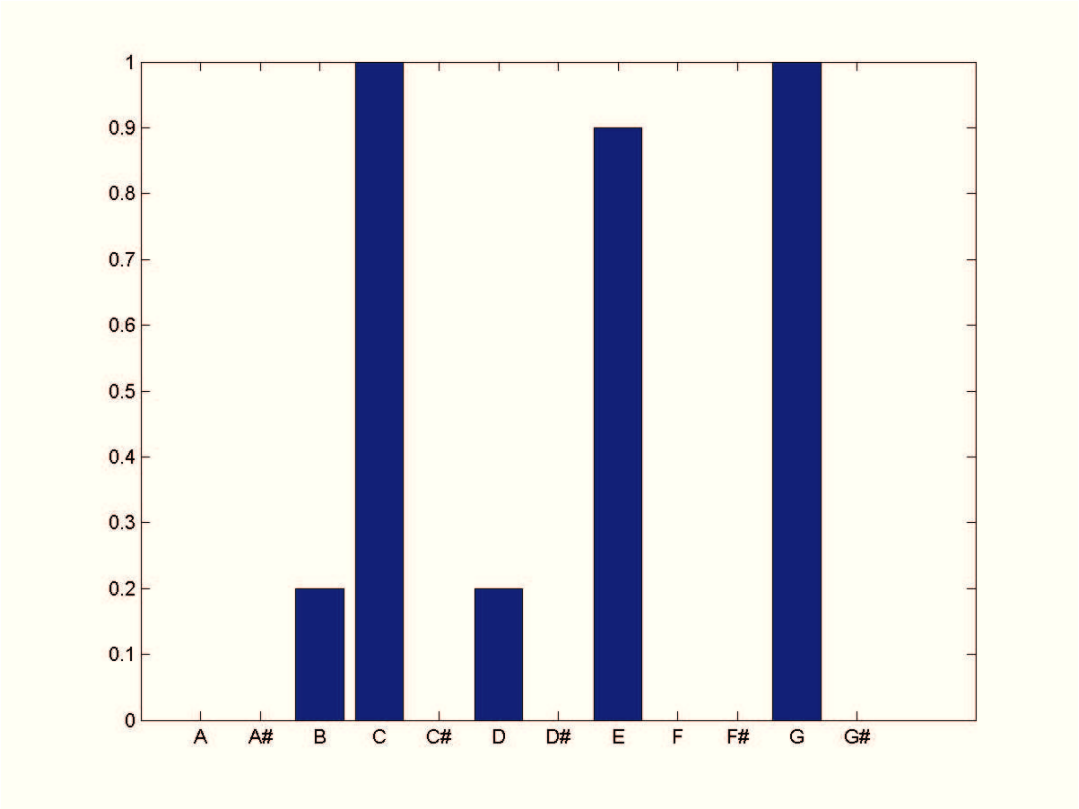

Figura 3.12: CTT di Do Maggiore considerando anche la terza armonica di ogni nota

I template per tutti gli altri accordi maggiori vengono generati operando uno *shift circolare* del vettore di k punti con $k = 1...11$. I CTT per gli accordi minori vengono costruiti allo sesso modo.

3.2.4 Pitch Class Profile

Un Pitch Class Profile (PCP) è un vettore di dodici punti che rappresenta la distribuzione dei dodici semitoni all'interno di un frammento di segnale musicale a prescindere dall'ottava in cui si trovano.

Per ottenere il PCP per ogni frame sono stati seguiti i seguenti passi:

1. Analisi spettrale del singolo frame.
2. **Peak Detection**: vengono considerati solo i picchi presenti all'interno dello spettro del frame.
3. **Calcolo del pitch**: Per ogni picco frequenziale, viene calcolato il pitch come distanza in semitoni dal riferimento a 440 Hz.

4. **Calcolo del modulo 12**: Per gli intervalli ottenuti viene calcolato il modulo 12; in questo modo tutte le note vengono fatte collassare all'interno di un'unica ottava.

5. **Calcolo delle frequenze relative**: Viene calcolato un istogramma di 12 bin che rappresenta la frequenza relativa con cui ciascun *Pitch Name* si presenta.

Peak Detection

Il riconoscimento dei picchi viene effettuato utilizzando la funzione *findpeaks* messa a disposizione da MATLAB.

```
1                [~,idx] = findpeaks(signal(:,i),'MINPEAKDISTANCE',4,
2                'MINPEAKHEIGHT',medpeak(i)*20,'NPEAKS',50);
```

La funzione confronta ogni elemento del vettore $signal(:, i)$ con i suoi vicini; la funzione ignorerà picchi non separati da più di $MINPEAKDISTANCE = 4$ bin e al di sotto di una soglia definita dalla mediana delle ampiezze dello spettro del frame, moltiplicata per un fattore 20. In questo modo, la soglia non è fissata a priori, ma si adatta a quella che è la distribuzione dei picchi all'interno di un determinato frame.

Calcolo del Pitch

Si supponga di avere due note con frequenze f_1 e f_2 il cui rapporto è f_2/f_1. Un'ottava ha un rapporto di $2 : 1$, quindi il numero di ottave tra f_1 e f_2 è (ricordando quanto detto a proposito del rapporto tra frequenza e pitch):

$$n_0 = log_2(f_2/f_1).$$

All'interno del temperamento equabile dove tutti i semitoni hanno lo stesso rapporto frequenziale ($2^{1/12}$), una volta definito il riferimento (440 Hz), per una nota che si torova n semitoni sopra tale riferimento, la frequenza è data da:

$$f_n = 2^{n/12} * 440Hz$$

da cui è possibile ottenere n, la distanza di f_n da 440Hz, espressa in semitoni:

$$n = 12 * log_2(f_n/440) \tag{3.1}$$

Questa è la formula utilizzata per il calcolo del pitch all'intrno del PCP.

Calcolo delle frequenze relative

Dopo aver fatto collassare tutte le note all'interno di un'unica ottava calcolando il modulo 12 dei pitch ottenuti al punto precedente, viene calcolato un istogramma di 12 bin ottenendo il PCP del frame.
Nella Fig. 3.13 è rappresentato il PCP ottenuto su un accordo di Do maggiore:

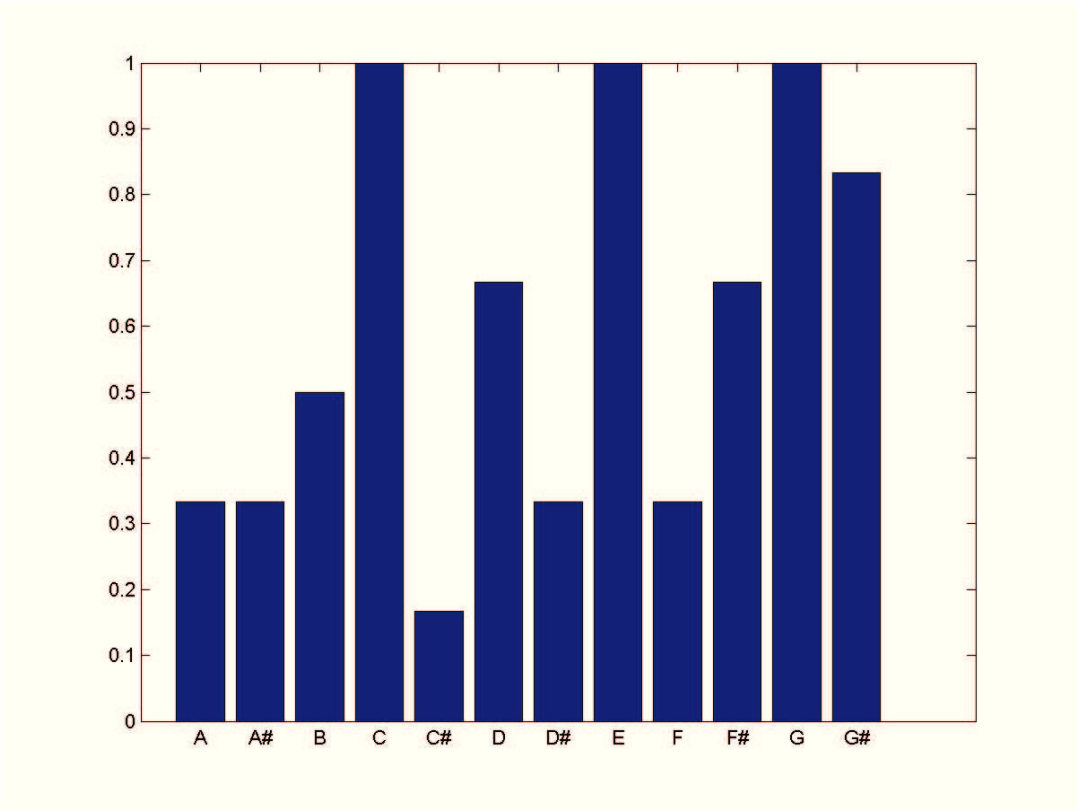

Figura 3.13: PCP di un accordo di DO maggiore

3.2.5 Enhanced Pitch Class Profile

Mettendo a confronto le figure 3.12 e 3.13 si nota come differiscano in modo notevole. Sebbene, nel PCP di Fig. 3.13 i picchi corrispondano alle note dell'accordo di Do Maggiore, la figura mostra la presenza di note su tutta la scala cromatica.
Questo è imputabile al fatto che gli strumenti acustici producono sovratoni oltre alle frequenze fondamentali. Se inoltre si fa riferimento a brani musicali

reali, si nota come questi contengano una molteplicità di strumenti tra cui strumenti percussivi, i quali producono suoni inarmonici.

Un PCP di questo tipo potrebbe causare confusione nel sistema di riconoscimento al momento del template matching.

Il problema può essere risolto facendo in modo che il PCP convenzionale si avvicini maggiormente ai template *quasi* binari esposti in precedenza.
Questo può essere fatto utilizzando l'HPS, il quale permetterebbe di escludere buona parte degli overtones e delle frequenze appartenti ai suoni percussivi.

A tal fine, per ogni frame, viene prima calcolato l'HPS, quindi l'Enhanced Pitch Class Profile (EPCP) vine ottenuto a partire dall'HPS invece che dalla DFT originale.
L'HPS applicato al chord recognition,può essere ottimizzato: decimando lo spettro per interi si rischia di generare energia per pitch classes che non sono contenute nell'accordo; ad esempio la 5^a armonica di un LA3 è DO6#, che non fa parte dell'accordo di LA minore.

Per questo motivo, la decimazione ideale per il chord recognition e quella per potenze di 2. L'HPS viene quindi modificato come segue:

$$HPS(\nu) = \prod_{r=1}^{R} |X(2^r \nu)| \,.$$

L'EPCP viene quindi ottenuto allo stesso modo del PCP convenzionale, partendo però dall'HPS della DFT e non dalla DFT originale. L'EPCP dello stesso accordo di Do Maggiore usato in precedenza è mostrato nella Fig. 3.14 a confronto con il CTT di Do maggiore.

3.2.6 Template Matching

Ogni EPCP ottenuto viene confrontato con i CTT mediante un algoritmo di template matching.
La *measure of fit* scelta per il template matching è la *distanza euclidea* definita come segue:

$$D_{EUC} = \sqrt[2]{\sum_i (x_i - y_i)^2}$$

Dove x è il vettore contenente l'EPCP per un dato frame e y è un CTT. La distanza viene calcolata per ogni frame, tra l'EPCP e i CTT di tutte le triadi maggiori e minori.

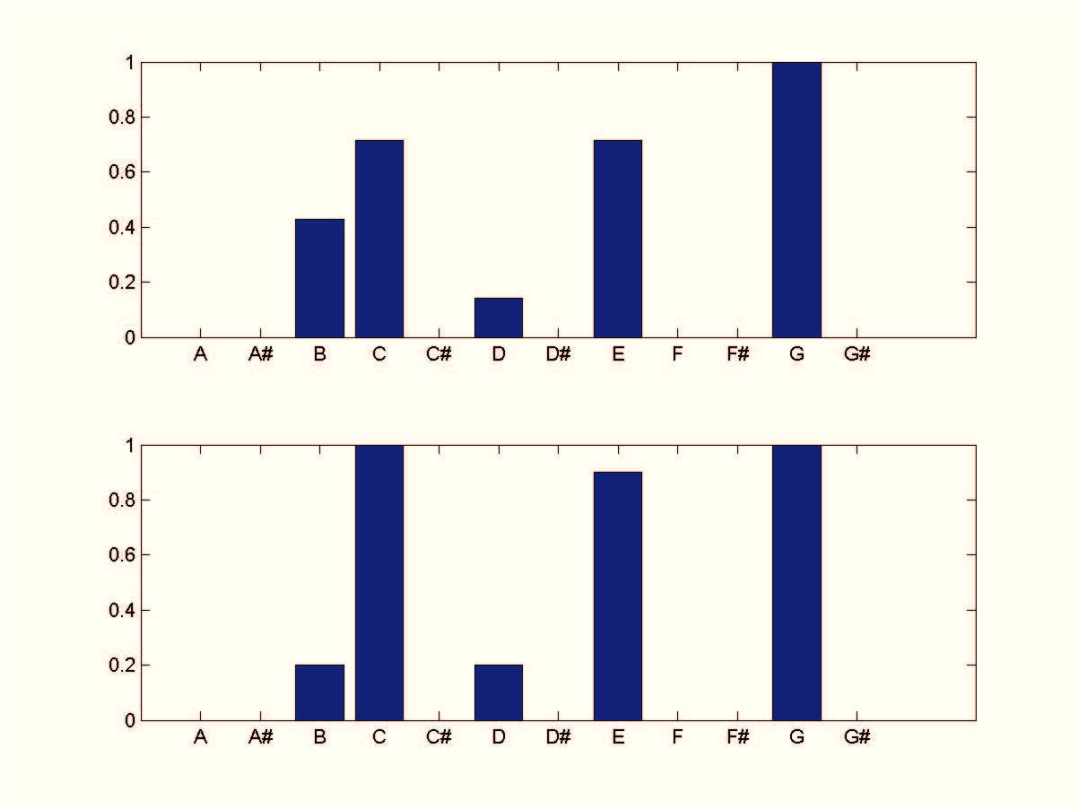

Figura 3.14: EPCP di un accordo di DO maggiore (in alto) a confronto con il template di do maggiore arricchito con la terza armonica di ogni nota (in basso)

L'accordo che minimizza la distanza euclidea sarà l'accordo riconosciuto per quel frame.
Nella Fig. 3.15 sono mostrati i risultati ottenuti sui primi 45 secondi del brano Candy di Paolo Nutini.

3.2.7 Post-processing

Come si nota nella Fig. 3.15, i risultati presentano degli errori di tipo impulsivo, simili a quelli riscontrati nel pitch tracking.
Questi possono essere corretti utilizzando un filtro mediano; sebbene non abbia senso utilizzare un filtro di questo tipo su di un'informazione simbolica, osservando il tipo di risultati che l'algoritmo fornisce, è plausibile considerare la moda sempre uguale alla mediana, quindi possiamo usare un filtro mediano come selettore della moda.

La Fig. 3.16 mostra i risultati ottenuti applicando un filtro mediano con una finestra di 13 punti.

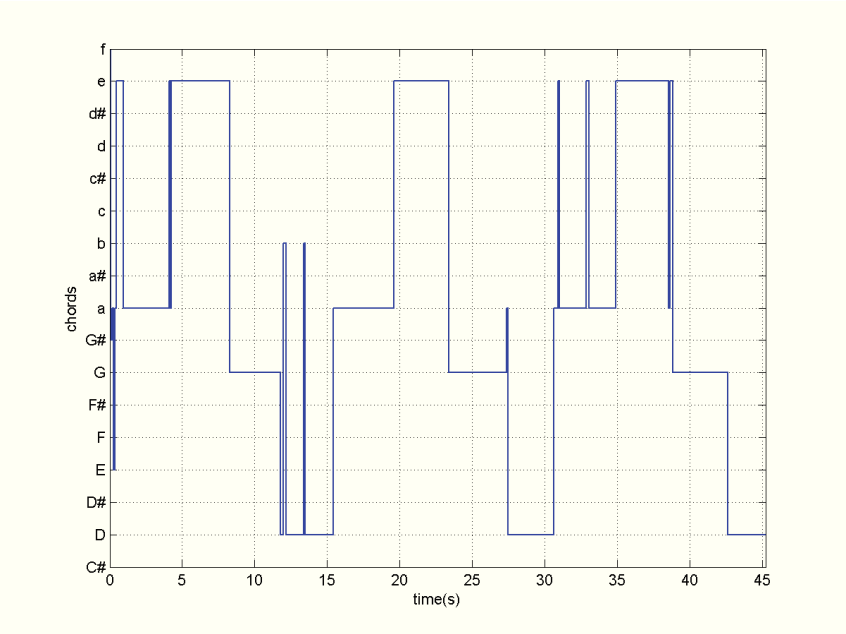

Figura 3.15: Accordi riconosciuti nei primi 45 secondi di Candy di Paolo Nutini

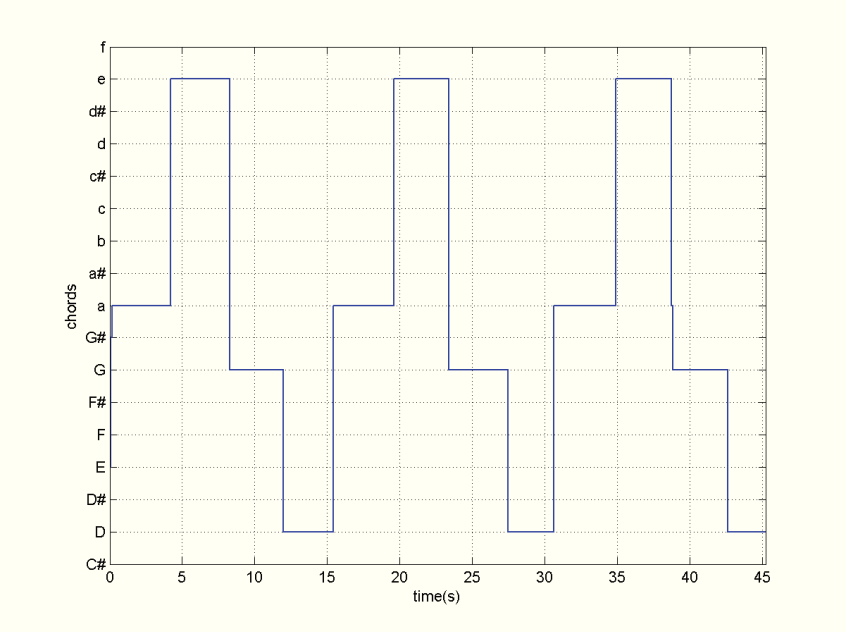

Figura 3.16: Accordi riconosciuti nei primi 45 secondi di Candy di Paolo Nutini dopo l'applicazione di un filtro mediano

Capitolo 4

Estrazione della tonalità

Nella prima e seconda sezione di questo capitolo verranno illustrati i procedimenti utilizzati per stimare la tonalità nei segnali audio monofonici e polifonici. A tal fine verranno utilizzate le features estratte con gli algoritmi esposti nel capitolo precedente.

4.1 Segnali monofonici

La linea melodica estratta nell'esempio di Fig. 3.8 è rappresentata con un grafico della frequenza in funzione del tempo.
Per estrarre la tonalità utilizzeremo un PCP generato sulla base di quelle che sono le frequenze trovate dal pitch tracker. Il primo passo è quello di convertire le frequenze in pitch class. Per farlo utilizzeremo l'equazione 3.1; ogni nota sarà rappresentata come la distanza in semitoni dal riferimento a $440Hz$.

Il PCP viene quindi ottenuto calclolando prima il modulo 12 degli intervalli e successivamente la frequenza relativa con cui ogni intervallo si presenta. Il PCP relativo al segnale utilizzato nella sezione 3.1 è mostrato nella Fig. 4.1

L'estrazione della tonalità avviene con un template matching tra il PCP e i key/mode template delle 24 tonalità (12 maggiori e 12 minori).

4.1.1 Creazione dei key/mode Template

I key/mode template sono dei vettori di 12 punti che rappresentano la distribuzione dei pitch class all'interno di ciascuna delle tonalità maggiori o minori.
Vengono creati sulla base dei 3 accordi principali della tonalità ovvero quelli costruiti sul I(tonica) IV(sottodominante) e V(dominante) grado della scala. Per ogni tonalità maggiore e minore, ciascuna triade viene scomposta; il template viene quindi creato sommando il contributo di ciascuna nota che

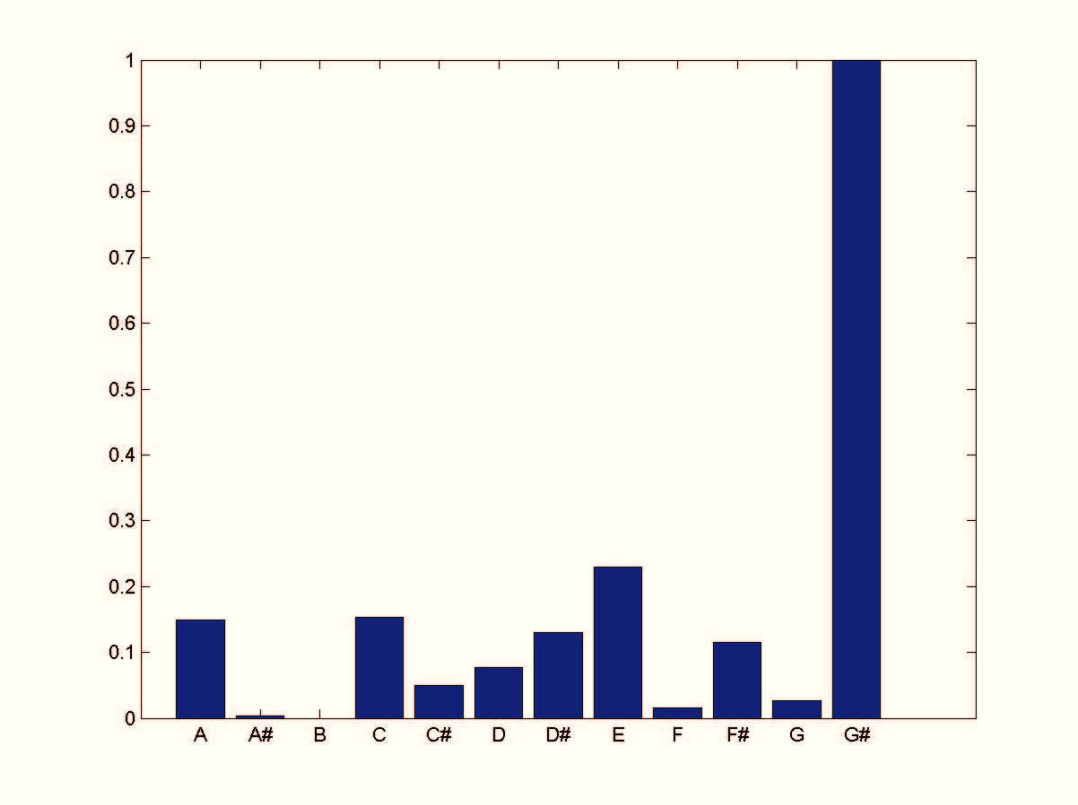

Figura 4.1: PCP linea melodica

compone ognuno dei tre accordi più, per ognuna di esse, la 3^a armonica, ma solo se facente parte della tonalità d'impianto.
Quindi, ad esempio, in tonalità di DO maggiore, verranno scomposti gli accordi di DO maggiore, FA maggiore e SOL maggiore; per ognuna delle note dei tre accordi si aggiungerà la 3^a armonica, eccetto che per il SI, la cui 3^a armonica sarebbe un FA# che non fa parte della tonalità.
Il template viene quindi normalizzato affinchè l'ampiezza massima sia 1. Nella Fig. 4.2 sono mostrati i key/mode template della tonalità di Do maggiore e della sua relativa minore (La).

In modo analogo al caso del Chord Recognition, la tonalità stimata sarà quella che minimiza la distanza euclidea tra il PCP e ciascuno dei key/mode Template.

Il grafico in Fig. 4.3 rappresenta i risultati ottenuti sulla linea melodica di Fig. 3.8.

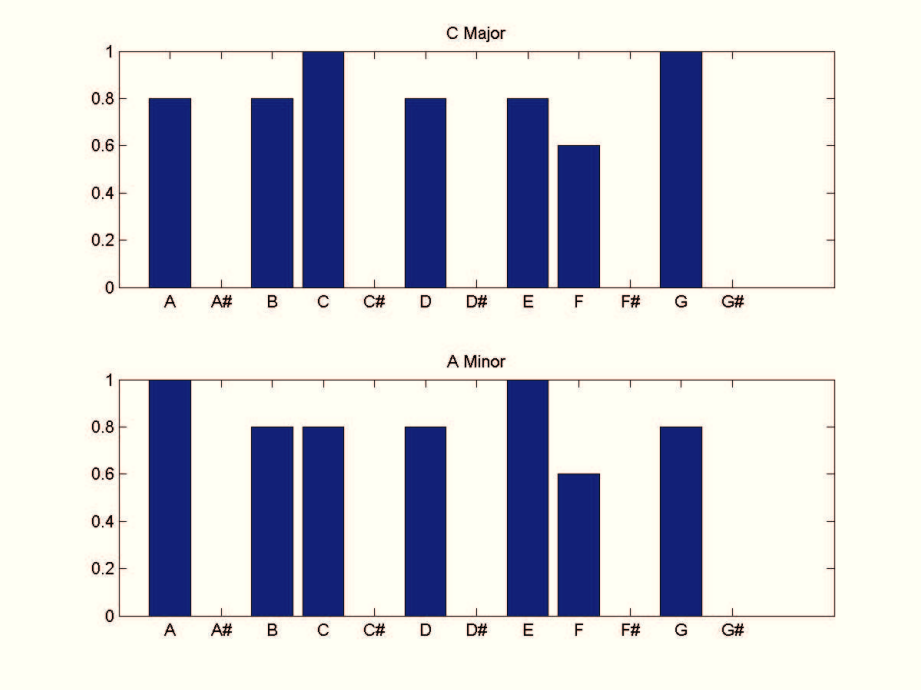

Figura 4.2: Key/mode Template di DO maggiore e LA minore

4.2 Segnali polifonici

Per quanto riguarda i segnali polifonici, il procedimento è analogo.
Viene creato un PCP a partire da una sequenza di accordi; ogni accordo rilevato dall'algoritmo di chord recognition viene scomposto e il PCP viene creato sommando il contributo di ciasuna nota. Segue una normalizzazione del vettore.

La tonalità stimata sarà anche in questo caso, quella che minimizza la distanza euclidea tra il PCP e i key/mode Templates.

Nella Fig. 4.4 sono rappresentati i risultati ottenuti sul brano Candy di Paolo Nutini a partire dagli accordi rinconosciuti nel capitolo precedente.

4.2.1 Riconoscimento delle modulazioni

Questo modo di operare consente di stimare la tonalità considerando il segnale nella sua interezza; quindi se il brano presenta delle modulazioni (cambio di tonalità), queste non solo non verranno riconosciute, ma probabilmente la stima della tonalità per quel segnale sarà errata.
Per ovviare a questo problema è possibile applicare l'algoritmo di estrazio-

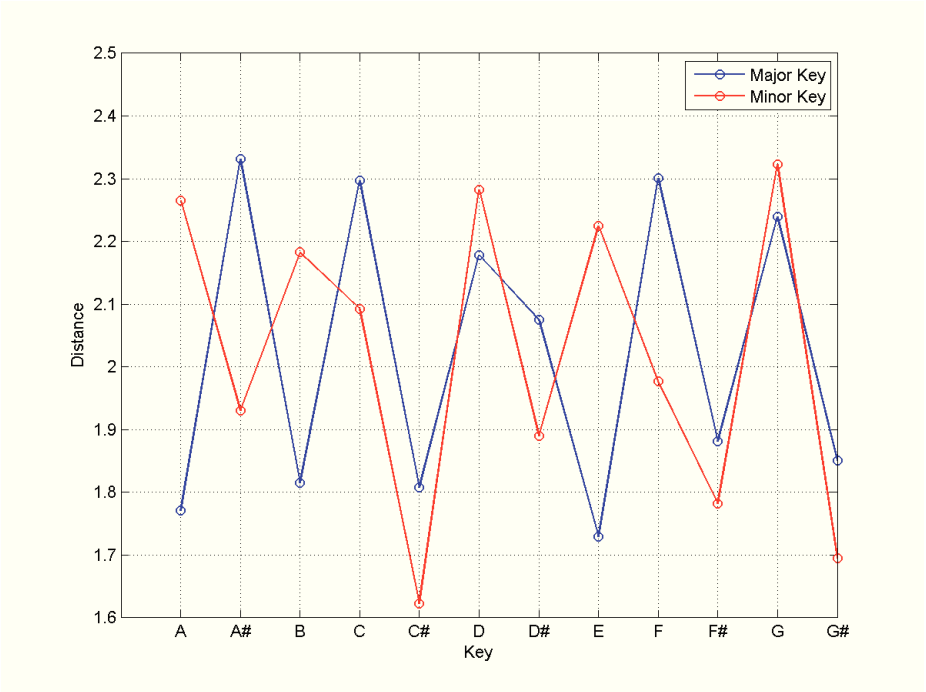

Figura 4.3: Distanza del PCP da ciascuna delle tonalità maggiori e minori (segnale monofonico). La tonalità stimata è DO# minore

ne della tonalità, su porzioni del segnale ottenute attraveso una finestratura temporale.

4.2.2 Calcolo della dimensione della finestra

La dimensione della finestra è di fondamentale importanza per il corretto funzionamento dell'algoritmo. Finestre troppo piccole porterebbero a considerare una sequenza di accordi non sufficiente per riconoscere la tonalità; al contrario, finestre di grosse dimensioni renderebbero impreciso il riconoscimento della modulazione.

L'idea è quella di stimare la durata media di un accordo; una volta ottenuta questa informazione è possibile scegliere dimensioni della finestra che contengano mediamente n accordi.
La dimensione della finestra viene calcolata con il seguente codice:

```
1          change = abs(diff(chords));
2
3          [~,locs] = findpeaks(change);
4          n_accordi = 4;
5
```

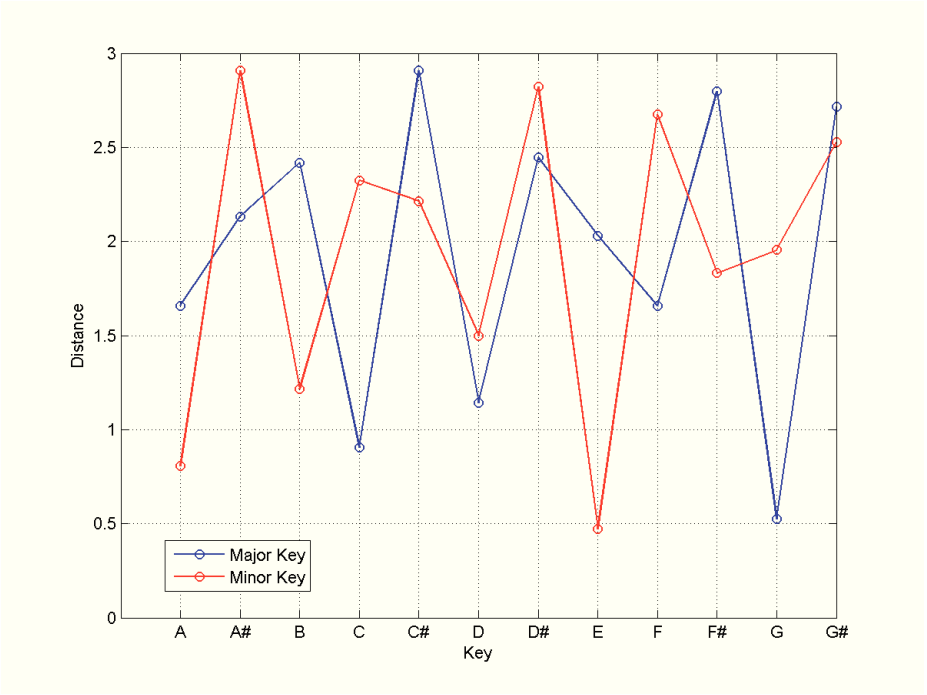

Figura 4.4: Distanza del PCP da ciascuna delle tonalità maggiori e minori (Segnale Polifonico). La tonalità stimata è Mi minore

```
6           framesize2 = round(mean(diff(locs))) * n_accordi;
7
8           if(framesize2 > length(chords))
9               framesize2 = length(chords)/2;
10          end
```

chords è il vettore contenente gli accordi trovati dall'algoritmo di chord recognition (Fig. 3.16).
change presenterà un picco in corrispondenza di ogni cambio di accordo (Fig. 4.5). *change* è, infatti, il differenziale di *chords* (che contiene dati simbolici, ma in questo caso viene trattato come una funzione); siamo infatti interessati allo stato binario *change != 0*.

Quello che si vuole ottenere è la distanza media tra un picco e il successivo, che rappresenta la durata media di un accordo.

locs rappresenta gli istanti di tempo (in campioni) ai quali si ha un cambio di accordo.
La durata media in campioni di un accordo, quindi, è pari alla media delle differenze tra questi intervalli, arrotondata all'intero più vicino. *framesize2* sarà quindi pari a questo valore moltiplicato per 4 (ipotizzando di voler avere almeno 4 accordi per ogni frame).

Il vettore *chords* viene quindi suddiviso in frames di *framesize2* campioni

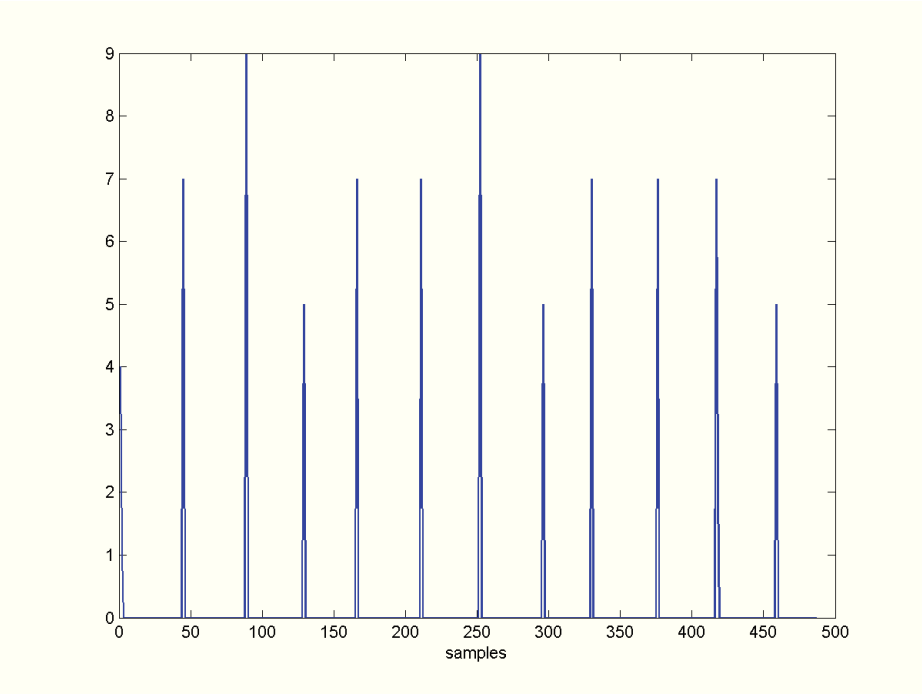

Figura 4.5: Chord change

con un overlap del 50%. Per ognuno di questi viene applicato l'algoritmo per il riconoscimento della tonalità che restituisce la tonalità trovata per quel frame.

Nelle Fig. 4.6 e 4.7 sono rappresenati rispettivamente gli accordi trovati su un brano che da Do Maggiore modula a La Maggiore, e le tonalità stimate con l'algoritmo appena esposto.

4.2.3 Post-processing

Anche in questo caso il segnale di Fig. 4.7 è stato processato utilizzando un filtro mediano.
In questo caso, l'ordine del filtro (numero di campioni della finestra) non è fisso come nelle precedenti applicazioni, ma vine calcolato secondo la seguente formula:

$$filtOrder = 2 * floor(numFrames * 0.7/2) + 1$$

che trova il più grande intero dispari minore del 70% del numero dei frame. Si presuppone, infatti, che i cambi di tonalità, se presenti, avvengano entro intervalli di tempo abbastanza lunghi; ecco quindi perchè ha senso utilizzare un filtro mediano con un ordine relativamente ampio che si *adatta* alla grandezza del segnale.

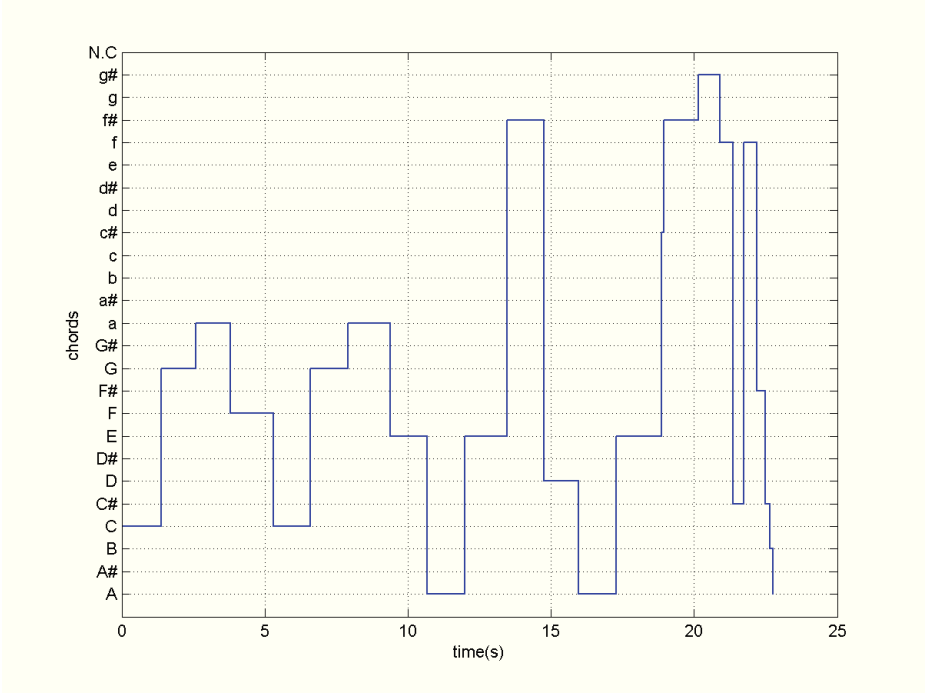

Figura 4.6: Accordi riconosciuti

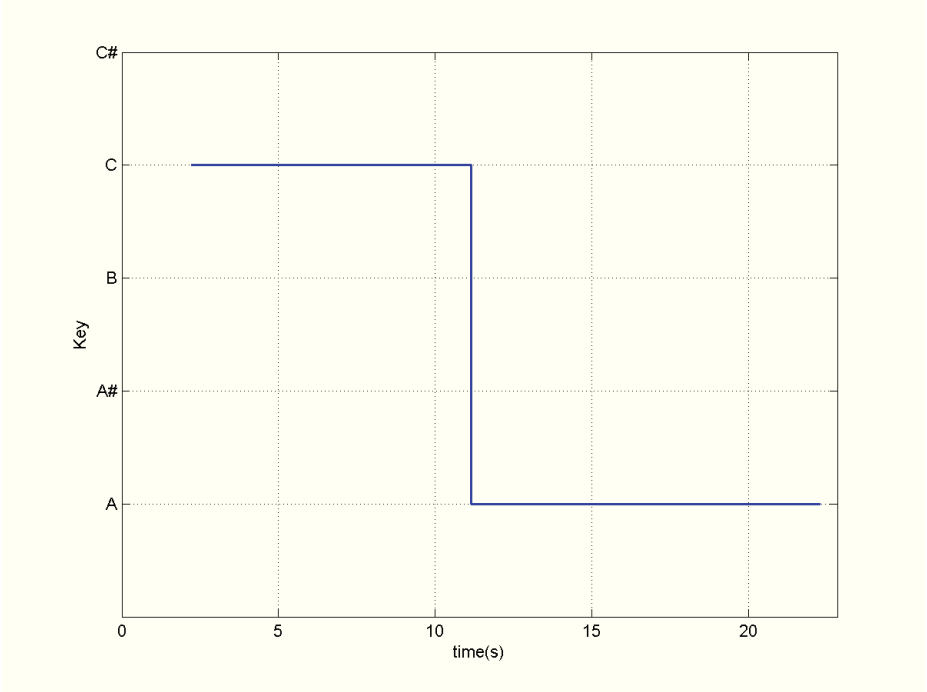

Figura 4.7: Riconoscimento della modulazione

Capitolo 5

Test e presentazione dei risultati

5.1 Segnali Polifonici

Per la valutazione dell'algoritmo di riconoscimento degli accordi presentato nella seconda sezione del 3° capitolo, è stato utilizzato un dataset di 10 brani POP. I risultati vengono confrontati con quelli ottentuti sfruttando le funzioni del *MIR Toolbox* [3], e validati rispetto alla successione di accordi di riferimento rappresentata dalle annotazioni di *Giorgi, Zanoni, Sarti, Tubaro* presentate in [10].(Fig 5.1)

Il punteggio per ogni brano è calcolato come il rapporto tra la lunghezza (in campioni) degli accordi riconosciuti correttamente e la lunghezza totale del brano:

```
1          difference = abs(reference - accordo);
2          correct = sum(difference == 0);
3
4          score = correct/length(accordo);
```

Un esempio sul calcolo del punteggio è presentato nella Fig 5.2.
Tutti i risultati sono riassunti nella tabella 5.1.

5.2 Segnali monofonici

Per quanto riguarda i segnali monofonici, i test sono stati effettuati su un dataset di 10 segnali audio contenenti linee melodiche ottenute casualmente utilizzando la funzione *random* messa a disposizione da un sequencer midi.

Ogni linea melodica è stata sottoposta all'algoritmo di *pitch tracking* esposto nel Capitolo 3; i risultati sono stati confrontati con quelli ottenuti

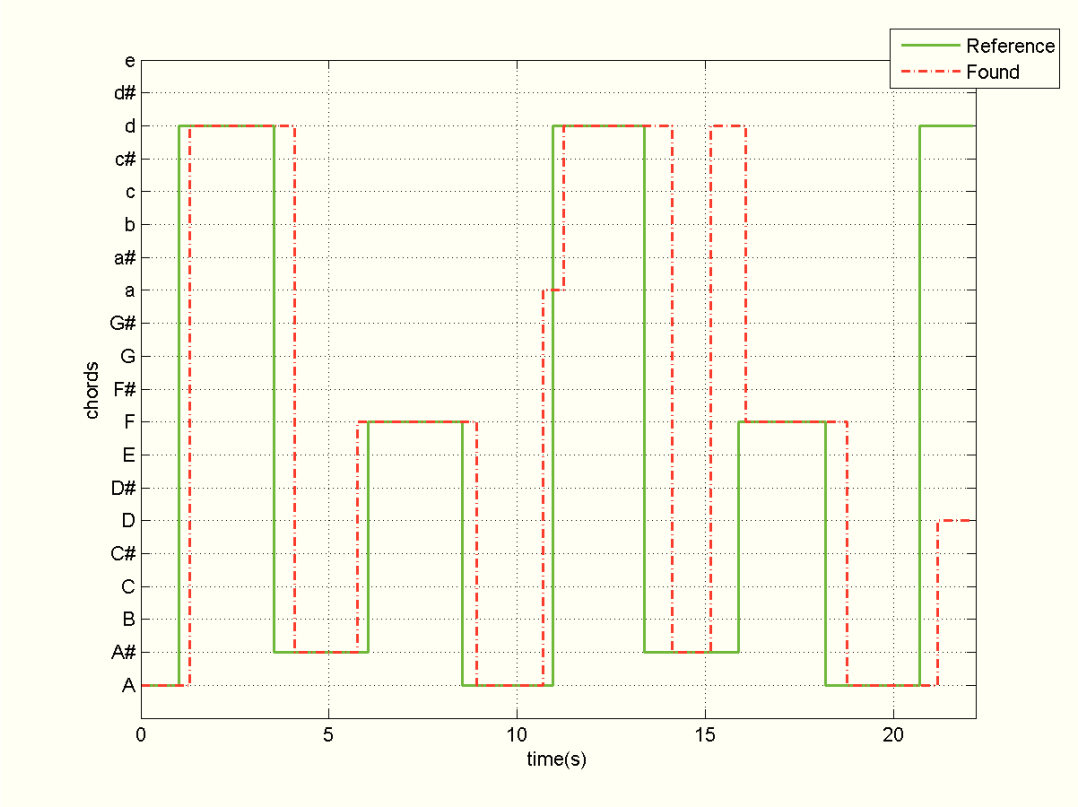

Figura 5.1: Confronto dell'algoritmo con la successione armonica di riferimento

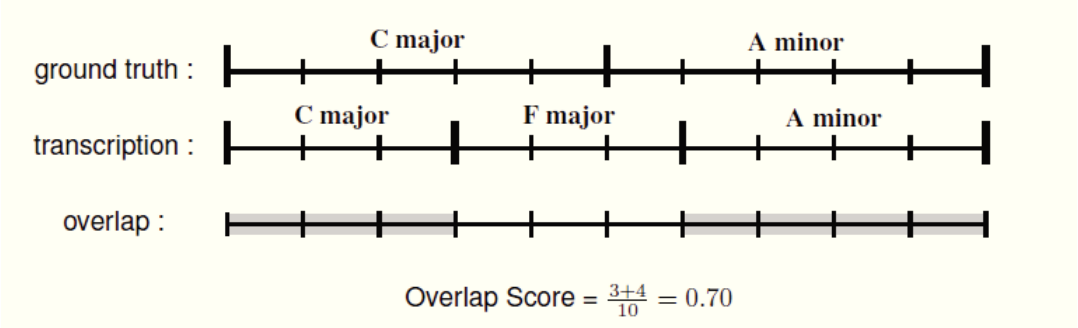

Figura 5.2: Esempio di calcolo del punteggio

dal pitch tracker messo a disposizione dal *MIR Toolbox*.

Entrambi i risultati vengono validati rispetto alla linea melodica di riferimento, ottenuta asscociando ad ogni *note name* il rispettivo valore in frequenza. (Fig 5.4).

	% ChordRec	% MIR Toolbox
Brano 1	0.94	0.57
Brano 2	0.95	0.59
Brano 3	0.91	0.61
Brano 4	0.96	0.74
Brano 5	0.95	0.58
Brano 6	0.9	0.57
Brano 7	0.97	0.65
Brano 8	0.92	0.68
Brano 9	0.89	0.75
Brano 10	0.93	0.60
Total	**93%**	**64%**

Tabella 5.1: Tabella riassuntiva risultati riconoscimento degli accordi

Per entrambi gli algoritmi sono stati utilizzati gli stessi paramteri:

1. Framesize della STFT = 16384 campioni
2. Overlap = 14336 campioni

Un esempio dei risultati ottenuti è rappresentato nella Fig 5.3

Gli algoritmi vengono valutati sulla base della percentuale di note riconosciute all'interno di ogni segnale audio utilizzando lo stesso metodo impiegato nei segnali polifonici: per ogni campione viene calcolata la differenza tra la frequenza trovata e quella di riferimento; una nota è riconosciuta correttamente se questa differenza è minore o uguale alla risoluzione frequenziale della DFT.

Tutti i risultati sono riassunti nella tabella 5.2.

5.3 Riconoscimento della tonalità

I test per il riconoscimento della tonalità sono stati effettuati su due differenti dataset, uno di musica classica, l'altro di musica POP, entrambi contenenti 10 brani.

Anche in questo caso l'algoritmo presentato in questo elaborato è stato confrontato con quello messo a dispozione dal *MIR Toolbox*.

Per quanto riguarda i brani di musica classica la tonalità stimata dagli algoritmi è confrontata con quella riportata nel nome del brano stesso (es *Mozart, Piano Sonata in C Major*).

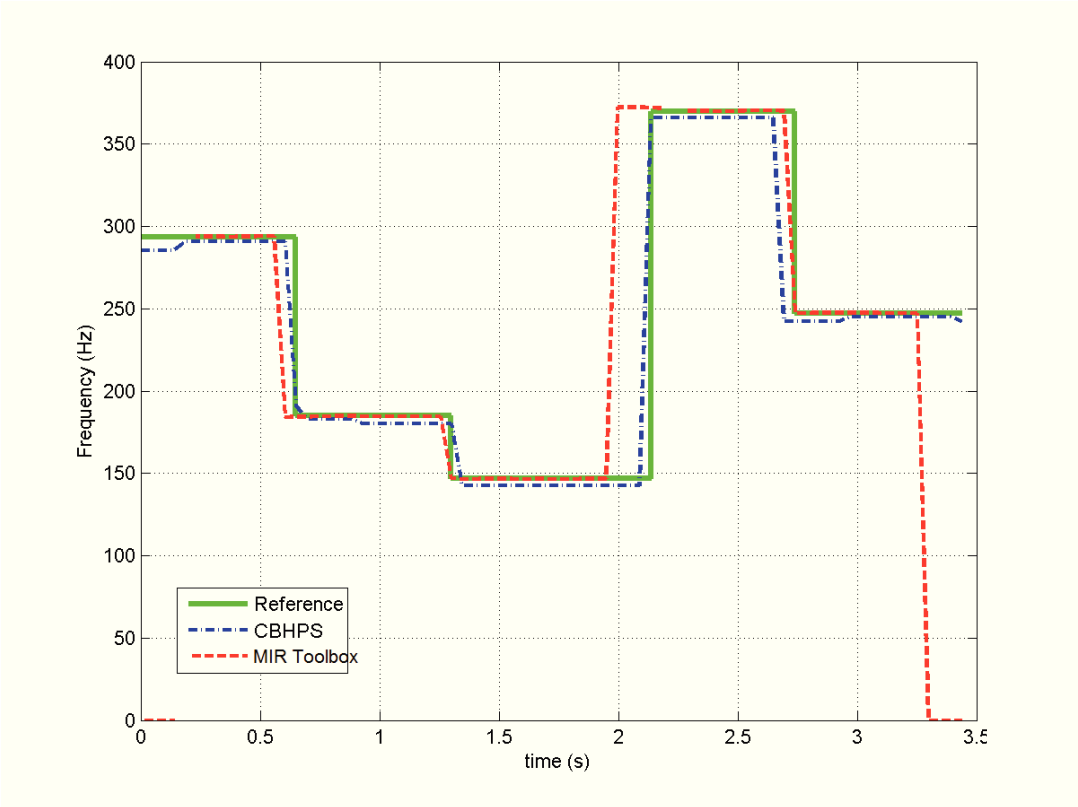

Figura 5.3: Confronto tra i due algoritmi

Per quelli di musica POP, invece, vengono ancora una volta utilizzate le trascrizioni contenute in [10].

5.3.1 Metodo di valutazione

Il metodo di valutazione adottato è lo stesso utilizzato nel *MIREX 2014: Audio Key Detection*: viene determinato quanto ogni tonalità identificata sia vicina alla tonalità reale.
Le tonalità stimate saranno considerate più o meno vicine a quella reale se presentano con essa una delle seguenti relazioni: distanza di una quinta giusta, relativa maggiore o minore e parallela maggiore o minore.

L'assegnazione dei punti fa riferimento alla tabella 5.3

I risultati per i segnali polifonici sono rappresentati nella tabella 5.4, quelli per i segnali monofonici nella tabella 5.5.

Intrument	% CBHPS	% MIR Toolbox
Piano	0.97	0.98
Organ	0.92	0.59
Rhodes	0.95	0.97
Voice	0.94	0.96
Flute1	0.95	0.85
Flute2	0.84	0.75
Flute3	0.85	0.75
Flute4	0.75	0.77
Flute5	0.85	0.79
Flute6	0.50	0.61
Total	**85%**	**80%**

Tabella 5.2: Tabella riassuntiva risultati Pitch Tracking

Relazione	Punti
Stessa	1
Quinta Giusta	0.5
Relativa maggiore/minore	0.3
Parallela maggiore/minore	0.2
Altro	0

Tabella 5.3: Procedura di valutazione dell'algoritmo per il riconoscimento della tonalità

Segnali Polifonici				
	Dataset POP		**Dataset Classica**	
	Key Recognition	**MIR Toolbox**	**Key Recognition**	**MIR Toolbox**
1	1	1	1	1
2	1	0	1	1
3	1	0.3	0	1
4	1	0	0.3	1
5	1	0.3	1	1
6	1	1	0.3	0.3
7	1	1	1	0.3
8	1	1	0.2	0.2
9	1	1	1	0
10	0	0.2	0.2	1
Media	**90%**	**58%**	**59%**	**68%**

Tabella 5.4: Tabella riassuntiva risultati per il riconoscimento della tonalità nei segnali polifonici

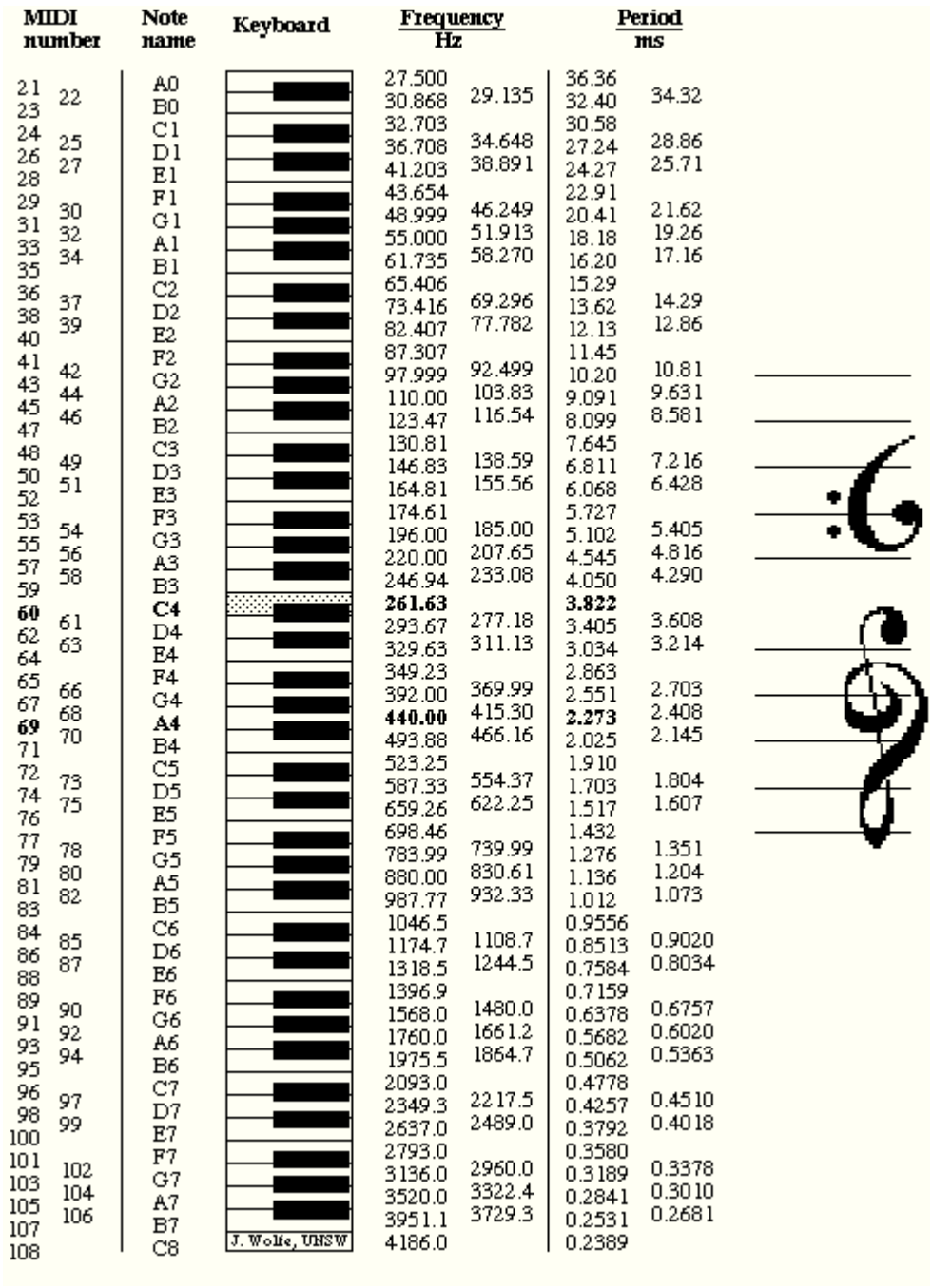

Figura 5.4: note names e rispettive frequenze

Segnali Monofonici		
	CBHPS	**MIR Toolbox**
1	1	1
2	1	1
3	1	1
4	1	1
5	1	1
6	0.3	1
7	1	1
8	1	1
9	1	1
10	0.3	0.3
Media	**86**%	**93**%

Tabella 5.5: Tabella riassuntiva risultati per il riconoscimento della tonalità nei segnali monofonici

Capitolo 6

Conclusioni e sviluppi futuri

6.1 Conclusioni

Confrontando i risultati dell'algoritmo per il riconoscimento della tonalità presentato in questo elaborato, con quello messo a disposizione dal MIR Toolbox, si nota come il MIR Toolbox raggiunga risultati migliori sui segnali monofonici.
Per quanto riguarda quelli polifonici, facendo riferimento al dataset POP, il MIR Toolbox presenta risultati corretti solo nel 58% dei casi. Il restante 42% è composto da errori che in prevalenza consistono nell'aver stimato una tonalità che è la relativa maggiore/minore rispetto alla tonalità reale. Per lo stesso dataset, l'algoritmo presentato in questo elaborato presenta risultati corretti nel 90% dei casi.

Il motivo di questa differenza, è da ricercarsi nella natura dei due algoritmi; mentre il MIR Toolbox fa una stima della tonalità ottenendo il PCP direttamente da una trasformazione nel dominio della frequenza, l'algoritmo presentato in questo elaborato, lo ottiene a partire da un'informazione simbolica, gli accordi. Questo permette di ottenre PCP più *puliti* e quindi una stima della tonalità più performante.

É ovvio che la qualità di questo algoritmo dipenda direttamente dai risultati del riconoscimento degli accordi. Questo è il motivo per cui con lo stesso algoritmo si ottiene una percentuale di successo solo del 59% sul dataset di musica classica in cui il *chord recognition* ottiene una percentuale di successo minore.

6.2 Sviluppi futuri

Questo problema può essere attenuato ampliando il numero di *Chord Type Templates*, considerando quindi non solo accordi maggiori e minori, ma an-

che accordi diminuiti, eccedenti, nonchè estesi a quattro o cinque voci. La struttura dell'algoritmo permette di fare questo in maniera semplice e immediata. In aggiunta si prevede di implementare un algoritmo di *intonazione* del PCP. In alcuni generi, infatti, (uno su tutti la musica classica) non è detto che l'audio musicale sia intonato rispetto al riferimento a 440Hz.

Altri sviluppi sono legati al riconoscimento automatico del tipo di segnale (monofonico o polifonico) e all'implementazione di un sistema di riconoscimento dei suoni *pitched* o *unpitched*.

Sono infine da valutare gli eventuali risvolti positivi derivanti dall'utilizzo di una *Q Transform* invece che la Trasformata di Fourier.

Bibliografia

[1] Casey, Michael A., et al. Content-based music information retrieval: Current directions and future challenges. Proceedings of the IEEE 96.4 (2008): 668-696.

[2] Gómez, Emilia, and Perfecto Herrera. Automatic extraction of tonal metadata from polyphonic audio recordings. Proceedings of 25th International AES Conference, London. 2004.

[3] Lartillot, Olivier, and Petri Toiviainen. A Matlab toolbox for musical feature extraction from audio. International Conference on Digital Audio Effects. 2007.

[4] Oudre, Laurent, Yves Grenier, and Cédric Févotte. Template-based Chord Recognition: Influence of the Chord Types. ISMIR. 2009.

[5] Lee, Kyogu. Automatic chord recognition from audio using enhanced pitch class profile. Proc. of the International Computer Music Conference. 2006.

[6] Harte, Christopher. Towards automatic extraction of harmony information from music signals. Diss. Department of Electronic Engineering, Queen Mary, University of London, 2010.

[7] De La Cuadra, Patricio, Aaron Master, and Craig Sapp. Efficient pitch detection techniques for interactive music. Proceedings of the 2001 International Computer Music Conference. 2001.

[8] Fujishima, Takuya. Realtime chord recognition of musical sound: A system using common lisp music. Proc. ICMC. Vol. 1999. 1999.

[9] Noll, A. Michael. Pitch determination of human speech by the harmonic product spectrum, the harmonic sum spectrum, and a maximum likelihood estimate. Proceedings of the symposium on computer processing communications. Vol. 779. 1969.

[10] Bruno di Giorgi, Massimiliano Zanoni, Augusto Sarti, Stefano Tubaro "Automatic chord recognition based on the probabilistic modeling

of diatonic modal harmony", in Proceedings of the 8th international workshop on multidimensional (nD) systems - nDS13 - 2013, Erlangen, Germany

[11] Alberto Bertoni, Giuliano Grossi. Elaborazione Numerica dei Segnali

[12] http://newt.phys.unsw.edu.au/jw/notes.html