Università di Padova  Facoltà di Ingegneria

Corso di Laurea in Ingegneria Elettronica

Dipartimento di Elettronica ed Informatica

Tesi Di Laurea

# Un metodo per la separazione di suoni polifonici per la trascrizione automatica del pianoforte.

Relatore: Prof. GIOVANNI DE POLI

Correlatore: Dott. Ing. CARLO DRIOLI

Laureando: LUCA TIENGO


Anno Accademico 2000-2001

**Ai miei genitori**
**Ad Enzo ed Ivan**
**Ai Ledel**

**“O voi che errate nel Paese Oscuro,**
**Non disperate! Benché d’aspetto a volte cupo e duro,**
**Ogni bosco finisce**
**Ed il sole apparisce:**
**Il sole dell’alba, il sole del vespro,**
**Il giorno che nasce o che muore grandioso,**
**Poiché il bosco finisce, ad Ovest o ad Est…”**
**J.R.R. Tolkien “*Il Signore Degli Anelli*”**

Luca Tiengo
lucatiengo@libero.it
http://digilander.iol.it/lucatiengo

# 1 Indice

# 2 Sommario

Questa tesi fa parte del campo di studi che riguardano la trascrizione automatica della musica, dove l'idea è di registrare una *performance* musicale, convertirla in un formato digitale che il calcolatore possa trattare (questi primi due punti possono coincidere) e quindi, attraverso opportuni algoritmi, ricavarne una rappresentazione simbolica. Un orecchio bene allenato, riesce a ricavare da un brano musicale non troppo complesso diversi parametri. Generalmente si riescono a distinguere i vari strumenti musicali che intervengono in tale pezzo, le note che questi suonano, il loro volume e la loro durata, il tempo del pezzo (inteso come ritmo) e, molto approssimativamente, la velocità del brano musicale (*bpm*, ossia "*beats per minute*": la velocità del metronomo). Il nostro lavoro si pone come obiettivo di poter ricavare solo due di questi parametri: l'altezza delle note e la loro durata. Le ipotesi fondamentali sono che la sorgente di tali note sia un pianoforte e il brano musicale sia polifonico.

Per prima cosa si è compiuta una ricerca che coprisse il campo di interesse di questa tesi: la trascrizione automatica della musica, la meccanica e la fisica del pianoforte. Si è cercato di stabilire quale sia lo stato dell'arte della ricerca e si sono confrontate diverse filosofie di approccio, cercando di valutare quale fosse la migliore sia dal punto di vista dei risultati, che delle aspettative che lo sviluppo di tali filosofie offrivano, che del tempo (poco) che avevamo a disposizione.

Alla fine si è implementato e simulato un algoritmo in ambiente MatLab che permettesse di trascrivere un brano per pianoforte solo, memorizzato in un *file wave*.

# 3 Introduzione

**“In principio salute, alla fine scompiglio”**
**I Ching**[1]

## 3.1 Qual è il problema?

Per trascrizione della musica si intende l’atto di ascoltare un pezzo musicale e scrivere una notazione musicale che rappresenti le note che compongono il pezzo [Martin96]. In altre parole, ciò significa trasformare un segnale acustico in una rappresentazione simbolica che comprenda le note, la loro altezza (*pitch*), il *tracking* e la classificazione degli strumenti utilizzati. Si dovrebbe infine notare che nella notazione, non si scrive l’intensità (*loudness*) di ogni singola nota, ma questa è determinata dal musicista dalla lettura di simboli che suggeriscono l’andamento di tutta la *performance*.

Una persona che non abbia un’educazione musicale, generalmente non riesce a trascrivere musica polifonica, in cui suoni diversi sono suonati simultaneamente, e, a dire il vero, questo processo risulta abbastanza difficoltoso anche a chi sia in possesso di tale educazione [Klapuri98]. Comunque, l’esperienza necessaria al trascrittore per effettuare una trascrizione, è tanto maggiore quanto più è complesso e ricco il brano musicale, sia dal punto di vista della quantità di strumenti presenti, che da quello del numero di note che, contemporaneamente, suona ogni singolo strumento. Comunque, un musicista provetto, è generalmente in grado di superare queste difficoltà… così non è per gli algoritmi recentemente implementati…

Infatti, mentre il problema della trascrizione automatica di musica **monofonica** è, in pratica, risolto, così non è per quanto riguarda la musica **polifonica**, pur se tale musica sia suonata da un unico strumento. Esistono molti algoritmi che trascrivono auto-

[1] Altrimenti detto *Libro dei Mutamenti*. Libro divinatorio Cinese attribuito, tra gli altri, a Confucio.

maticamente musica monofonica, e molti *softwares* commerciali dalle prestazioni soddisfacenti sono oggi disponibili.

I primi tentativi di risolvere il problema della trascrizione automatica di musica polifonica (usando, quindi, un calcolatore) risalgono ai primi anni ’70, quando Moorer progettò un sistema per trascrivere dei duetti, composti, per esempio, da composizioni a due voci [Klapuri98, Moorer75]. Nonostante le sue limitazioni, è stato il primo trascrittore automatico di musica polifonica. A tutt’oggi esistono diversi algoritmi per la trascrizione di musica polifonica. Sostanzialmente tutti si basano sull’ipotesi fondamentale che il brano sia suonato da un unico strumento, e su ulteriori eventuali restrizioni, come la limitatezza del numero di note suonate contemporaneamente o del *range* di note che si possono suonare (una ottava, due ottave etc…).

Le potenzialità che può offrire un trascrittore automatico di musica sono notevoli, ad esempio per la progettazione di strumenti che musicisti e compositori possano utilizzare per meglio analizzare composizioni disponibili solo in formato audio. Si consideri, per esempio, l’utilità che uno strumento del genere può avere nell’analisi di registrazioni musicali in cui i musicisti improvvisino, come nel caso del jazz, senza contare le possibilità di utilizzare le caratteristiche stesse di due *files wave* per sincronizzarli tra loro.

Come si può vedere, le applicazioni di sistemi per la trascrizione automatica della musica sono confrontabili con quelle per il riconoscimento vocale. Entrambi sono compiti complicati, ma il forte interesse commerciale che investe il secondo, ha portato sicuramente una concentrazione maggiore della ricerca nell’ambito vocale, lasciando a pochi ricercatori appassionati lo studio del problema riguardante la musica [Klapuri98].

La trascrizione automatica della musica è strettamente collegata a più discipline: una di queste, assolutamente fondamentale, è la **psicoacustica**, che studia la percezione del suono (inclusi la musica ed il parlato) da parte dell’organo uditivo. Un'altra scienza fondamentale è quella che in inglese si chiama ***Auditory Scene Analysis***[2], e

[2] Una traduzione approssimativa di tale termine potrebbe essere *Analisi del Luogo Acustico* (*Uditivo*). Tuttavia tale traduzione non rende merito alla vastità dell’argomento, che comprende sia la psicoacu-

che in qualche modo comprende i nostri scopi e molti altri. E' necessario poi conoscere quali siano la **meccanica** dello strumento e la **fisica** che interviene nella generazione del suono da parte di tale strumento. Inoltre, usando il calcolatore come strumento di analisi, e fondamentale la conoscenza dell'**Analisi dei Segnali Digitali** (***Digital Signal Processing***). Se poi l'approccio al problema ha come punto di partenza l'imitazione dei processi che intervengono quando un essere umano trascrive la musica, ecco che è importante avere una conoscenza di base della **struttura** e del **funzionamento** dell'organo uditivo.

Nel capitolo successivo descriveremo quali siano gli approcci e gli algoritmi più interessanti che abbiamo trovato durante la fase di ricerca bibliografica.

Di seguito presentiamo un breve glossario di termini che ricorreranno abbastanza di frequente durante la trattazione del problema [Klapuri98].

| | |
|---|---|
| **Frequenza Fondamentale** | Vedi *pitch* |
| ***Loudness*** | Percezione attribuita ad un suono che corrisponde alla misura fisica dell'intensità percepita. Il *loudness* è una descrizione psicologica dell'ampiezza della sensazione acustica. |
| **Nota** | Attribuiamo due significati a questo termine: il primo è riferito al simbolo utilizzato nella notazione musicale, il secondo al suono prodotto da uno strumento musicale quando quel simbolo sia suonato. |
| ***Onset Detection*** | Estrazione degli eventi che determinano l'inizio di una nota da un bramo musicale. |
| ***Pitch*** | Percezione attribuita ad un suono, che permette di ordinare i suoni in una scala che si estenda dal più "basso" al più "alto". La frequenza fondamentale è il corrispondente fisico. Noi usiamo i termini *pitch* e frequenza fondamentale come sinonimi. |
| ***Pitch Detection*** | Estrazione del (dei) pitch da un *frame*, o, più in generale, da un intero brano musicale. |
| ***Tracking*** **(Ritmico)** | Anche questo termine ha due accezioni diverse: trovare gli istanti in cui una nota comincia e finisce, e dare una descrizione della struttura ritmica del brano qualora siano dati tali istanti. |

## 3.2 Nozioni di base

Vedremo ora quali siano le nozioni necessarie non solo per affrontare il problema, ma anche per comprendere fino in fondo come funzionino le soluzioni già proposte ed, eventualmente, quali siano i loro punti deboli. Tralasceremo, gli aspetti troppo

---

stica, sia lo studio della fisiologia dell'organo uditivo, sia elementi di psicologia, percezione psicologica, statistica e teoria della *gestalt*. La *summa teologica* sull'argomento sembra essere [Bregman90].

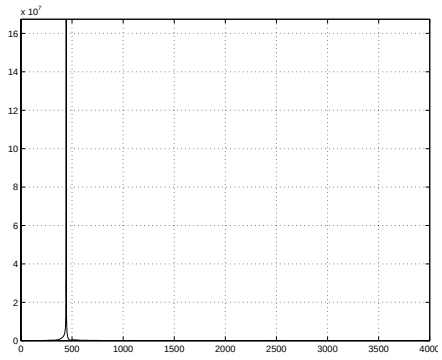

**Figura 3.1: Tono puro a 440 Hz ($A_4$)**

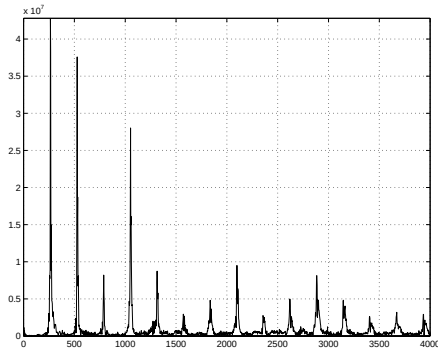

**Figura 3.2: Suono armonico: sassofono ($C_4$)**

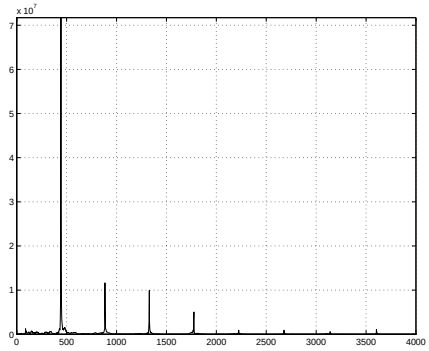

**Figura 3.3: Suono armonico: pianoforte (A4) Bosendorfer a coda.**

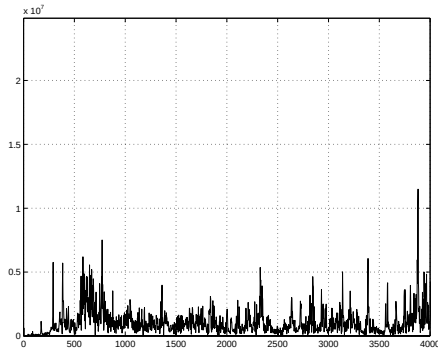

**Figura 3.4: Suono inarmonico: un piatto di batteria (*crash*).**

legati all'acustica, alla fisica del suono, ed alla sua propagazione, soffermandoci sulle nozioni che consideriamo fondamentali alla comprensione dei capitoli successivi.

### 3.2.1 Spettro sonoro e composizione del suono

Per **spettro sonoro** intenderemo l'insieme di frequenze che caratterizzano un suono. Queste lo identificano sia per quanto riguarda il suo *pitch*, sia per quanto riguarda il colore, il timbro. E' grazie alla diversa composizione spettrale che chiunque può distinguere suoni diversi tra loro, seppure con lo stesso *pitch*, come lo squillo di una tromba piuttosto che il rintocco di una campana, o la corda di una chitarra che vibra (si vedano Figura 3.2 e Figura 3.3).

Il suono più semplice che si possa immaginare è quello che presenta una sola riga nel suo spettro, quella riga rappresenta una frequenza (l'unica in questo caso) di quel suono. Va da sé che, essendo l'unica, è anche la fondamentale, il *pitch* di quel suono. Questo tipo di suoni sono detti **toni puri** (Figura 3.1).

In natura non esistono toni puri, né nessuno strumento acustico (leggi: non elettronico) è in grado di generarli. I suoni presenti in natura sono più complessi, il loro spettro presenta più righe, ed, a seconda di come sono poste queste righe, i suoni si dicono **armonici** o **inarmonici**. I suoni armonici sono quelli le cui frequenze presenti nello spettro stanno tra loro in rapporto come numeri interi ($^1/_2$, $^2/_3$, $^3/_4$ e così via), in quelli inarmonici, non c'è alcuna rela-

zione del genere tra le parziali (Figura 3.4). Le parziali dei suoni armonici sono dette armoniche.

Diretta conseguenza delle definizioni appena date è un teorema semplice ma fondamentale.

**Teorema 1**: In un suono armonico esiste una armonica, detta fondamentale, tale che tutte le armoniche presenti nello spettro del suono sono multiple di questa. Tale armonica è detta prima armonica, o anche armonica fondamentale.

**Dimostrazione**: per definizione di suono armonico, prese due armoniche qualsiasi $f_x$ ed $f_y$ presenti nello spettro, allora:

$\frac{f_x}{f_y} = \frac{m}{n}$ con $n$ ed $m$ non nulli $\in$ **N.**

da questo segue che

$\frac{f_x}{m} = \frac{f_y}{n} = f_0$ ovvero $f_x = mf_0$ ed anche $f_y = nf_0$

$f_0$ risulta essere la frequenza fondamentale cercata. •

Si deve aggiungere che le armoniche prendono il nome dal coefficiente moltiplicativo che le mette in relazione con la fondamentale, così $f_n = (n+1)f_0$ è la $(n+1)$-*esima* armonica.

Bisogna notare, che il fatto che esista la fondamentale, non indica che questa abbia energia superiore a quella delle altre armoniche, la sua, anzi, potrebbe essere addirittura trascurabile, così come non è detto che tutte le armoniche siano presenti o abbiano energia comparabile le une le altre. Vedremo qualche esempio nel capitolo successivo, quando parleremo del pianoforte.

Da queste poche considerazioni, risulta che per riconoscere una nota, seppure isolata e non sovrapposta ad altre, non è sufficiente trovare quale sia la sua armonica ad energia maggiore, in quanto questa potrebbe benissimo non essere la sua fondamentale.

### 3.2.2 Suoni armonici non isolati

Considerando due suoni presenti contemporaneamente, denotiamo il suono che vogliamo analizzare con la lettera *S*, ed il suono a lui sovrapposto, che consideriamo un'interferenza, con la lettera *R*.

Come detto il problema della trascrizione automatica di musica monofonica è stato risolto in modo molto soddisfacente, ma il passaggio dall'analisi di suoni singoli a quella di più suoni sovrapposti non risulta affatto immediato. Questo perché nella maggior parte degli strumenti, la banda coperta da una nota è molto ampia (anche diversi kHz) e, soprattutto, come vedremo le armoniche delle note si sovrappongono, rendendo non facile un eventuale riconoscimento.

**Teorema 2**: Siano *S* ed *R* due suoni armonici. Se un'armonica $f_j^S$ di un suono *S* si sovrappone ad un'armonica $f_i^R$ di un suono *R* interferente, allora la frequenza fondamentale del suono *R* sarà:

$f_0{}^R = \frac{m}{n} \cdot f_0{}^S$ con *n* ed *m* non nulli $\in$ **N.**

**Dimostrazione:** considerando le armoniche fondamentali dei due suoni si ha:

$f_j{}^S = (j+1) \cdot f_0{}^S$ per *S*, e $f_i{}^R = (i+1) \cdot f_0{}^R$ per *R*

il fatto che la *j-esima* armonica di *S* di sovrapponga con la *i-esima* di *R* si può esprimere con la relazione:

$$(i+1) \cdot f_0{}^R = (j+1) \cdot f_0{}^S \Leftrightarrow f_0{}^R = \frac{j+1}{i+1} \cdot f_0{}^S$$

e se si riducono *j* e *i* semplificando i fattori comuni si trova:

$f_0{}^R = \frac{m}{n} \cdot f_0{}^S$ con *n* ed *m* non nulli $\in$ **N** •

**Teorema 3**: Se le fondamentali di due suoni armonici *S* ed *R* sono tali per cui $f_0{}^R = \frac{m+1}{n+1} \cdot f_0{}^S$, allora le armoniche di *R*, $f_{nk}{}^R$ si sovrappongono con le armoniche di *S*, $f_{mk}{}^S$ per $k \in \{1, 2, 3, 4\ldots\}$.

**Dimostrazione**: Dal Teorema 2, si ha che $f_0{}^R = \frac{m+1}{n+1} \cdot f_0{}^S$, e quindi la catena di implicazioni $(i \cdot f_0{}^R = j \cdot f_0{}^S) \Leftrightarrow (i\frac{m+1}{n+1} \cdot f_0{}^S = j \cdot f_0{}^S) \Leftrightarrow [i \cdot (m+1) = j \cdot (n+1)]$, che risulta vero per ogni $i = (n+1)k$ e $j = (m+1)k$ con $k \in \{1, 2, 3, 4\ldots\}$. •

Conseguenza importante di quest'ultimo teorema è che se un'armonica di *S* si sovrappone alla fondamentale di *R*, il che vuol dire che le due fondamentali sono divise da una o più ottave, allora tutte le armoniche di *R* si sovrapporranno con le armoniche di *S*.

### 3.2.3 Il Sistema Temperato

Il "sistema musica" occidentale, è fondato (direi quasi governato) su relazioni matematiche che mettono in relazione note che "suonano bene" assieme. In termini di frequenza, la distanza tra due note è detta **intervallo**. Due note sono in **relazione armonica** se le loro fondamentali soddisfano relazioni del tipo

$f_{0_1} = \frac{m}{n} \cdot f_{0_2}$ con *n* ed *m* non nulli $\in$ **N** e "piccoli".

Se consideriamo *m* ed *n* primi tra loro, tanto più piccoli sono, tanto migliore è la relazione armonica tra i due suoni, e tanto meglio suonano assieme. Per esempio, *f*, $^4/_5 \bullet f$ e $^4/_6 \bullet f$ costituiscono un accordo maggiore, mentre $^4/_6 \bullet f$, $^4/_5 \bullet f$ e *f* costituiscono un accordo minore.

Proseguendo con queste relazioni, fissata una nota, si potrebbe ricostruire tutta la scala. Il problema che nasce sarebbe che, costruita tale scala, se si cambiasse tonalità, le note suonerebbero stonate… e di molto!

Ecco quindi che è stato introdotto il sistema temperato, in cui le note sono organizzate in un sistema logaritmico, dove la fondamentale di una nota è pari a:

$f_{0_k} = 440 \cdot 2^{\frac{k}{12}} Hz$, con *k=-48:39* e questo comporta che due note successive abbiano le fondamentali che stanno in rapporto tra loro di $2^{\frac{1}{12}} Hz$. La conseguenza immediata è che le note saranno sempre stonate! Nel senso che in nessuna tonalità le relazioni armoniche saranno rispettate, e gli intervalli non saranno uguali a quelli ideali (Tabella 1).

| **Nota 1** | **Nota 2** | **$f_{01}/f_{02}$** | **m** | **n** | **m/n** | **Deviazione dalla relazione armonica** |
|---|---|---|---|---|---|---|
| C# | C | 1.0595 | 16 | 15 | 1.0667 | -0.68% |
| D | C | 1.1225 | 9(8) | 8(7) | 1.125(1.143) | -0.23% (+1.8%) |
| D# | C | 1.1892 | 6 | 5 | 1.2 | -0.91% |
| E | C | 1.2599 | 5 | 4 | 1.25 | +0.79% |
| F | C | 1.3348 | 4 | 3 | 1.3333 | +0.11% |
| F# | C | 1.4142 | 7 | 5 | 1.4 | +1.0% : molto!!! |
| G | C | 1.4983 | 3 | 2 | 1.5 | -0.11% |
| G# | C | 1.5874 | 8 | 5 | 1.6 | -0.79% |
| A | C | 1.6818 | 5 | 3 | 1.6667 | +0.91% |
| A# | C | 1.7818 | 16(7) | 9(4) | 1.778(1.75) | +0.23% (-1.8%) |
| H | C | 1.8877 | 15 | 8 | 1.875 | +0.68% |

**Tabella 1: Confronto tra gli intervalli dovuti alle relazioni armoniche e quelli dovuti al sistema temperato** [**Klapuri98**]

L'adozione del sistema temperato porta al fatto che le parziali di due note non si sovrappongono più, in realtà, e questo perché le relazioni armoniche non sono più rispettate dalle fondamentali. Tuttavia lo scostamento non è così drammatico, e continueremo a dire che le parziali si sovrappongono come se le relazioni fossero rispettate, questo perché le armoniche, seppur non sovrapposte, risultano comunque molto vicine.

| **Accordo** | **Note nell'accordo** | | | | | **fondamentali delle note in forma *m/n*** | | | | | **Percentuale di armoniche sovrapposte** | | | | | **Media** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Cmajor | c | e | g | | | 1 | 5:4 | 3:2 | | | 47 | 33 | **60** | | | 47 |
| Cminor | c | d# | g | | | 1 | 6:5 | 3:2 | | | 33 | 20 | **50** | | | 33 |
| $C^0$add6 | c | d# | f# | a | | 1 | 6:5 | 7:5 | 5:3 | | **43** | 31 | 33 | 33 | | 35 |
| $C^+$ | c | e | g# | c2 | | 1 | 5:4 | 8:5 | 2 | | 60 | 25 | 30 | **100** | | 54 |
| $C^9$ | c | d | e | g | a# | 1 | 9:8 | 5:4 | 3:2 | 16:9 | 50 | 23 | 39 | **65** | 11 | 38 |
| c+Cmajor | c1 | c3 | e3 | g3 | | 1 | 4 | 5 | 6 | | 47 | **100** | **100** | **100** | | 87 |
| inarmonico | c | c# | d | d# | e | 1 | 16:15 | 9:8 | 6:5 | 5:4 | **41** | 11 | 20 | 25 | 33 | 26 |

**Tabella 2: Analisi delle armoniche che si sovrappongono negli accordi più comuni** [**Klapuri98**]

In Tabella 2 si nota come la sovrapposizione delle note possa davvero essere una cosa drammatica. In alcune tipologie di accordi (si veda il caso di Cmaj con un ulteriore basso C) la sovrapposizione può essere totale. Cosa vuol dire? Che nel caso appena citato, per esempio, se noi togliessimo la seconda nota (un E) cambierebbe solo l'energia delle parziali, ma non ne sarebbe compromessa la presenza, continuerebbero ad esserci, seppur con un'energia minore. E' pur vero che questo non succederebbe nel sistema temperato, ma si deve tener presente che, le deviazioni delle posizioni

delle fondamentali, non sono poi così evidenti (se non in alcuni casi) e basterebbero piccole inarmonicità tipiche di uno strumento (ne parleremo per il pianoforte nel capitolo successivo) a far spostare le parziali.

### 3.2.4 Conclusioni

Dai dati e dalle relazioni imposte fino ad ora, si può dedurre che, se l'algoritmo creato per trovare le note si basa su un'analisi delle parziali, è bene che tali parziali si sovrappongano il meno possibile con quelle di altre note. Un semplice teorema ci viene in aiuto:

**Teorema 4**: Se consideriamo due suoni *R* ed *S* tali che $f_0^R \neq f_0^S/n$ con *n* non nullo $\in$ **N** (ovvero *R* non deve essere **subarmonico** di *S*), allora *R* si sovrappone ad *S* solamente con una armonica che stia in una posizione contrassegnata da un numero primo (la prima, o fondamentale, la seconda, la terza, la quinta e così via…). Le chiameremo **armoniche prime** (da non confondere con la prima armonica, che corrisponde alla fondamentale).

**Dimostrazione**: Supponiamo per assurdo che ci siano due armoniche prime di *S*, che stiano in una posizione contrassegnata da un numero primo, tali che queste siano anche armoniche di *R*. Siano ora $f_0^S$ ed $f_0^R$ le armoniche fondamentali. Consideriamo un numero primo arbitrario $p_j$. La condizione che due armoniche prime di *S* primo si sovrappongano a due armoniche di *R* si può scrivere come:

$$\begin{cases} i_1 \cdot f_0{}^R = p_1 \cdot f_0{}^S \\ i_2 \cdot f_0{}^R = p_2 \cdot f_0{}^S \end{cases}$$ risolvendo il sistema si ottiene $$p_2 = \frac{p_1 \cdot i_2}{i_1}$$

Perché $p_2$ sia un numero primo diverso da $p_1$ , condizione necessaria perché le armoniche siano distinte, deve essere $i_1 = n \bullet p_1$, dove *n* è un intero tale che $i_2 = n \bullet p_2$. Sostituendo questa condizione nel sistema si ottiene

$$f_0{}^R = \frac{p_1 \cdot f_0{}^S}{i_1} = \frac{p_1 \cdot f_0{}^S}{n \cdot p_1} = \frac{f_0{}^S}{n}$$ con *n* non nullo $\in$ **N**

e questo porta alla negazione dell'ipotesi, dimostrando il teorema.•

# 4 Il Pianoforte

**"Know your Enemy"**
**The Manic Street Preachers**

Come passo iniziale della nostra ricerca abbiamo pensato fosse opportuno studiare la storia e la struttura di questo strumento così versatile e presente in tutti i generi musicali, dalle composizioni settecentesche alle avanguardistiche "preparazioni" di John Cage.

Esula, ovviamente, dagli scopi di questa tesi il fornire nozioni esaustive ed esaurienti su come sia nato, come si sia diffuso, quali siano state le maggiori e più seminali innovazioni che hanno portato questo strumento ad essere così importante per musicisti e compositori di tutti i generi, ma il problema che abbiamo affrontato, è fortemente correlato alla struttura stessa del pianoforte, a come la sua cassa armonica "ammorbidisca" il suono, a come le corde lo producano, a come i suoi legni lo smussino.

## 4.1 Breve storia del pianoforte

Il moderno pianoforte è diretto discendente del clavicembalo. In questo strumento le corde sono "pizzicate" da dei gancetti collegati, attraverso delle meccaniche, ai tasti. Nel 1709 Bartolomeo Cristofori sostituì questi gancetti con dei martelletti, che andavano a percuotere le corde. Non si pensi che questo piccolo stratagemma abbia portato subito al pianoforte come lo conosciamo ora, anzi! Tuttavia permetteva il controllo del volume con cui si suonava[1], proporzionale alla forza con cui si schiaccia il tasto, ed introduceva un nuovo colore nel suono. L'invenzione di Cristofori fu introdotta in Germania dai costruttori di organi e clavicembali Gottfried Silbermann, Johannes Zumpe e Andreas Stein. Le critiche di Bach contro i primi prototipi, li convinsero a

[1] A dire il vero anche il clavicembalo può offrire dinamiche diverse, a seconda di quanto velocemente si premono i tasti, tuttavia, paragonato al piano, il *range* della dinamica è più limitato, tant'è vero che le variazioni dinamiche non sono indicate nella musica scritta per clavicembalo. [Askenfelt90, p.18]

portare miglioramenti alle meccaniche. Zumpe si spostò in Inghilterra, dove con John Broadwood portò notevoli miglioramenti ai meccanismi. Stein invece, costruì il pianoforte di tipo "Viennese", che sarà suonato da Mozarth. Per finire Pierre Erard è accreditato come l'inventore della **graffa** e del **meccanismo a doppia ripetizione**.

Il pianoforte verticale fu inventato da Joseph Hawkins di Filadelfia e Robert Wornum di Londra nella metà del diciannovesimo secolo. Il primo, in particolare, introdusse l'utilizzo di una lastra d'acciaio, che, rinforzando il telaio, permette di ottenere maggiore tensione dalle corde, senza il pericolo di danni strutturali.

Ecco che in circa tre secoli, due differenti famiglie di pianoforte si sono sviluppate separatamente, ognuna con le sue peculiarità timbriche e costruttive, e con strumenti di misura diversa. [Fletcher-Rossing98, Askenfelt90]

## 4.2 Struttura e proprietà acustiche del pianoforte

Un moderno pianoforte a coda si può trovare in diverse misure, ma generalmente possiede 88 tasti[2], coprendo più di sette ottave (da $A_0$ fino a $C_8$). E' a tutt'oggi lo strumento acustico (leggi: non elettronico) che copre la maggior estensione, e per la sua struttura è anche uno di quelli con la maggior ricchezza armonica.

La struttura di un pianoforte è la seguente: una lastra di metallo è attaccata alla parte superiore della **cassa armonica**, e le corde sono tese su tale lastra in una direzione pressoché perpendicolare alla tastiera.

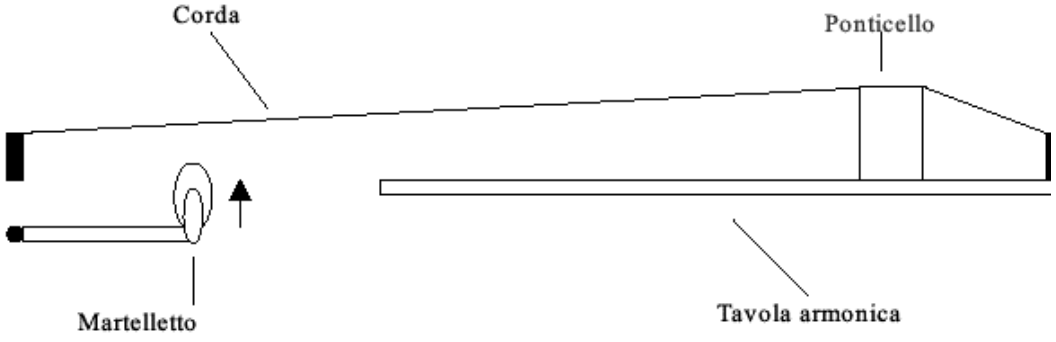


**Figura 4.1: Schema semplificato delle meccaniche** [**Bank00**]

[2] Alcuni tipi di pianoforte da concerto possono arrivare anche a 92 tasti. [Askenfelt90, p.18]

La parte terminale della corda che finisce vicino alla tastiera è collegata a delle **meccaniche** che servono ad accordare lo strumento, mentre l'altra terminazione passa per il **ponte** (o ponticello) per attaccarsi alla lastra di metallo. Il ponte è composto da una barra di legno che trasmette le vibrazioni della corda alla cassa armonica. Uno schema semplificato si può vedere nella Figura 4.1.

Bank [Bank00] classifica i meccanismi di produzione del suono propri di un pianoforte in tre diversi momenti:

1. Il martello colpisce la corda. Le meccaniche del pianoforte sono strettamente correlate a questa prima fase, poiché trasmettono l'energia cinetica del martelletto, alla corda, e la trasformano in energia sonora.

2. Questa energia è immagazzinata dalla corda sottoforma di vibrazioni (modi normali). Ovviamente una parte viene dissipata da fenomeni di perdita interna, la restante raggiunge la cassa armonica, attraverso il ponte.

3. La cassa armonica, infine, converte le vibrazioni in energia acustica: il suono che noi udiamo.

### 4.2.1 Il ponticello

Il ponte funziona come un trasformatore d'impedenza, presentando alla corda un'impedenza maggiore rispetto al caso in cui il ponte non ci fosse e la corda fosse collegata direttamente alla cassa armonica [Bank00, p.27]. E' un componente molto importante del pianoforte, infatti attraverso una accurata progettazione di questo, e della cassa armonica, si può arrivare ad un buon compromesso tra volume del suono prodotto e tempo di decadimento della nota, che sono due fattori generalmente in rapporto inverso tra loro [Fletcher-Rossing98].

### 4.2.2 I martelletti

Alcune ricerche sembrano dimostrare che il suono risultante è unicamente controllato dalla velocità finale del martelletto[3], ciò non di meno gli artisti prestano molta attenzione al "tocco": secondo la loro opinione notevoli variazioni si possono ottenere variando il tipo di tocco. Studi di Askenfelt e Jansson dimostrano che il martelletto è soggetto a varie risonanze, dipendenti dal diverso tipo di tocco [Askenfelt90, pp. 39-58]. Nel legato, per esempio, l'ampiezza della risonanza del martelletto è considerevolmente inferiore rispetto allo staccato. La frequenza di risonanza si trova a 50, 250 e 600 Hz, con un fattore di qualità compreso tra 15 e 30. E' evidente che tali risonanze possono influenzare la vibrazione delle corde.

I martelletti sono molto importanti per il suono prodotto da un pianoforte, sia dal punto di vista del volume emesso, che del timbro. Il cuore ligneo dei martelletti è ricoperto da un feltrino di lana, più o meno soffice, a seconda delle corde che tale martelletto è destinato a colpire, e del timbro che si vuole ottenere. Un feltrino più duro, infatti, produce parziali più presenti, e quindi un suono più "chiaro". Viceversa se i feltrini sono più morbidi il suono sarà caratterizzato da parziali meno forti, e quindi risulterà più scuro e spostato più verso le basse frequenze.

Le corde risultano maggiormente sollecitate quando il tempo di contatto con il martelletto è pari al semi periodo del tono fondamentale di tale corda[4]. Questo è verificato per le corde che stanno al centro della tastiera. Al contrario nella parte inferiore della tastiera il contatto dura meno, in quella superiore dura di più. Ecco quindi che nei moderni pianoforti, si utilizzano martelletti di massa diversa, a seconda delle corde che devono colpire.

Lo spettro del suono è determinato anche dal punto in cui il martelletto va a colpire la corda. Questo, infatti, introduce un effetto filtrante di tipo "a pettine" qualora i

[3] Ciò non significa che il suono di un pianoforte non sia influenzato dalle sue dimensioni, dai legni usati, dalle vernici o quant'altro. Qui si parla di *controllo*, ovvero di come il musicista possa controllare il suono.

[4] Il tempo di contatto medio tra corda e martelletto è determinato dal rapporto tra le masse del martelletto e della corda. Più il martelletto è pesante, maggiore è il tempo di contatto [Bank00].

modi interessati avessero un nodo in prossimità del punto d'impatto[5], si cercherà, quindi, di progettare il martelletto in modo che le frequenze eliminate da tale effetto filtrante coincidano il meno possibile con quelle fondamentali della nota.

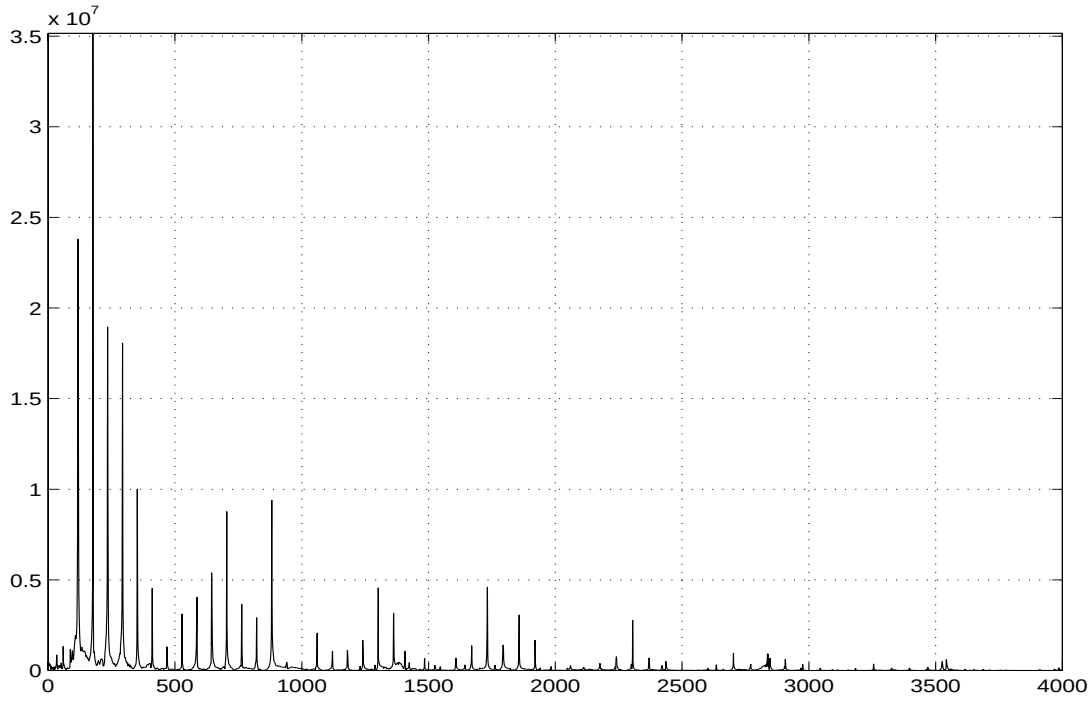


**Figura 4.2: Rappresentazione in frequenza di un A#$_1$ di un pianoforte Bosendorfer.**

Un'alta velocità del martelletto al momento dell'impatto non influenza solamente il volume del suono emesso, ma anche e soprattutto il tipo di onda emessa e la ricchezza armonica dello spettro.

Askenfelt e Jansson, infatti, mostrano come la velocità del martelletto agisca come una specie di filtro passa basso: maggiore è la velocità con cui il martelletto colpisce la corda, maggiore è l'estensione armonica del suono, in altre parole le parziali si fanno vedere a frequenze più alte [Askenfelt90, pp. 55-56]. Ovviamente questo influisce sul volume sonoro (ovvero sull'energia sonora emessa) ma l'effetto principale

[5] Se il martelletto va a colpire la corda in una frazione $\beta$ della sua lunghezza, mancheranno le armoniche multiple di $^1/_\beta$. Nei pianoforti $\beta$ tipicamente varia tra $^1/_7$ nelle corde basse e $^1/_{20}$ in quelle alte. Si sta comunque ancora studiando il problema, per vedere quale sia il punto migliore per ottenere il miglior suono possibile [Fletcher-Rossing98, p338]. Si prenda come esempio la nota raffigurata in Figura 4.2: si vede che manca l'ottava armonica, la sedicesima, la ventiquattresima e così via. Presumibilmente, quindi, tale corda è stata colpita dal martelletto ad un'altezza pari ad un ottavo della sua lunghezza.

è la diminuzione della banda del suono, piuttosto che il calo di energia delle parziali più vicine alla fondamentale.

### 4.2.3 Le corde

E' logico che le corde, o meglio la loro struttura ed il modo in cui sono collegate al ponte ed alla cassa armonica, giocano un ruolo molto importante sulla definizione del suono di un pianoforte. In realtà non c'è una sola corda per ogni nota, quando si preme un tasto di un pianoforte, non si va a sollecitare una singola corda. In effetti la tastiera del pianoforte è così organizzata: cominciando dalla sinistra della tastiera, ogni martelletto appartenente alla prima parte (dieci tasti), sollecita una singola corda, la seconda (dieci tasti) ne sollecita due contemporaneamente, la terza parte (il resto della tastiera) sollecita tre corde contemporaneamente, accordate all'unisono.

Perché si è scelto di far suonare più corde contemporaneamente? Un modo per migliorare l'accoppiamento tra corde e piano armonico sarebbe quello di aumentare il loro diametro (e quindi la loro massa), ma questo porterebbe ad un aumento dell'inarmonia delle corde (ne parleremo più avanti). Quindi un modo migliore è quello di mettere più corde assieme, accoppiate tra loro, in modo da raggiungere una massa ed un diametro equivalente soddisfacenti. Quando un martelletto colpisce tre corde messe in questo modo, le fa vibrare con la stessa fase, quindi esercitano una forza verticale sul ponte, con la stessa fase, aumentando di molto l'energia che una singola corda avrebbe trasferito al piano armonico. Tuttavia, a causa di piccole (inevitabili) differenze di accordatura, presto vanno fuori fase, e la forza risultante sul ponte diminuisce. Tuttavia, un accordatore esperto, riesce a bilanciare il suono iniziale, ed anche quello a regime, delle corde unisone. Infatti, se queste sono leggermente "scordate" tra loro, presto entrano in battimento. La fisica ci insegna che, se queste corde si potessero rappresentare come due oscillatori indipendenti, ognuno con una propria frequenza caratteristica, inizierebbero ad oscillare ad una frequenza intermedia, e l'emissione sonora sarebbe modulata ad una frequenza pari alla differenza tra le due frequenze. Nella realtà le due (o tre) corde non oscillano indipendentemente tra loro, sono accoppiate al ponte, e quindi danno luogo al sorgere di nuove frequenze modali [Fletcher-Rossing98].

Tale accoppiamento pare anche essere la causa, del cosiddetto “decadimento a due stadi”, il fenomeno per cui il tono del suono decade più velocemente nella prima parte (appena dopo il transitorio dovuto al battito del martelletto) che nella seconda [Fletcher-Rossing98]. Ma sembra esserci un’altra spiegazione per questo fenomeno: il comportamento delle due diverse polarizzazioni. La polarizzazione verticale della corda si accoppia più facilmente al ponte rispetto a quella orizzontale, e questo fa in modo che i tempi di decadimento differiscano di molto. Il martelletto, evidentemente, sollecita molto di più la polarizzazione verticale, sicché è questa che trasporta la maggior quantità di energia, ma dato che questa decade molto più velocemente, ecco che anche il volume del suono sarà maggiore all’inizio, per poi stabilizzarsi sui valori dominati dalla polarizzazione orizzontale. Questa è solo una semplificazione del fenomeno reale, in cui le due polarizzazioni sono accoppiate tra loro, ed in cui entra in gioco anche una polarizzazione che è longitudinale alla corda colpita.

La vibrazione delle corde non è scomponibile solamente in polarizzazione verticale ed orizzontale, entra in gioco anche la polarizzazione longitudinale. Questo avviene soprattutto per le corde basse. Si è detto precedentemente che è necessario raggiungere un certo diametro ed una certa massa. Spesso non è sufficiente “dividere” la massa tra due corde, perché comunque dovremmo avere diametri troppo grossi, e la corda perderebbe di elasticità. Si pensi a cosa può significare l’avere una corda di metallo del diametro, seppur piccolo, di un centimetro! Sarebbe una barra che farebbe fatica a vibrare, e soprattutto non avrebbe un gran suono! Ecco quindi che, in alternativa, si usano corde composte da un’anima avvolta di filo metallico. Si può comprendere anche dalla loro forma (somigliano a delle molle) che la polarizzazione longitudinale può essere molto importante.

### 4.2.4 I pedali

Il piano, inoltre, possiede due o tre pedali. Il più importante è il pedale del *sustain* (sulla destra) che alza dei feltrini che bloccano le corde. Questo, da un lato cambia il timbro della corda suonata, aumentando, tra l’altro, il tempo di decadenza, e dall’altro cambia completamente il suono del pianoforte, poiché tutte le corde sono lasciate libere di vibrare e, quindi, di entrare in risonanza tra loro. Il pedale di sinistra, permette di suonare una sola delle corde corrispondenti ad un tasto (ricordiamo

che la prima decina di tasti colpisce una singola corda, la seconda due, i restanti tre, accordate all'unisono), questo causa una caduta di volume sonoro pari a circa un *decibel*, valore abbastanza modesto, ma cambia notevolmente la timbrica dello strumento.

### 4.2.5 La cassa armonica

La cassa armonica degli strumenti moderni è generalmente composta di strati di legno di circa 5-15 cm. sovrapposti ed incollati tra loro. La cassa armonica è molto importante, dato che è proprio attraverso i suoi legni e la sua struttura che il suono viene amplificato e diffuso. Non parleremo estesamente delle sue risonanze e di come crei, arrotondi il suono proveniente da martelletti, corde e ponticello, rimandiamo ad altre letture per questo [Askenfelt90], [Fletcher-Rossing98].

E' però importante far notare che la cassa armonica funge da filtro passa alto, tagliando le frequenze molto basse, più o meno sotto i 100 Hz. Infatti (si veda anche Figura 4.2) nelle note molto basse si vede benissimo come risultino pressoché assenti, o notevolmente attenuate, le prime parziali.

### 4.2.6 Analisi dello spettro sonoro

Abbiamo già parlato di come le prime parziali siano poco visibili. Questo è un fattore di cui bisogna tener conto. Supponiamo di voler trascrivere musica monofonica per pianoforte, se noi partissimo dall'ipotesi (valida per altri strumenti) che la frequenza fondamentale è anche la più forte, quella con maggior energia, sbaglieremmo la trascrizione, e compieremmo un errore abbastanza tipico: troveremmo una nota distante una o più ottave da quella realmente presente.

Anche l'assenza delle parziali dovuta al punto d'impatto dei martelletti può risultare un problema, a seconda del tipo di soluzione scelta.

Altra discrepanza dal caso ideale è la cosiddetta inarmonicità. Se il pianoforte seguisse le regole matematiche che stanno alla base dei suoni cosiddetti armonici, allora il suo spettro presenterebbe una serie di righe poste alla frequenza $f_0$, $2f_0$,$3\ f_0$, ecc… dove $f_0$, è la fondamentale. Così non è. Le armoniche si troveranno, per esempio, alla

frequenza $f_0$, $2.0001f_0$, $3.045\ f_0$, ecc… e quindi, se noi andassimo a vedere quale sia l'ampiezza della terza parziale, non la troveremmo! Questa è una cosa da tener ben presente se si effettua uno studio in frequenza, si dovranno considerare degli intorni della posizione ideale in cui si dovrebbe trovare la parziale.

Si sono fatte diverse ipotesi sulle origini dell'inarmonicità del pianoforte, quasi tutte attribuiscono questo fenomeno alla struttura delle corde, a come sono fatte, ai materiali utilizzati ed a come sono accoppiate tra loro [Askenfelt90], [Fletcher-Rossing98].

+

# 5 Metodiche incontrate

**"L'esperienza è una lanterna che ci portiamo dietro le spalle,**
**purtroppo illumina solo il cammino già percorso"**
**Confucio**

**"Non esiste un'avanguardia,**
**ci sono solo persone un po' in ritardo"**
**Edgard Varèse**[1]

Prima di cercare una soluzione al problema della trascrizione automatica della musica, abbiamo effettuato una ricerca sia attraverso la rete Internet, che tramite canali più tradizionali, come biblioteche e riviste.

La trascrizione automatica di brani monofonici è praticamente un problema risolto, esistendo molti sistemi capaci di operare addirittura in tempo reale, ma se si considera il caso polifonico, con uno strumento che suoni più note contemporaneamente o, addirittura, con più strumenti diversi che suonino assieme, la discussione è ancora aperta, ed il problema può presentare difficoltà drammatiche.

Bisogna risalire agli anni settanta per poter trovare un primo tentativo di soluzione. Risale, infatti, al 1975, quando Moorer propose un sistema per trascrivere duetti, o comunque composizioni a due voci [Moorer75]. Il suo sistema soffriva di limitazioni molto restrittive, per quanto riguardava la relazione che doveva intercorrere tra le due note che suonavano contemporaneamente ed il *range* delle note ammesse.

Da allora sono stati pubblicati molti altri lavori sull'argomento, anche se, a detta degli studiosi del ramo, questo segmento del "*signal processing*" non è oggetto dello stesso interesse che invece si rivolge al ramo che studia la voce, la trascrizione ed il riconoscimento del parlato.

[1] Paris, 1883-New York, 1965. Universalmente riconosciuto tra i padri della musica elettrmonica. Ispirò moltissimi musicisti, tra cui Frank Zappa [sito12].

Le metodiche si possono dividere in due grandi famiglie: la prima analizza il suono nel dominio del tempo, la seconda nel dominio delle frequenze. Al primo gruppo appartengono due metodi in particolare, quello basato sull'autocorrelazione[2] e quello che analizza le caratteristiche delle forme d'onda, come i picchi, le valli e gli *zero crossing*. Quest'ultimo fu utilizzato da Moorer per il suo trascrittore, il primo risulta utile nel caso del *pitch detection* di segnali monofonici, ma non applicabile a quelli polifonici. In questo campo, le soluzioni basate sul dominio del tempo, risultano avere un valore esclusivamente storico [Klapuri98].

Secondo noi, allo stato attuale, gli studi, gli approcci e le pubblicazioni analizzate trovano la loro massima espressione in tre lavori.

Il primo cerca di modellare le proprietà dell'orecchio umano. Se un essere umano riesce a trascrivere musica, un *software* che emuli il suo sistema uditivo non potrebbe essere un buon punto di partenza?

Il secondo cerca di affrontare il problema attraverso le reti neurali. Sistemi basati sulle reti neurali hanno trovato varie applicazioni, con risultati più che soddisfacenti, per molti problemi di riconoscimento, come il riconoscimento vocale, per esempio.

Il terzo è, invece, un procedimento iterativo, basato sull'analisi dello spettro del suono, che ha ottenuto ottimi risultati. Sull'iterazione **trova nota -> elimina nota -> trova nota successiva** si basa anche la nostra soluzione.

## 5.1 Modello Uditivo

La prima soluzione che presenteremo, è quella proposta da Malcolm Slaney e Richard F. Lyon. Il loro "*Perceptual Pitch Detector*" è un algoritmo che si basa sulla modellizzazione del comportamento dell'orecchio umano, per trovare quale sia il *pitch* dominante in un *frame* musicale. Questa filosofia è, come detto, una delle pri-

[2] La funzione di autocorrelazione è particolarmente utile nel trovare periodicità nascoste in un segnale. L'autocorrelazione di un segnale di lunghezza N nel tempo *x(k)* è data da:

$r_{xx}(n) = \frac{1}{N} \cdot \sum_{k=0}^{N-n-1} x(k) \cdot x(k+n)$ dove *n* è un lasso temporale.

me strade che intuitivamente verrebbe in mente di percorrere: se l'orecchio umano funziona, perché non imitarlo? In realtà vedremo che, a tutt'oggi, questa strada non porta a grandi risultati, almeno non nella trascrizione della musica polifonica [Klapuri98] e non abbiamo trovato nessuno studio che applicasse questa teoria per trascrivere musica polifonica. Ciò non toglie che altre soluzioni, ad esempio la seconda che presenteremo, utilizzino il modello uditivo di Slaney, come componente del sistema.

### 5.1.1 Il Modello Cocleare di Lyon[3]

Questo modello descrive la propagazione del suono attraverso l'orecchio interno, e la trasformazione dell'energia acustica in una rappresentazione neuronale.

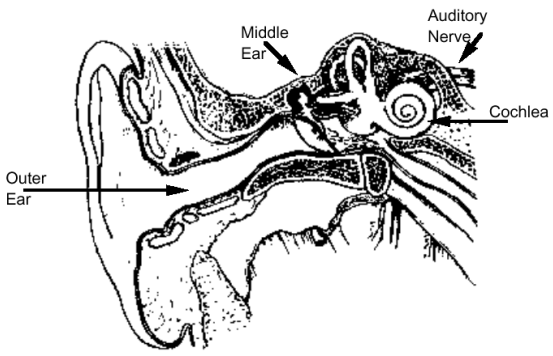


**Figura 5.1: Struttura dell'orecchio** [**Slaney88**]

E' ben noto l'effetto sul suono quando questo entra nell'orecchio e scende per il canale uditivo: è quello di un semplice filtro lineare. Nell'orecchio medio l'energia sonora passa attraverso il timpano ed una serie di ossicini fino al fluido di cui è riempita la camera della coclea (chiamata anche **chiocciola**). Si pensa anche che sia l'orecchio medio a provvedere ad una sorta di controllo automatico del guadagno (AGC, *Automatic Gain Control*) attraverso il riflesso del nervo stapediale[4].

Anche il comportamento della coclea, che invece è stato considerato, non viene analizzato, relativamente alla struttura della coclea stessa, in maniera rigorosa. In pratica la coclea viene vista come una "scatola nera", e viene fatto un modello del suo comportamento e delle sue funzioni complessive. Il suono, che entra nella coclea, viene

---

[3] *Lyon's Cochlear Model*, abbozzato per la prima volta da Lyon, ma basato su lavori precedenti si Schroder e Zweig [Slaney88]

[4] E' il processo attraverso il quale l'organo uditivo modifica la sua risposta a seconda dell'intensità della pressione sonora cui è sottoposto. E' esperienza comune che, dopo essere stati immersi in suoni a volumi elevati, le orecchie rispondano in maniera diversa alle sollecitazioni sonore: generalmente si può avere un calo dell'udito, e comunque una sensazione di "taglio" delle alte frequenze. Questo meccanismo, che è parte integrante del modello di Lyon, non è stato considerato nell'implementazione di Slaney.

convertito in impulsi nervosi, che quindi viaggiano attraverso il nervo uditivo fino al cervello. L'uscita di questo modello, quindi, è un vettore proporzionale alla frequenza degli impulsi nervosi dei neuroni in ogni punto della coclea.

Scendiamo ora nel dettaglio dell'implementazione. Questa combina una serie di filtri che utilizzano rettificatori a mezza onda (HWR, *half wave rectifiers*) per determinare l'energia del segnale, e diversi stadi di AGC. Tale struttura è mostrata in Figura 5.2.

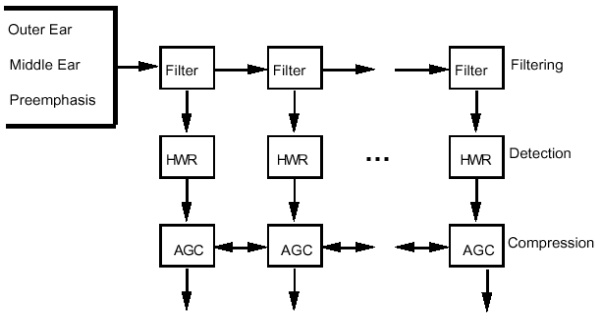


**Figura 5.2: Struttura del modello uditivo** [**Slaney99**]

IL suono che entra nell'orecchio esterno (il padiglione auricolare e la prima parte del canale uditivo) e medio, passa poi, attraverso la cosiddetta **finestra ovale** all'interno della coclea. Una volta nel condotto cocleare, le onde di pressione sonora si propagano attraverso la **membrana basilare**. La rigidezza di questa membrana varia lentamente lungo tutta la sua lunghezza, come conseguenza ogni punto della membrana risuona più fortemente se "colpita" da un'onda di pressione di una particolare frequenza. Parte di questo movimento è catturato dalle **cellule cigliate**, che convertono questi stimoli meccanici in stimoli elettrici per il nervo uditivo, che comunica direttamente con il cervello.

Un'importante caratteristica della coclea è che l'energia nelle onde sonore viene divisa per frequenze, ed ogni punto della coclea risponde meglio ad una determinata frequenza. In questo senso la coclea mappa il contenuto spettrale di un segnale in un dominio che potremo definire "spaziale".

Slaney ha scritto una serie di *scripts* per Matlab che implementano il modello di Lyon[5] [SlaneyWeb]. La peculiarità di queste implementazioni è che forniscono un modo nuovo per "visualizzare" il suono: il **correlogramma**. In Figura 5.3 si può vedere quale sia il percorso da seguire per arrivare alla creazione del correlogramma.

[5] A dire il vero gli *scripts* implementano vari modelli uditivi, molti dei quali specifici per il riconoscimento vocale [Slaney98].

Quest'ultimo è una rappresentazione tridimensionale, dove una delle dimensioni è il tempo: in pratica è un "film", una "animazione".

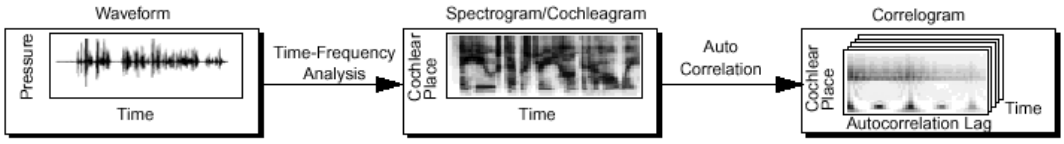


**Figura 5.3: Creazione del correlogramma [Slaney99].**

Da una forma d'onda, che sarà verosimilmente la rappresentazione di un evento sonoro, attraverso il modello cocleare di Lyon si arriva ad una rappresentazione grafica, uno **spettrogramma** che, in pratica, vuole essere una fotografia di quale sia la risposta della membrana cocleare, e che prende il nome di **cocleogramma**. In pratica nell'asse delle ascisse è rappresentato il tempo, in quello delle ordinate le frequenze relative ad ogni banco di filtri. Quindi ad ogni coppia tempo-frequenza è associato un valore (rappresentato dal colore che l'immagine assume) che indica quale sia la probabilità che un impulso, dovuto ad una certa frequenza, in un determinato istante, arrivi al nervo uditivo. Ovviamente, da una singola forma d'onda in ingresso, a seconda della sua durata, si ricavano più cocleogrammi.

A questo punto da ogni singolo cocleogramma si ricava, attraverso l'autocorrelazione, un *frame* del correlogramma. Ogni *frame* riporta nell'asse delle ascisse il *lag* temporale dell'autocorrelazione, ed in quello delle ordinate una trasformazione di quella che era la probabilità di generare un impulso neuronale, rappresentata anche nel cocleogramma.

Attraverso il correlogramma, quindi, si ha una rappresentazione grafica del fatto che ogni singola porzione della membrana cocleare risponde ad una ristretta banda di frequenze, centrata in una in particolare. Se un suono è periodico, il correlogramma mostrerà un picco nella posizione che corrisponde al ritardo della correlazione uguale al periodo di ripetizione. In pratica se un suono è un tono puro, tale picco si presenterà in corrispondenza di un valore $\tau$ il cui inverso è pari alla frequenza di quel tono puro. Allo stesso modo, in presenza di suoni più complessi, il picco rappresenterà la frequenza fondamentale… quasi sempre…

In Figura 5.4 si vedono più picchi, e non solo quello relativo ai 440 Hz ($\tau \sim 2.27$ ms) ma anche quelli relativi ai multipli di $\tau$, ovvero alle subarmoniche del segnale (220,

110, 55 Hz. etc…). Questo ce lo aspettavamo, visto che l'autocorrelazione mette in evidenza tutte le periodicità del segnale, e non solo quella minore.

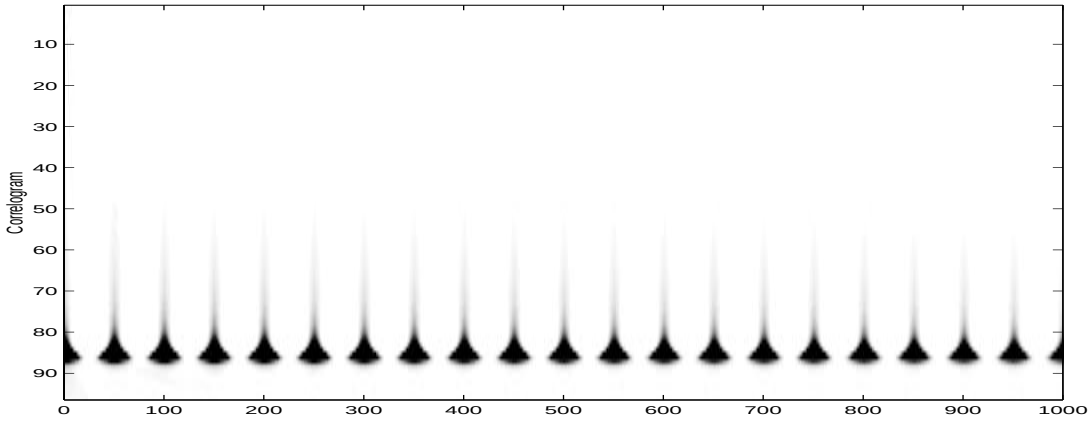


**Figura 5.4: Correlogramma di un tono puro a 440 Hz.**

### 5.1.2 La soluzione proposta: un rivelatore di *pitch* percettivo

La soluzione proposta da Slaney e Lyon è basata sul modello uditivo di cui abbiamo già parlato. C'è da dire che questa è l'unico caso analizzato in cui il segnale venga trattato sempre nel dominio del tempo, attraverso vari filtraggi e l'autocorrelazione. Come già anticipato, infatti, gli algoritmi che lavorino solo nel tempo, non si sono dimostrati particolarmente efficaci.

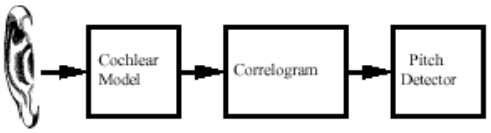


**Figura 5.5: I tre stadi utilizzati nel modello percettivo** [**Lyon-Slaney90**]

Abbiamo già parlato di come funzioni il modello cocleare di Lyon, che in ingresso ha un segnale audio, ed in uscita presenta il correlogramma. Proprio quest'ultima rappresentazione funge da ingresso al *pitch detector* implementato da Slaney e Lyon, per fare questo l'algoritmo combina le informazioni in tutti i canali del correlogramma per decidere un singolo *pitch*. In [Lyon, Slaney90] si dice che scopo dello studio è quello di trovare il *pitch* migliore, si vede, quindi, come questo studio sia molto distante dal proporsi di trovare una soluzione per la trascrizione automatica della musica polifonica.

Il correlogramma mostra chiaramente molti aspetti della percezione uditiva [Lyon, Slaney90]. Un suono vocale, ecciterà le varie parti della coclea in maniera diversa.

Ogni frequenza nella voce è modulata dalle corde vocali; la periodicità che attraversa tutti i canali della coclea, che è rappresentata nel correlogramma tramite una struttura verticale, è un indicatore del *pitch*.

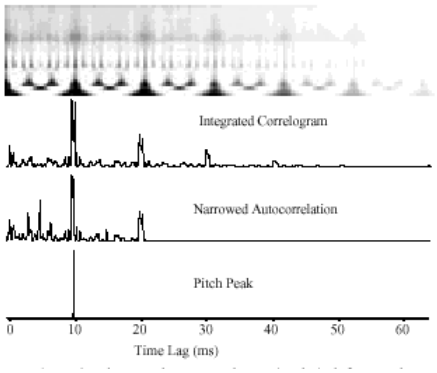


**Figura 5.6: Dal correlogramma al *pitch* per una *'u'* pronunciata da un uomo [Lyon-Slaney90].**

Il *pitch detector* di Lyon e Slaney è diviso in quattro parti: uno stadio di *pre-processing*, che modifica il correlogramma per esaltarne i picchi. I valori associati ad ogni lasso temporale del correlogramma così trattato vengono sommati su tutte le frequenze, in pratica si sommano i valori delle colonne del correlogramma, ricavando, da una matrice, un vettore. Il terzo passo consiste nel rendere la stima del *pitch* più robusta, considerando anche l'esistenza delle subarmoniche. Per finire il picco più forte è preso come *pitch* predominante. Il processo è descritto in Figura 5.6.

Per quanto riguarda il *pre-processing*, questo è diviso in due stadi: prima di tutto viene fatta una convoluzione tra il correlogramma ed un operatore che ne enfatizzi le linee verticali, successivamente si tratta il correlogramma in modo che siano enfatizzati i picchi che sono indicativi delle periodicità nel suono. Per fare questo si fa passare il correlogramma attraverso uno stadio di *enhancement* non lineare.

Ecco tutto il processo visto più nel dettaglio:

Il correlogramma $C(\tau,f)$ si integra sui canali (ovvero sulle frequenze) per calcolare una stima mono-dimensionale dei *pitches* presenti nel suono

$P(\tau) = \sum_i C(\tau, f_i)df$ , quindi $P(\tau)$ è funzione del lasso di tempo dell'autocorrelazione, $\tau$, e rappresenta la possibilità che un *pitch* di frequenza $1/\tau$ sia presente. Si definisce quindi una funzione particolare, detta NAC (*Narrowed Auto-Correlation Function*) che si calcola *da* $P(\tau)$.

$$P_N(\tau) = \sum_i^N (N-i)P(i\cdot\tau)$$

In assenza dello stadio non lineare di *enhancement*, la NAC si può considerare equivalente ad una versione modificata dell'autocorrelazione, data da:

$$C_N(f,\tau) = \int [R(f,t) + R(f,t+\tau) + ... + R(f,t+N\cdot\tau)]^2 \, dt$$

dove $R(f,t)$ è l'intensità istantanea con cui partono gli impulsi nel nervo uditivo, con una frequenza centrale $f$ al tempo $t$.

Per esempio un *pitch* di 100 Hz. mostrerà un picco in $P(\tau)$ con $\tau$ = 10 ms e le varie subarmoniche a 20, 30, 40ms e così via. La funzione NAC permette di considerare anche le subarmoniche quando si determina il *pitch*.

Il passo successivo è quello di scegliere, tra tutti i valori possibili, quale sia il *pitch* predominante. Secondo Slaney e Lyon, in genere, i picchi nella funzione di *pitch* $P(\tau)$ risultano essere abbastanza simmetrici, ed una stima accurata della loro posizione si può effettuare utilizzando un *fitting polinomiale* ai punti vicini al picco. Utilizzando più punti per determinare la posizione del picco, permette di poter calcolare il periodo relativo al *pitch* con una risoluzione migliore rispetto a quella che si avrebbe con l'intervallo (tra punto e punto) dovuto alla frequenza di campionamento, e consente una stima migliore e più robusta in situazioni rumorose.

Nel loro articolo Slaney e Lyon riportano i risultati ottenuti attraverso diversi esperimenti.

Il primo, in particolare, risulta molto interessante, in quanto mostra come il modello considerato emuli molto efficacemente il comportamento dell'organo uditivo: il *pitch* fondamentale può essere trovato anche in suoni che non presentino alcuna energia nella frequenza fondamentale. Una cosa simile la fa anche l'orecchio umano, o meglio, il sistema orecchio-cervello di un essere umano. In effetti, come detto nel capitolo 4, le note più basse del pianoforte non presentano energia nella fondamentale, non solo, le parziali che realmente hanno energia non trascurabile, in alcuni casi sono le terze o quarte. E' il nostro cervello che "ricostruisce" le fondamentali dalle parziali che percepisce. Come pure è vero che, specie gli orecchi più allenati, riescono ad associare un *pitch* anche a suoni "rumorosi", come tamburi, piatti o fruscii. Questo effetto si chiama *pitch* residuo (*residue pitch*). Quindi, per vedere come questo fenomeno fosse risolubile dal sistema, Lyon e Slaney hanno considerato un esempio di

*pitch* residuo, con sommato un rumore a basse frequenze, in modo che il rapporto segnale-rumore fosse SNR = -20 dB. Il *pitch* calcolato, istante per istante, si può vedere in Figura 5.7. Praticamente si tratta di estrarre un *pitch* da un suono che abbia un elevato rumore di fondo a bassa frequenza. Un effetto simile si verifica quando il rumore sia presente ad alta frequenza.

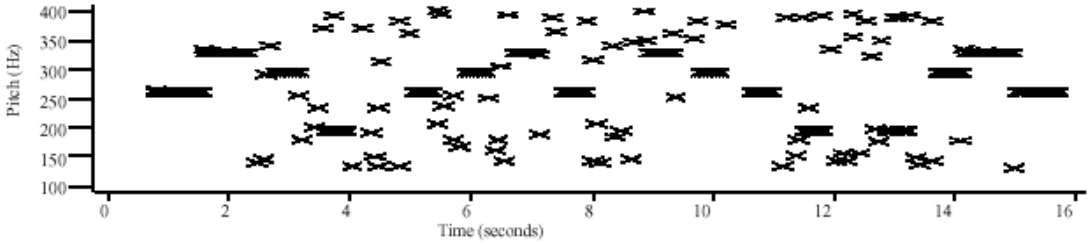


**Figura 5.7: esempio relativo al *pitch* residuo. In questo esempio la melodia delle campane di Westminster è suonata alternando un tono basso (solo la fondamentale) ed un tono residuo allo stesso *pitch*. Durante i 12 secondi centrali dell'esempio, viene applicato un filtro passa-basso, in modo che sia SNR=-20 circa. Il *pitch detector* non decide se sia presente un *pitch* valido oppure no, quindi nel tempo che intercorre tra i *pitch* residui, il *pitch* è determinato dal rumore [Lyon-Slaney90].**

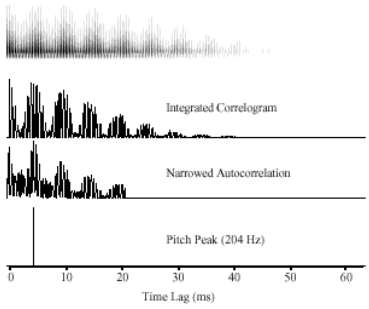


**Figura 5.8: Correlogramma e *pitch* percepito di un suono inarmonico [Lyon-Slaney90].**

L'esempio precedente si può spiegare con la presenza di una certa periodicità della forma d'onda filtrata, ma non si può dire la stessa cosa di suoni fortemente inarmonici. Si consideri un suono complesso, formato da tre toni, prodotto modulando in ampiezza (AM) un tono di frequenza 2kHz, con una modulante a 200 Hz. Questo suono sarà percepito come avente un *pitch* a 200 Hz, dato che i tre toni corrispondono alla nona, decima ed undicesima parziale di una fondamentale a 200 Hz[6]. Se si sposta la frequenza portante a 2040 Hz, i tre toni non formano più un gruppo armonico.

In prima approssimazione, il complesso sonoro è comunque percepito con un *pitch* pari a 204 Hz, come se il suono fosse ancora armonico, formato da nona, decima ed undicesima. Il risultato è visibile in Figura 5.8. Questo è un esempio lampante di come sia percepito un *pitch* che "non esiste", ricostruito dalle parziali presenti.

[6] In effetti, se la portante è a 2 kHz e la modulante a 200 Hz, il risultato sarà un suono che avrà, nello spettro, tre righe: una a 1,8 kHz, una a 2 kHz, una a 2,2 kHz. Ecco, quindi, che corrispondono rispettivamente alla nona, decima ed undicesima parziale di un suono con fondamentale a 200 Hz.

Se ora si lasciasse crescere lentamente la frequenza della portante, la gente componente un uditorio sentirebbe il *pitch* calare lentamente, per poi "rimbalzare" e crescere ancora. La lenta caduta del *pitch* percepito è un effetto noto come "effetto secondario". Gli autori ammettono di non saperlo sfruttare per un algoritmo robusto, pur essendo visibile nel correlogramma.

### 5.1.3 Risultati e conclusioni

Il *pitch detector* percettivo implementato da Slaney e Lyon, è un ottimo modello di quello che il comportamento dell'udito negli esseri umani, ma si devono tenere in considerazione diverse cose:

- il modello in questione non è stato implementato per la trascrizione automatica della musica, e risulta affidabile solo per stabilire quale sia il *pitch* fondamentale di un evento sonoro
- simula il comportamento dell'udito, ma ne ricalca anche i "difetti", si consideri, per esempio, il caso del *pitch* residuo.
- non risulta efficace nel caso di musica polifonica

Per quanto riguarda quest'ultimo punto in particolare, abbiamo fatto delle prove per verificare se il *pitch detector* percettivo fosse o meno utile ai nostri scopi. Abbiamo applicato questo sistema a suoni complessi, monofonici e polifonici. Più nel dettaglio, abbiamo considerato proprio note di pianoforte. Come si vede nelle figure che seguono, diventa sempre più difficile individuare le note mano a mano che il loro numero cresce. Si riesce comunque ad estrarre un *pitch* fondamentale, ma non è detto che abbia a che fare con le note presenti.

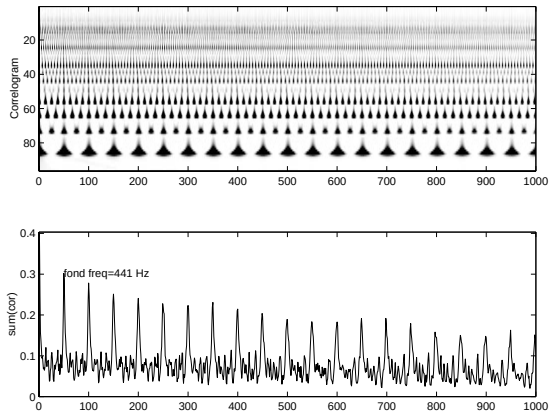


**Figura 5.9: Correlogramma e *pitch detection* di un $A_4$ di pianoforte. Il *pitch* è stato trovato correttamente e la visualizzazione è chiara.**

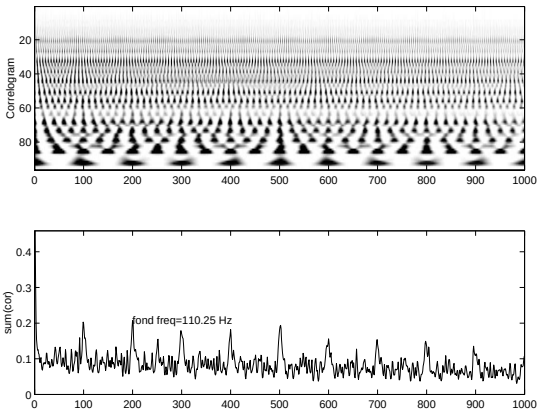


**Figura 5.10: Correlogramma e *pitch detection* di un bicordo formato da due note separate da un intervallo di terza minore ($A_3$ e $C_4$). Il *pitch* rivelato è relativo all'$A_2$.**

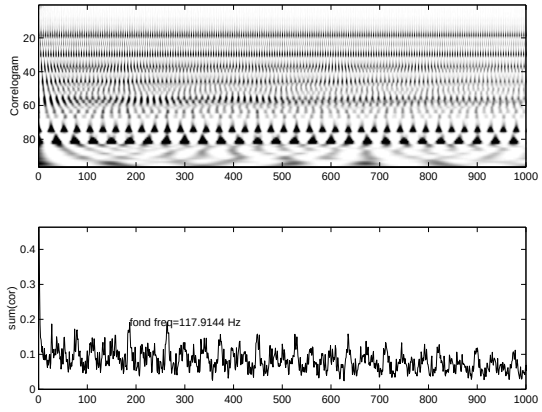


**Figura 5.11: Correlogramma e *pitch detection* di un accordo molto complesso ($A_2$+$C_3$+$E_3$+$D_5$+$G\#_5$). Il *pitch* rivelato è relativo ad un $A\#_2$, nota non presente nell'accordo. La visualizzazione non fornisce altre informazioni.**

A dire il vero, riferendoci alla Figura 5.10, lo sbagliare di un'ottava la nota trovata (viene trovato un $A_2$ quando in realtà è presente un $A_3$) risulta essere un errore abbastanza tipico, anche negli altri algoritmi analizzati. Il motivo principale che ci ha indotti a non utilizzare questo algoritmo come base del nostro trascrittore è che, pur trovando la frequenza fondamentale del suono, e pur ammettendo che questa frequenza possa essere la fondamentale di una delle note presenti nel suono, risulta davvero difficile pensare di andare oltre e trovare la nota successiva. Le visualizzazioni su cui si basa l'algoritmo del *pitch detector* percettivo, infatti, non risulta abbastanza chiara, in altre parole non sono deducibili neppure "ad occhio" le note successive.

## 5.2 Reti Neurali

Il lavoro analizzato affronta le problematiche legate alla trascrizione automatica della musica con le reti neurali. Come già detto, essendo la trascrizione automatica un problema legato, fondamentalmente al riconoscimento, è abbastanza istintivo il pensare di poter risolverlo attraverso le reti neurali, già usate con successo in altri problemi di riconoscimento (come il riconoscimento vocale).

### 5.2.1 Qualche veloce nozione sulle reti neurali

Le reti neurali sono [Fausett94] modelli matematici per l'elaborazione dell'informazione con caratteristiche di *performance* comuni (simili) a quelle delle reti neurali biologiche. Le **principali caratteristiche** sono:

- L'elaborazione dell'informazione avviene attraverso molte unità elementari dette **neuroni**.
- I segnali sono passati attraverso i neuroni tramite delle **connessioni** (l'equivalente informatico delle sinapsi biologiche).
- Ogni collegamento ha un "**peso**" associato, che, in una tipica rete neurale, amplifica il segnale trasmesso.
- Ogni neurone applica una **funzione di attivazione** (tipicamente non lineare) al suo ingresso (somma dei pesi dei segnali in ingresso) per determinare la sua uscita.

**Caratteristiche peculiari** che differenziano i vari tipi di rete neurale sono infine:

- Il ***pattern*** (modello) di interconnessione tra i neuroni, ovvero la sua **architettura**.
- Il metodo di determinazione del peso delle connessioni. Questo metodo è un vero e proprio allenamento (***training algorithm***), ed è per questo che si dice che le reti neurali sono sistemi che "imparano" e che si devono "addestrare".

- Per finire la **funzione di attivazione**, di cui si è già parlato.

In Figura 5.12 si può vedere quale sia la struttura di un neurone. I vari ingressi vengono pesati e sommati, il risultato di questa operazione viene passato attraverso una funzione che si occupa di stabilire quale sia la soglia (*threshold*) per cui il segnale debba essere passato all'uscita. L'uscita è una sola, anche se può essere passata a molti neuroni.

Consideriamo, a scopo esemplificativo, il caso di un particolare tipo di rete neurale: *feed-forward* network. I neuroni sono suddivisi tra tre diverse sezioni (*layers*):i neuroni che gestiscono i segnali in ingresso alla rete neurale, fanno parte della sezione d'ingresso (*input layer*), viceversa quelli che gestiscono l'uscita della rete neurale, sono parte ella sezione d'uscita (*output layer*). I neuroni che mettono in contatto queste due sezioni, non hanno nessun contatto diretto con il "mondo esterno", sono nascosti, per questo fanno parte dell'*hidden layer*.

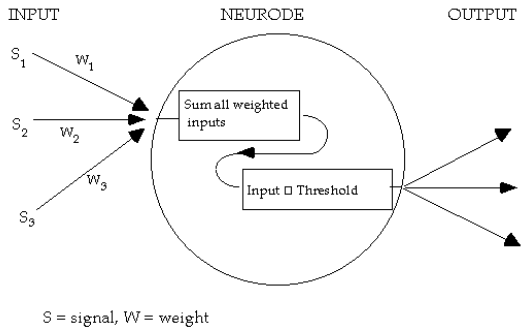


**Figura 5.12: Un neurone** [**sito4**]

Vediamo ora, con un gustoso esempio, come funziona il processo di decisione da parte di un singolo neurone. L'abbiamo scovato in un sito [sito4] che si occupa di applicare le reti neurali alle decisioni in ambito giuridico. Nell'esempio in questione, il nostro neurone deve decidere se concedere o no la libertà sotto cauzione (*bail granted*) ad un criminale, dopo aver pesato due fattori: se il criminale in questione abbia precedenti penali (*prior convictions*) e se il crimine commesso sia stato o no commesso in modo violento. Il neurone, risponde con uno "0" se decide che la libertà su cauzione sia ammissibile, o con un "1" se invece decide che non lo è. Si noti che entrano in gioco anche i pesi degli ingressi *w*, e la funzione di *threshold*, che si

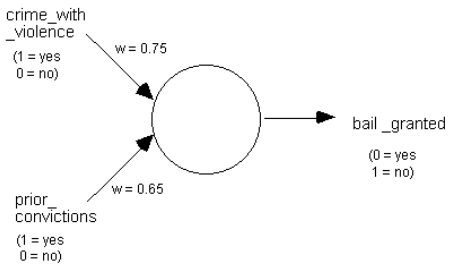


**Figura 5.13: Come decide un neurone** [**sito4**]

suppone essere all'interno del neurone (supponiamo che in questo caso sia un semplice troncamento della parte decimale). Nel caso in cui il crimine sia stato violento, ed il criminale abbia avuto precedenti penali, l'uscita sarà pari a

$$TRUNC[(1 \times 0.75) + (1 \times 0.65)] = TRUNC[1.4] = \underline{1}$$

che ci dice che la libertà su cauzione non sarà concessa. Viceversa, se il criminale ha precedenti penali, ma il crimine commesso non è stato violento, l'uscita sarà

$$TRUNC[(0 \times 0.75) + (1 \times 0.65)] = TRUNC[0.65] = \underline{0}$$

Nell'esempio appena visto, era assente l'*hidden layer*, mentre *input* ed *output layers* coincidevano. Vediamo, con un altro esempio [sito4], in cui i vari neuroni sono collegati in modo da avere distintamente un *input layer*, un *output layer* ed un *hidden layer*. E' il caso del *feed-forward network* dove tutti i *layers* sono presenti, dove l'uscita di ogni nodo finisce in ingresso a tutti i nodi del *layer* successivo e dove, come logica conseguenza, ogni nodo avrà in ingresso le uscite di tutti i nodi del *layer* precedente (vedi Figura 5.14). In questo esempio, Si deve decidere se concedere la libertà sotto un'alta cauzione (*high bail granted*), una bassa cauzione (*low bail granted*) o se non concederla affatto (*no bail granted*) dopo aver pesato gli ingressi, ovvero se il crimine commesso sia una violenza sessuale, se sia stato perpetrato con violenza e se il criminale in questione abbia precedenti penali. All'uscita della rete si avrà la decisione.

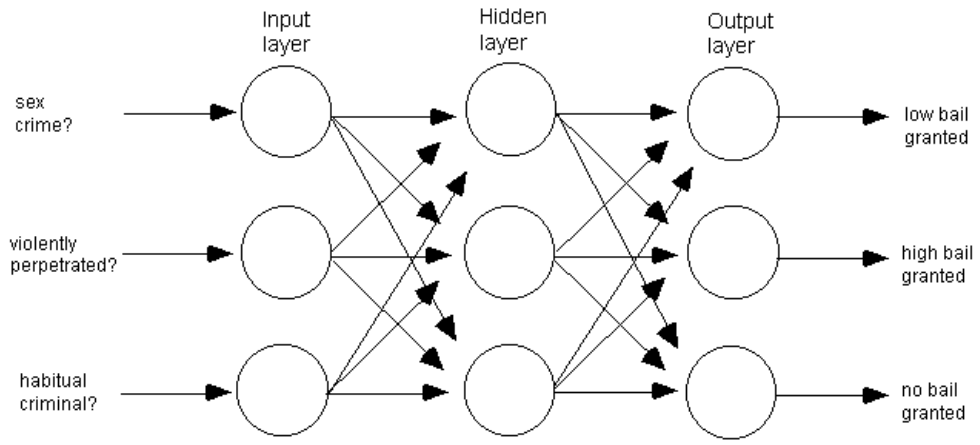


**Figura 5.14: Rete neurale del tipo *feed-forward*. Si notino *input, output* ed *hidden layers* [sito4].**

Ma cosa si intende col dire che le reti neurali vanno allenate? Non è lo scopo di questa tesi lo spiegare come avviene il *training* della rete, ma diremo, in modo veloce,

che una rete, come gli uomini, impara dai suoi sbagli e dagli esempi che le sono sottoposti. E' intuitivo il fatto che, avendo reti molto grandi, è impensabile il poter riuscire ad imporre i pesi giusti (*w*) agli ingressi "al primo colpo", e quindi, con un'altissima probabilità, questi saranno sbagliati, e la rete prenderà decisioni sbagliate, concedendo la libertà su cauzione ad un assassino con precedenti penali che abbia violentato sette minorenni, e non concedendola ad un poveraccio che abbia passato il semaforo con il rosso e che non abbia precedenti penali. Ecco quindi che, per allenarla, si dovranno sottoporre alla rete diversi casi giuridici di cui si conosca la soluzione, e si dovrà fare in modo che la rete confronti gli ingressi con le uscite e con le decisioni esatte, e che ritocchi in maniera adeguata i pesi agli ingressi dei suoi nodi. In questo modo la rete impara dai suoi errori e dagli esempi che le vengono sottoposti.

### 5.2.2 La soluzione proposta: reti neurali ed oscillatori adattivi

Matija Marolt, della Facoltà di Informatica dell'Università di Lubiana (Slovenia), presenta il suo lavoro attraverso due scritti [Marolt99,00] e le pagine dl suo sito [MaroltWeb]. Il lavoro non è ancora concluso, ed attraverso contatti epistolari intrattenuti con Marolt, siamo venuti a sapere che questi studi fanno parte della sua tesi di dottorato, cominciata due anni fa, e che verrà probabilmente pubblicata entro la fine dell'anno, esclusivamente in lingua Slovena, né ci è dato sapere se sia prevista una traduzione.

Gli scritti citati, comunque, risultano molto esaurienti, e nel sito [MaroltWeb] è possibile scaricare vari esempi di trascrizioni del suo *software*, per verificarne la bontà.

Per descrivere questo sistema crediamo sia utile seguire l'iter imposto dall'uscita degli scritti di Marolt, dove per prima cosa si propone una soluzione semplice di massima, che già offre buoni risultati [Marolt99], e poi si apportano modifiche più complesse ed efficaci [Marolt00].

In un primo approccio [Marolt99], si è costruito un sistema semplice, per la trascrizione di musica polifonica per piano solo. Il sistema cerca di determinare correttamente le note e l'istante in cui iniziano (*onset*) e comprende tre stadi principali.

In un primo stadio di *pre-processing*, si trasforma il segnale in tempo-frequenza. Quella scelta è una semplice trasformazione basata sulla correlazione, che permette

di scegliere arbitrariamente, per ogni banda di frequenza della trasformazione, la risoluzione sia nel tempo, che in frequenza. Viene scelta una divisione delle bande di frequenze di tipo logaritmico, di modo che queste siano spaziate di un semitono nelle basse frequenze, e di un quarto di tono nelle alte. La banda di frequenza più bassa è situata a 50 Hz, la più alta a 9000 Hz, la risoluzione nel tempo va dagli 80 ms per le basse frequenze, fino ai 10 ms per le alte. Per la finestratura si è usata una finestra di Hamming.

Dopo questo stadio, si entra nel cuore vero e proprio dell'algoritmo: un sistema formato da 88 reti neurali (del tipo *feed-forward*), ognuna delle quali è addestrata a riconoscere, in mezzo alle altre, una singola nota del pianoforte. Ogni rete neurale può avere come ingresso uno o più *frames* ottenuti dallo stadio di *pre-processing*, ma ha un solo *output*: un valore che, se maggiore di 0,5, indica la presenza della nota, se minore di 0,5, ne indica l'assenza. Quindi l'uscita da ogni singola rete neurale, un valore sempre compreso tra zero e uno, indica come ogni rete classifichi i suoi ingressi.

Il *post-processing* consiste in un algoritmo molto semplice di media temporale, per ridurre il rumore e per evitare che una singola attivazione di una rete (corrispondente ad un valore alto all'uscita) possa far fallire il sistema, che indicherebbe, erroneamente, quella certa nota come presente. Quindi molte attivazioni successive sono necessarie perché una nota sia dichiarata come presente. L'uscita finale dal sistema è l'elenco delle note e dei loro *on-set*.

Si sono testati vari tipi di reti neurali, li citeremo solamente:

**MLP** (*Multi-Layer Perceptrons*): è l'architettura più largamente usata nelle reti neurali, ed in questo caso ha ottenuto ottimi risultati.

**SVM** (*support vector machines*): Sono una classe relativamente nuova di reti neurali. I risultati ottenuti con le SVM sono paragonabili a quelli ottenuti con le TDNN.

**RBF** (*Radial Basis Function*): sono stati provati, ma la loro performance non è stata affatto buona. I risultati erano all'incirca del 5% peggiori rispetto alle MLP.

**TDNN** (*Time-Delay Neural Networks*): è stato l'unico tipo di reti neurali che avesse il tempo come parametro implicito. sono infatti sistemi *time-aware* ("consapevoli" dello scorrere del tempo), e sono quelli che hanno prodotto i risultati migliori.

Tutti i tipi di reti presi in considerazione sono stati addestrati con il medesimo set: circa 300.000 accordi con 1/3 dell'accordo stesso contenente la nota che la singola rete doveva riconoscere. Alla fine, i sistemi venivano testati con 5.000 accordi non presenti nel *training set*. Per le migliori prestazioni si sono scelte le TDNN.

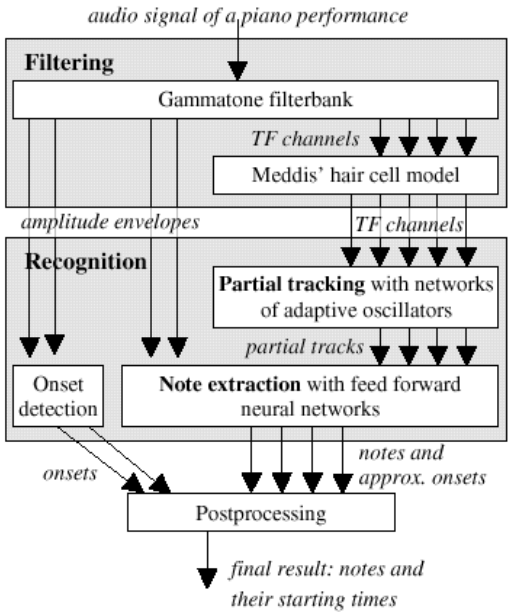


**Figura 5.15: Struttura del sistema** [**Marolt00**]

Il sistema è stato successivamente modificato [Marolt00] aggiungendo uno stadio di *tracking* delle parziali eseguito da reti di oscillatori adattivi accoppiati[7]. Altri cambiamenti che sono stati apportati, hanno reso il sistema più robusto ed hanno portato ad una maggiore accuratezza nei risultati. Tali cambiamenti concernono lo stadio di *pre-processing* (*filtering*), in cui il segnale subisce la già nota trasformazione tempo-frequenza. In [Marolt00] viene specificato meglio quale sia tale trasformazione. Si tratta di un modello uditivo, che emula il funzionamento della coclea umana. Il modello, prima di tutto, usa un banco di filtri *gammatone* per dividere il segnale in più canali. Il banco consiste in un *array* di filtri passa-banda, le frequenze centrali di questi vanno dai 50 ai 10000 Hz. Successivamente l'uscita di ogni filtro viene processata dal modello di Meddis[8], che emula il processo di trasmissione delle cellule cigliate dell'orecchio. Questo modello converte l'uscita di ogni filtro in una rappresentazione probabilistica dell'attività degli impulsi elettrici del nervo uditivo.

[7] Gli oscillatori adattivi sono una classe di oscillatori che adatta fase e frequenza in funzione degli ingressi.

[8] Il modello di Ray Meddis simula il comportamento delle cellule cigliate presenti nell'orecchio umano. Una implementazione di tale modello è presente nel toolbox di Slaney [Slaney98].

L'uscita dello stadio di filtraggio si può intendere come una serie di "canali di frequenze", ognuno dei quali contiene la rappresentazione degli impulsi generati (*firing activity*) dalle cellule cigliate dell'orecchio interno, ed i segnali costituiti da queste serie di impulsi si possono considerare quasi-stazionari. Come già detto, da questo modello, si può ricavare un correlogramma, rappresentazione di cui abbiamo già parlato, tuttavia Marolt sceglie un altro tipo di rappresentazione per ricavare la periodicità: l'uscita finale del modello sarà una stima della forza delle parziali con frequenza centrale (fondamentale) pari a quella delle note del piano che si vogliono riconoscere.Come detto per far ciò si avvale di oscillatori adattivi. Quando all'ingresso di uno di tali oscillatori si presenta un segnale periodico, questo cerca di sincronizzarsi al segnale aggiustando la sua fase ed il suo periodo. Osservando la frequenza di un oscillatore che si sia sincronizzato, si può fare una stima più accurata della frequenza del segnale in ingresso.

Un oscillatore ha tre variabili che cambiano nel tempo: fase, periodo ed ampiezza dell'uscita. La fase si definisce come

$$\phi(t) = (t - t_x) / p$$

dove $t$ è il tempo, $p$ il periodo di oscillazione e $t_x$ il tempo in cui l'oscillatore si aspetta un evento. Quando $t$ e $t_x$ arrivano ad eguagliarsi, la fase si annulla e l'oscillatore manda un impulso. Qualora all'ingresso sia presente un segnale periodico $s(t)$, l'oscillatore cerca di aggiustare fase e periodo a quello del segnale in ingresso, sicché gli impulsi mandati saranno sincronizzati con il periodo dell'ingresso. Fase e periodo sono aggiornati secondo le formule:

$$\Delta t_x = \eta_1 s(t) \frac{p}{2\pi} \sec h^2 \chi (\cos 2\pi\phi(t) - 1) \sin 2\pi\phi(t)$$

$$\Delta p = \eta_2 s(t) \frac{p}{2\pi} \sec h^2 \chi (\cos 2\pi\phi(t) - 1) \sin 2\pi\phi(t)$$

dove $\eta_1$ e $\eta_2$ sono dei parametri che determinano la forza con cui l'oscillatore si sincronizza allo stimolo, $\chi$ è un parametro fisso che definisce il campo ricettivo dell'oscillatore. L'uscita dell'oscillatore indica con quanta precisione si sia sincronizzato al segnale in ingresso: più l'uscita è alta, migliore è la sincronizzazione. Caratteristica di ogni oscillatore, inoltre, è la cosiddetta "frequenza (o periodo) preferi-

ta". Questa è la frequenza iniziale dell'oscillatore, ed un oscillatore può (in questo caso) sincronizzarsi a frequenze che siano un semitono sopra o sotto quella preferita: in questo modo si evita che vada alla deriva e si sincronizzi con frequenze qualsiasi, l'idea di fondo è, infatti, fare in modo che ogni oscillatore si possa sincronizzare con una sola frequenza.

Nel modello considerato, si è posta ogni uscita dello stadio di filtraggio (ovvero ogni uscita del modello delle cellule cigliate di Meddis, che abbiamo già deciso di chiamare "canale") in ingresso ad un oscillatore adattivo: un canale con frequenza centrale $f_0$, sarà posto in ingresso ad un oscillatore con frequenza preferita $f_0$.

Ecco, quindi, quello che succede: quando un segnale audio contenente una componente a frequenza $f_0$ viene fatto passare attraverso lo stadio di filtraggio: l'uscita del canale $f_0$ risulta quasi-periodica, e l'oscillatore che ha questo canale in ingresso inizia a sincronizzarsi, producendo un alto valore in uscita. Siccome gli oscillatori sono adattivi, possono sincronizzarsi anche se le frequenze deviano leggermente da quella preferita, o in caso di battimenti o leggere modulazioni in frequenza.

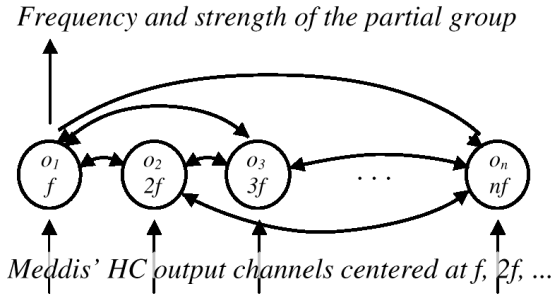


**Figura 5.16: Rete di oscillatori accoppiati** [**Marolt00**]

Ora, molti di questi oscillatori si possono mettere in relazione, visto che le loro frequenze preferite possono risultare in relazione armonica tra loro: ecco, quindi, che creiamo una rete di oscillatori. Più nel dettaglio, ogni rete consiste di dieci oscillatori, la frequenza preferita del primo oscillatore (frequenza di base della rete), coincide con quella fondamentale di una delle 88 note del pianoforte, le frequenze preferite degli altri oscillatori nella rete sono multiple della frequenza di base, ed ogni oscillatore della rete si può accoppiare internamente con tutti gli altri oscillatori (vedi Figura 5.16).

Una rete di oscillatori risulta avere due variabili in uscita:

- la **forza delle parziali del gruppo**, che si calcola come somma pesata di tutte le uscite degli oscillatori della rete. Le uscite vengono pesate in modo che o-

scillatori con una frequenza preferita più bassa abbia un impatto maggiore rispetto ad una con frequenza caratteristica più alta.

- la **frequenza media delle parziali del gruppo** che è calcolata come la media pesata delle frequenze degli oscillatori sincronizzati.

Tutta la rete funziona nel seguente modo:

- ogni oscillatore con una frequenza preferita *f* cerca di sincronizzarsi al suo ingresso, che proviene dal canale con frequenza centrale *f*, ed aggiorna la sua fase, frequenza ed uscita.

- ogni oscillatore sincronizzato aggiorna le frequenze e le uscite di tutti gli altri oscillatori della rete. Le frequenze sono aggiornate in modo che si avvicinino alla frequenza media delle parziali del gruppo, mentre le uscite vengono aggiornate secondo la formula:

$$o_j = o_j + o_j w_{ij} o^2{}_j e^{-1000(jf_i/(if_j)-1)^2}$$

dove $o_i$ ed $o_j$ sono le uscite rispettivamente degli oscillatori sorgente (quello che aggiorna) e destinazione (quello che viene aggiornato), $f_i$ ed $f_j$ le loro frequenze e $w_{ij}$ il peso dell'arco che va dall'oscillatore sorgente all'oscillatore destinazione.

Le uscite finali di tutte le reti di oscillatori sono 88 canali (uno per ogni rete), ognuno contenente la "forza" di un gruppo di parziali e la frequenza media di quel gruppo.

Ovviamente si è dovuto implementare anche un algoritmo che si occupasse dell'*onset detection*, quindi di stabilire quando inizia una nota. L'algoritmo risulta di semplice implementazione, ma molto efficace[9]. Prima di tutto si dividono le uscite del banco di filtri *gammatone* in undici gruppi parzialmente sovrapposti, ognuno contenente 12 canali, che coprono una intera ottava. All'interno di ogni gruppo i canali sono rettificati (*full-wave*) passati attraverso un filtro (con un solo polo e costante di tempo pari a 5 ms) e ne viene fatta la media.

[9] Trova circa il 99% di tutti gli *onsets* e ne riporta pochi di non presenti.

I picchi e la forza degli attacchi vengono infine stabiliti per ogni gruppo, e vengono scelti i picchi con un attacco più forte di 3 dB. Se tali picchi compaiono in quattro o più gruppi di canali contenuti in un *frame* di 50 ms, si trova un *onset*.

La parte del sistema che si occupa dell'estrazione delle note consiste, come già anticipato, di 88 reti neurali di tipo *feed-forward*, una per ogni nota del piano. Le uscite delle reti di oscillatori, passate attraverso un filtro (costituito da un unico polo, con tempo caratteristico pari a 25 ms), costituiscono la metà degli ingressi di ogni rete neurale. L'altra metà e costituita dagli inviluppi delle uscite dei filtri *gammatone*.

Ogni rete è addestrata a riconoscere la presenza di una ben precisa nota del pianoforte, di conseguenza ha un'unica uscita: un segnale alto indica la presenza della nota, un segnale basso ne indica l'assenza.

Le reti sono state addestrate con un *database* di ben 400.000 accordi di pianoforte, caratterizzati da una polifonia che andava da un'unica nota presente, fino a sei, in modo da addestrare ogni rete a riconoscere la sua nota caratteristica in mezzo alle altre. Ogni singola rete veniva addestrata con un set di circa 30.000 accordi.

### 5.2.3 Risultati e conclusioni

Il sistema è stato testato attraverso la trascrizione di diversi pezzi per piano solo, successivamente, per valutare più facilmente le prestazioni, come notazione si è scelto il formato MIDI. I pezzi scelti sono: il *Contrappunto n°12*, dall'*Arte della Fuga*, la *Partita n°1* (BWV825), e la *Suite Francese n°1* (BWV812) di J.S.Bach, l'*Overture* ed il *Valzer dei Fiori*, dallo *Schiaccianoci* di Tchaikowsky. I Risultati sono riportati in Tabella 3.

L'errore più frequente (più del 50% del totale) consiste nello sbagliare di un'ottava la nota trovata, riconoscendo, per esempio, un $C_4$ al posto di un $C_3$. Altri errori comuni sono il non riconoscimento di note molto brevi, ossia suonate molto velocemente, o in successioni molto veloci. Le *ghost notes*, infine, sono note non presenti, ma che il programma "crede"di trovare. In linea di massima si possono dividere in due tipologie: quelle che in realtà sono il prolungamento di una nota precedentemente trovata, ma che sono intese come note indipendenti da un errato *onset detection*, e quelle il

cui spettro può coincidere con la sovrapposizione delle parziali di altre due note presenti.

| Pezzo | Polifonia media | Polifonia massima | Note trovate | Note extra (*Ghost notes*) |
|---|---|---|---|---|
| **Contrappunto n°12** | 1.8 | 5 | 95% | 13% |
| **Partita n°1** | 2.6 | 6 | 94% | 15% |
| **Suite Francese** | 3 | 6 | 91% | 14% |
| **Overture Schiac.** | 3.1 | 6 | 90% | 15% |
| **Valzer dei Fiori** | 5 | 15 | 81% | 25% |

**Tabella 3: Risultati della trascrizione [Marolt00]. Nella seconda colonna è evidenziata la polifonia media, ovvero quante note mediamente sono presenti contemporaneamente del pezzo, nella terza colonna si evidenzia, invece, la polifonia massima, ovvero quale sia il numero massimo di note contemporaneamente presenti nel pezzo musicale. Nella quarta colonna è riportato il rapporto (in percentuale) tra le note trovate e le note totali presenti nel pezzo. L'ultima colonna, infine, mostra il rapporto (in percentuale) tra le note non presenti nel pezzo, che il sistema, sbagliando, individua, e le note totali.**

Futuri sviluppi per questo sistema possono essere, secondo Marolt:

- Delle connessioni di *feedback* che permettano di sopprimere dal segnale in ingresso le note già trovate, in questo modo si ridurrebbe il tipico errore "di ottava".

- Cercare di addestrare le reti neurali anche riguardo il contesto musicale in cui viene fatta la trascrizione[10].

- Lo stadio di *post-processing*, si potrebbe migliorare tenendo in considerazione non solo le uscite delle reti neurali e l'*onset detection*, ma anche altri fattori, come le parziali trovate, gli inviluppi delle ampiezze ecc…

[10] Non è spiegato bene cosa si intenda per "contesto musicale". Probabilmente Marolt vorrebbe addestrare le reti in modo che queste, riconoscendo il "tipo" di musica suonata, possano applicare particolari soluzioni di tipo probabilistico.

## 5.3 Analisi Spettrale

La soluzione presentata nel capitolo precedente, imponeva al sistema di trovare tutte le note presenti contemporaneamente in un medesimo *frame* temporale. Più precisamente divideva il sistema in 88 sottosistemi, ognuno in grado di determinare se una data nota fosse presente nel *frame*, e questi 88 sistemi lavoravano in parallelo. Ora supponiamo, invece, di implementare un sistema che trovi una nota alla volta, la più "presente", che la elimini e passi poi a trovare la nota successiva. Questo sistema può lavorare in maniera iterativa (vedi Figura 5.17).

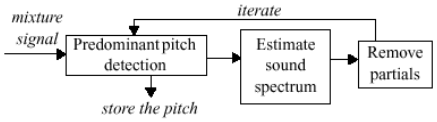


**Figura 5.17: Struttura del sistema** [**Klapuri-Virtanen-Holm00**].

### 5.3.1 Aspetti generali

La soluzione proposta che Anssi Klapuri, del Dipartimento di Informatica dell'Università di Tamperre (Finlandia) propone nella sua Tesi di Dottorato [Klapuri98], porta ad un sistema costituito fondamentalmente di tre parti. Un primo sottosistema si occupa di stabilire quando vi sia un evento, quindi quando inizi una nota, un altro sottosistema si occupa di trovare la nota suonata, infine l'ultimo la elimina dallo spettro del suono, ed il processo è iterato sino a quando si decida che non ci siano più note da trovare.

Questo sistema lavora sempre in frequenza, identificando le note esclusivamente dalle loro parziali, seppure applicando algoritmi complessi, che tengono conto dei teoremi enunciati nel Capitolo 3.2.

In comune con la soluzione precedente (e probabilmente con qualsiasi altra possibile) questo sistema ha il problema dell'*onset detection*. Il fatto di poter stabilire l'istante in cui avviene l'evento "nota suonata", infatti, esula dal tipo di soluzione proposta per stabilire quale sia la nota suonata. Per stabilire questo istante si effettua uno studio sulla variazione di energia lungo la retta temporale. Ecco il semplice algoritmo utilizzato:

1. Si filtra il segnale con sette filtri in parallelo[11].

2. Si divide ogni uscita in molti *frames* temporali, molto più piccoli di quelli che si useranno come finestre per la trasformata di Fourier. Si calcola l'energia di ogni *frame*.

3. A questo punto abbiamo sette vettori, $E_i(t)$ ad ogni istante corrisponde una frequenza: ne facciamo la derivata. La situazione, ora, è quella descritta in Figura 5.18, dove le righe tratteggiate corrispondono a $\frac{d}{dt}E_i(t)$, mentre le continue corrispondono a $\frac{d}{dt}(\log(E_i(t)))$. Le seconde evidenziano meglio i drastici cambiamenti dell'inviluppo caratteristici degli *onset*.

4. Si esegue una semplice operazione di *pick-picking*

5. Fissata una soglia, si eliminano i picchi che stanno sotto quella soglia

6. Ora si dividono i sette vettori in *frames* temporali di ampiezza adeguata (~50 ms) poi si eliminano, relativamente ad ogni frame, i candidati che risultano troppo vicini a candidati più prominenti, infine si sceglie quello che sta nel mezzo.

Un *onset* non è direttamente collegato ad una nota precisa, con questo intendo che ad un *onset* può essere associata una nota qualsiasi, così come un insieme di note, un accordo. L'importante è che si stabilisca un evento, un istante in cui iniziare la ricerca delle note presenti.

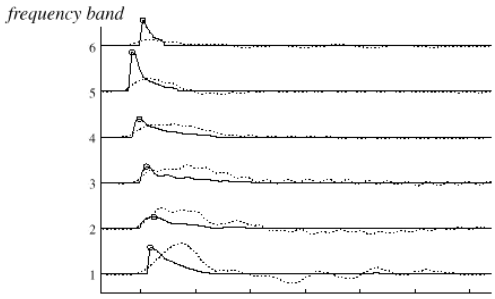


**Figura 5.18: Derivate degli inviluppi delle energie in sei bande differenti. Gli onset scelti sono cerchiati [Klapuri98].**

Questo sistema risulta molto valido, ma si deve tener presente che la cosa funziona bene per le caratteristiche percussive del pianoforte, che ha, infatti, ottime qualità di attacco, ed un transiente che dura poco.

[11] Il primo è un passa-basso, l'ultimo un passa-alto, gli altri cinque sono dei passa-banda. Le bande passanti dei filtri sono delimitate dalle frequenze: 127, 254, 508, 1015, 2032, 4064 Hz.

### 5.3.2 Identificazione di una nota

Consideriamo le parziali di *S* che occupino posizioni associate a numeri primi, denotiamole con {$h_p$ con $p$ numero primo}, $h_1$ è la frequenza fondamentale $f_0$; possiamo affermare che [Klapuri98] le parziali $h_p$ sono prove per l'esistenza del suono *S*, o indicatori di ogni sua proprietà che sia deducibile dalle sue parziali. Questa è una conseguenza del Teorema 4. Infatti ogni altro suono armonico può sovrapporsi a non più di una parziale si *S*. Come dire che se, oltre ad *S*, sono presenti altri tre suoni armonici che non siano sub-armonici di *S*, ecco che di tutte le $h_p$ di *S*, solo tre si sovrappongono a tali suoni.

In ogni modo, tali parziali possono essere completamente assenti, o avere, per qualche motivo, valori devianti rispetto alle caratteristiche del suono *S*. Le cause possono essere molteplici: per quanti riguarda il pianoforte abbiamo già parlato delle parziali mancanti, causate dal martelletto (vedi 4.2.2), ma i valori possono risultare "strani" anche a causa di rumore di fondo o risonanze. Quello che è importante affermare è che tali irregolarità sono, in ogni caso, isolate, e non sono valide per rappresentare il suono *S*. E' molto improbabile che un numero elevato di $h_p$ sia assente, o corrotto.

Le cose precedentemente dette stanno alla base dell'algoritmo considerato, poiché questo si basa proprio sull'analisi di tali parziali. Inevitabilmente, sorge un problema: se ci limitiamo alle $h_p$, vengono prese in considerazione troppo poche parziali. Questo fa in modo che l'algoritmo sia molto sensibile al contenuto tonale del suono. Si deve anche considerare che non tutte le parziali di un suono hanno la stessa importanza per riconoscere la nota: le parziali basse hanno un contenuto energetico generalmente maggiore (ovviamente anche qui ci sono le eccezioni) rispetto alle alte, e questo porta a due conseguenze: le prime sono più importanti per riconoscere il *pitch* di un suono, le seconde sono più soggette a degrado per interferenza con altre note. Ecco, quindi, che si è deciso di enfatizzare le parziali basse.

Klapuri ha costruito un filtro che, dal *file* sonoro, estraesse le informazioni sufficienti a ricavarne il *pitch*. E' necessario ora definire due concetti: la probabilità della scelta del campione *j-esimo* ($j^{th}$ *sample selection probability*) di un filtro $v$ è la probabilità che il campione $h_j$ in un insieme {$h_j$} sia scelto per essere l'uscita di $v${$h_j$}. Denotia-

mo questa probabilità con $P_s(j)$. La probabilità della scelta del *rank j-esimo* ($j^{th}$ *sample selection probability*) di un filtro $\upsilon$ è la probabilità che l'*i-esimo* valore più piccolo in un insieme ordinato $\{h_j\}$ sia scelto per essere l'uscita di $\upsilon\{h_j\}$. Denotiamo questa probabilità con $P_r(j)$.

Il tipo di filtro (non lineare) utilizzato fa parte della classe dei *median and order statistic filters*, la cui uscita è conseguenza di uno studio statistico del segnale filtrato[12]. Ecco le caratteristiche che devono soddisfare:

- dato un numero $N$ di suoni interferenti, questi possono disturbare le parziali, che devono venir scelte per l'analisi, solo con una probabilità $\lambda$

- il filtro deve utilizzare tutte le parziali del suono osservato, e deve utilizzarle quanto più possibile allo stesso modo, perché sia robusto ed applicabile a diversi tipi di suono. Prendere solo le $h_p$ non sarebbe soddisfacente

- il numero dei valori che porterebbero a sfalsare la statistica del filtro deve essere il più alto possibile.

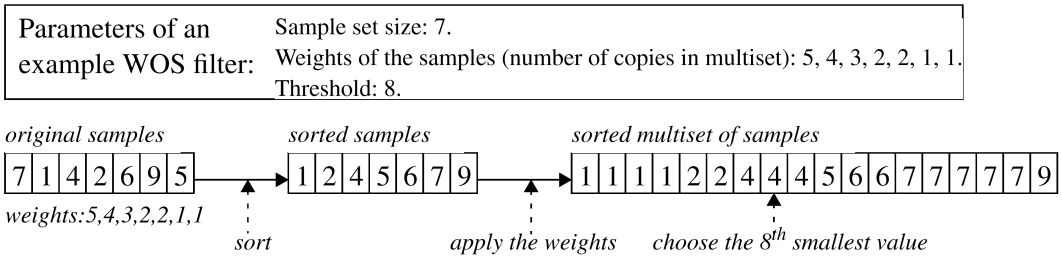


**Figura 5.19: Parametri e processo di calcolo di un filtro WOS. $\upsilon$={7, 1, 4, 2, 6, 9, 5}=4 [Klapuri98].**

Il filtro scelto è un WOS (*weighted order statistic filter*). La sua uscita è calcolata come indicato in Figura 5.19.

Tali filtri agiscono su *arrays* di campioni, nel nostro caso si può pensare che il vettore sia formato dai valori associati ad ogni frequenza. I campioni sono, per prima cosa, ordinati secondo il loro valore, successivamente ogni campione viene ripetuto

[12] Il filtro è stato implementato da Klapuri su algoritmi che Kousmanen ha presentato in Statistical Analysis and Stabilization of Stack Filters, Tech.D.Thesis (1994), Acta Polytechnica Scandinavia, Electrical Engineering Series.

proporzionalmente al suo peso (dato dai parametri del filtro), e quindi il *T-esimo* valore più piccolo viene scelto. *T* è la soglia (*Threshold*), ed esso stesso è un parametro del filtro.

Per ulteriori approfondimenti rimandiamo a [Klapuri98], la cosa da tener presente è che questo filtro fornisce in uscita un segnale dopo aver considerato diversi fattori: la probabilità che altre note si sovrappongano, quella che alcune parziali considerate siano corrotte, etc. Tutto questo si traduce nella funzione $P_s(j)$, di cui abbiamo già parlato. Questa finzione dipende fortemente da due parametri, $N$ (numero delle note interferenti presenti) e $\lambda$ (probabilità che queste note si sovrappongano, vale a dire che si sovrappongano le loro parziali).

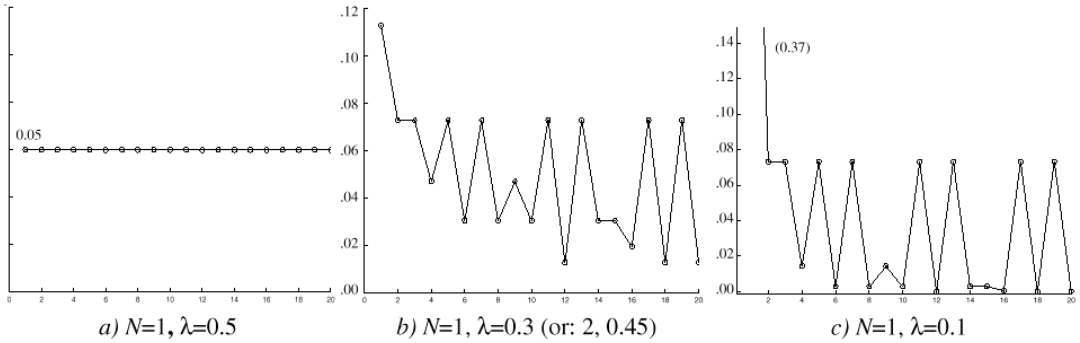


**Figura 5.20: $P_s(j)$ per venti parziali [Klapuri98].**

Come si vede in Figura 5.20, variando e $\lambda$ si varia molto la risposta del filtro, in pratica come il filtro consideri le varie parziali. $N$ è sempre pari a uno ciò vuol dire che si ipotizza che una sola nota interferente sia presente. In Figura 5.20b, si vede come la stessa risposta si può avere con due coppie diverse di $N$ e $\lambda$.

### 5.3.3 Applicazione alla Trascrizione Automatica di Musica Polifonica

Questo metodo passa attraverso un *training* del *software*, ma la cosa è molto diversa dal processo di *training* che abbiamo visto per le reti neurali. In questo caso si tratta solo di specializzare il *software* (e quindi la risposta del filtro) per il riconoscimento delle note del pianoforte, piuttosto che di altri strumenti. Il materiale usato per il *training* è costruito semplicemente da 88 campioni, uno per ogni nota del pianoforte. Una panoramica del sistema è presentata in Figura 5.17, uno schema più dettagliato

può, invece, essere analizzato in Figura 5.21. Come si vede, il corpo centrale del sistema ha tre ingressi:

- Un sottosistema che costruisce i *kernel* del filtro (*kernel creation process*)
- Un sottosistema che crea i modelli dei suoni (nel dettaglio note di pianoforte) che si devono trovare (*tone model creation process*)
- La registrazione della musica da trascrivere, in formato digitale (*musical recording*).

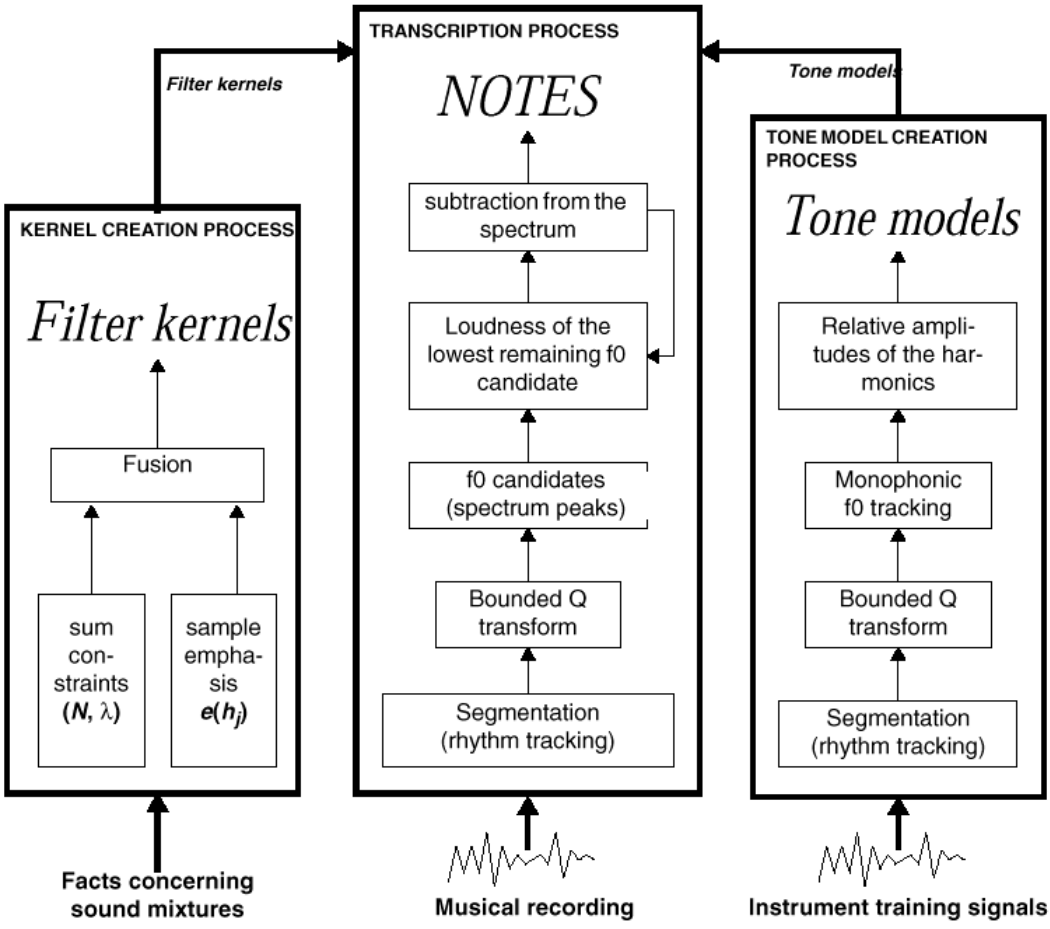


**Figura 5.21: Panoramica del sistema [Klapuri98]**

Il primo parte dall'ipotesi fondamentale che un solo strumento suoni nel materiale per il *training*, quindi comincia ad analizzare il materiale per determinare la frequenza fondamentale e la composizione spettrale di ogni singolo suono. Il secondo implementa gli algoritmi che costituiscono il filtro WOS (rimandiamo a [Klapuri98]).

Si deve porre l'accento sul fatto che il *training* va effettuato ogni qual volta si cambi lo strumento da analizzare, mentre il *kernel* del filtro WOS si può creare una volta sola.

In ingresso il *tone model creation process* ha un insieme di segnali monofonici, dove lo strumento è suonato in un numero di *pitches* e colori sufficiente a rappresentare tutto il *range* di suoni che possono essere prodotti dallo strumento in questione.

Il primo stadio (***rhithmic tracking***) segmenta il segnale, che ricordiamo essere monofonico, in modo da separare le varie note suonate[13], successivamente (***Bounded Q Transform***) il suono subisce una trasformata *Q*, dopo finestratura con finestra di Hamming. Questa particolare trasformazione ha il pregio di fornire una risoluzione logaritmica in frequenza. A questo punto si cerca la frequenza fondamentale[14] di ogni segmento (***Monophonic $f_o$ tracking***), che si può oramai identificare con una nota, e quindi si trovano le ampiezze delle varie parziali. Da qui si ricavano i modelli desiderati.

Ora vediamo come vengono trovate le note presenti in un *frame* temporale. Il suono, qui, subisce, in parte, le stesse trasformazioni di quello utilizzato per la costruzione dei modelli, ma qui la fondamentale $f_0$, deve essere estratta da un suono polifonico. Si tratta, quindi, di stabilire quale tra tutte le frequenze che sembrano rilevanti dopo la trasformata *Q*, sia effettivamente la fondamentale di una nota presente nel *frame*. Ecco l'algoritmo proposto:

1. Sia $\{C_i\}$ l'insieme dei picchi presi come candidati ad essere $f_0$, inserire in $\{C_i\}$ tutti i picchi dello spettro

2. Eliminare i candidati aventi *loudness* inferiore ad una soglia prestabilita, e tutti i picchi troppo vicini a candidati con *loudness* più grande

---

[13] Si deve pensare al segnale monofonico in ingresso come, per esempio, ad una scala cromatica, che suoni tutte le note del pianoforte. Per poterle analizzare, sono divise le une dalle altre. Per fare questo il *file* sonoro in ingresso è spezzettato.

[14] Considerando che si ha a che fare con un segnale monofonico, seppur non banale, non risulta difficilissimo trovare la frequenza fondamentale. Molto sbrigativamente si può considerare che il *pitch* sia pari alla frequenza del picco che nello spettro si presenta a frequenza più bassa, ma questo può non essere preciso, visto quanto detto sull'analisi spettrale del suono del pianoforte. Un'altra soluzione può essere quella di considerare quale sia la distanza "più frequente" tra due picchi successivi.

3. Trovare la frequenza precisa relativa ai picchi restanti

4. Aumentare i picchi aggiungendo a $\{C_i\}$ i picchi che distino un semitono da quelli già trovati[15].

5. (Opzionale) Se si considerano note molto basse (in termini di frequenza), si dovrebbe considerare un *pitch* che sia due volte più basso rispetto a $\{C_i\}$[16].

A questo punto l'insieme $\{C_i\}$ dei candidati è abbastanza ristretto. Si presti attenzione al fatto che le $C_i$ non sono note, ma parziali, righe nello spettro. Il procedimento per estrarre le note, da questo insieme di parziali è il seguente:

1. Si ispeziona $\{C_i\}$ in senso ascendente, a partire dalla parziale più bassa e si sceglie $f_0$

2. Si considera la parziale $f_0$ scelta e tutte quelle con frequenza multiplo di $f_0$, il sottoinsieme così trovato costituisce il "suono" $C$[17].

3. Attraverso una funzione di "peso" viene confrontato con il modello di tono

4. Il risultato passa attraverso il filtro WOS

5. Se $C$, dopo essere stato filtrato, supera una certa soglia di *loudness*, si decide che è la prima nota trovata.

Si noti che già al punto 2 di questo algoritmo, una parziale potrebbe essere scartata, nel senso che se non esistono abbastanza parziali multiple di $f_0$, è logico che la nota con fondamentale $f_0$ non esiste.

A questo punto, si deve eliminare la nota trovata, per poter così concentrarsi sulla successiva. Risulterebbe troppo "brutale" il semplice eliminare dallo spettro tutte le parziali contenute in $C$, questo perché, essendo molte parziali in comune tra più note, già dopo un passaggio il *file* risulterebbe troppo corrotto per poter trovare efficiente-

[15] Secondo Klapuri questo è un passo fondamentale, in quanto spesso questi picchi sono mascherati da quelli vicini [Klapuri98].

[16] Non è specificato, ma intuisco che in [Klapuri98] si intendesse "di due ottave più basso".

[17] Per ora, tale suono non ha ancora "dignità" di nota. E' semplicemente un suono, un insieme di armoniche.

mente altre note[18]. Ecco quindi che anche le parziali vengono eliminate solo parzialmente, e, per ogni parziale, la frazione eliminata dipende dal peso che si dà a quella parziale.

Ecco, quindi l'algoritmo di sottrazione, che tiene conto di due fattori: il modello di tono $T_i$, ed il peso $g_W(T_i,C)$ che si associa ad ogni parziale del suono $C$.

1. Inizializzazione: $j$=1

2. Si calcola la frequenza $f_j$ della parziale *j-esima* attraverso il semplice calcolo $f_j = f_0 \cdot (j+1)$ dove $f_0$ è la prima parziale del suono $C$[19]. Se ne calcola anche l'ampiezza teorica, moltiplicando il peso per l'ampiezza che tale parziale ha nel modello $g_W(T_i,C) \cdot g_A(f_i,T_i)$

3. Si costruisce un nuovo insieme $\{b_k\}$ formato da quelle frequenze abbastanza vicine ad $f_j$ da subirne l'influenza, nel senso che loro ampiezze sono influenzate da quella di $f_j$. La cardinalità di questo insieme si determina a priori, a seconda del tipo di finestratura. Con Hamming una cardinalità pari a due o tre è sufficiente.

4. Si fa il calcolo analitico della trasformata di Fourier (con finestratura di Hamming) $\{H_j\}$ della sinusoide a frequenza $f_j$ ed ampiezza pari a quella teorica trovata *precedentemente*, sui punti dell'insieme $\{b_k\}$. Praticamente questa è la quantità di ampiezza che le parziali hanno portato a questi punti.

5. Si sottrae $a \cdot \{H_j\}$ ai corrispondenti $\{b_k\}$ per rimuovere le sinusoidi dallo spettro. Discuteremo poi quanto debba valere $a$.

6. Se $j$ è inferiore alla posizione della parziale udibile più alta, si incrementa $j$ e si torna al punto 2.

---

[18] Basti pensare che (diretta conseguenza del teorema 3) se fossero presenti un $C_4$ ed un $C_5$, l'algoritmo troverebbe il $C_4$, come prima nota. Eliminando tutte le armoniche di $C_4$, si eliminerebbero anche tutte quelle di $C_5$, rendendone impossibile il riconoscimento.

[19] Si noti che in questo modo non si tiene in alcun conto l'inarmonicità, tipica del pianoforte e di altri strumenti.

Si noti che, nel punto 5, si sottrae dal suono una frazione della trasformata di una sinusoide. Questa sinusoide rappresenta la parziale a frequenza $f_j$. Ma quanto deve valere questa frazione $a$? Dipende da molti fattori. Se si è certi che non ci siano altre note presenti, si dovrebbe sottrarre tutta la parziale, quindi $a$=1, ma non ci è dato di sapere quante note siano presenti. Se pure si sapesse che, per esempio, fossero presenti due note, lo stesso non si potrebbe neppure vagamente ipotizzare come le parziali che si sovrappongono influiscano tra loro. Infatti stiamo lavorando con il modulo della trasformata di Fourier, ma non ci si dimentiche che, non considerando la fase, si perdono molte informazioni. Infatti le parziali, sovrapponendosi, possono, con probabilità del tutto uguali, rafforzare l'ampiezza del modulo per tale frequenza, così come possono annullarla.

### 5.3.4 Risultati e conclusioni

Klapuri presenta i risultati ottenuti tramite delle simulazioni effettuate su registrazioni di pianoforte[20], questo da un lato per le sue caratteristiche "percussive", dall'altro per l'irregolarità del suo suono, che rendono la prova abbastanza realistica. Il *range* di note andava da $C_2$ (65 Hz) a $C_7$ (2093 Hz). Le simulazioni presentate si dividono in due categorie: una prima si occupava di stabilire quante e quali note fossero presenti in un *frame* temporale, in regime stazionario, una seconda, invece, passava al regime tempo variante, facendo una trascrizione delle prime dodici battute di tre brani musicali: l'*Invenzione n°8* di Bach, *Für Elise* di Beethoven ed il *Rondò alla Turca* di Mozarth. Il capitolo relativo alle simulazioni è molto vasto, con molte tabelle che non ci sembra il caso di riprodurre integralmente; rimandiamo quindi a [Klapuri98] per maggiori approfondimenti sulle simulazioni, e qui riportiamo solo due tabelle, per dare un'idea dei limiti e dei risultati di questo lavoro. Per capire cogliere bene i risultati delle simulazioni, è bene dare uno sguardo a Figura 5.22.

[20] Anche se il suo *software*, con l'introduzione del suo *tone model creation process*, permette di trascrivere ogni strumento di cui si possegga un "segnale di addestramento".

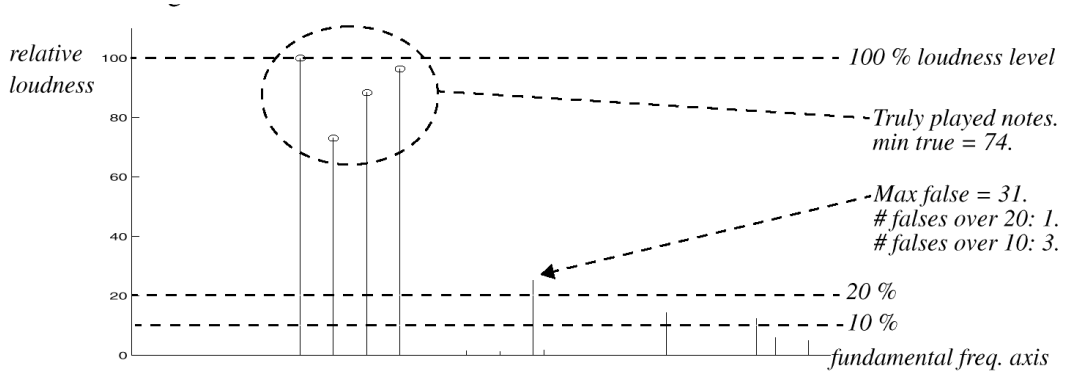


**Figura 5.22: Interpretazione dei risultati in una simulazione [Klapuri98].**

Tutte le note presenti in Figura 5.22 sono candidate ad essere presenti nell'accordo, poi si considera quale sia il valore minimo, in termini di *loudness* relativo, che assumono le note presenti, e quale il massimo di quelle non presenti, questo per poter trovare una soglia di decisione da applicare nel caso tempo-variante.

In Tabella 4 vediamo quali siano o risultati di una di queste simulazioni: il cercare le note presenti in un accordo formato da una triade più un ulteriore basso.

Nella seconda colonna vediamo quale sia il valore massimo, in termini percentuali, dello scostamento della parziale trovata da quella ideale. Questo algoritmo, infatti, non adegua il valore della parziale trovata a quello ideale della fondamentale che la dovrebbe rappresentare.

| **Basso: Accordo** | **Errore max valutaz. $f_0$** | ***Loudness* relativo** | | **Note vere non trovate** | ***Ghost notes* sopra il (*loudness*)** | |
|---|---|---|---|---|---|---|
| | | **Min. note vere** | **Max. *ghost n.*** | | **20** | **10** |
| **2:*c* 4:*Cmajo*** | 0.5% | 30 | 30 | 1 | 2 | 2 |
| **2:*c* 4:*Cmino*** | 0.9% | 23 | 18 | - | - | 2 |
| **3:*d* 5:*Dmajor*** | 1.1% | 59 | 8 | - | - | - |
| **3:*d* 5:*Dmino*** | 0.9% | 56 | 42 | - | 1 | 1 |
| **4:*e* 6:*Emajor*** | 1.5% | 42 | 13 | - | - | 1 |
| **4:*e* 6:*Eminor*** | 1.6% | 27 | 25 | - | 1 | 1 |
| **2:*g* 5:*Cmajor*** | 1.3% | 39 | 6 | - | - | - |
| **3:*e* 5:*Cmajor*** | 1.2% | 39 | 10 | - | - | - |

**Tabella 4: Risultati delle simulazioni: accordi con aggiunto un basso. 4:*e* 6:*Emajor* significa che l'accordo è un *Emajor* con $E_6$ ed il basso aggiunto è un $E_4$ [Klapuri98].**

Nella seconda e terza colonna vediamo quali sono i valori rispettivamente di minimo *loudness* per una nota realmente presente, e di massimo *loudness* per una nota falsa. Il *loudness* è espresso in termini relativi, quindi la nota più presente ha *loudness* pari a 100.

Nella quarta colonna vengono riferite le note non trovate. Queste note sono state eliminate dai possibili candidati dall'algoritmo presentato nel paragrafo precedente, e quindi non compaiono nemmeno con un basso *loudness*.

Quinta e sesta colonna, infine, presentano quante *ghost notes* vanno oltre valori di *loudness* pari a 20 e 10.

In Tabella 5 sono invece presentati i risultati delle simulazioni con brani musicali più o meno complessi, elencati nella prima colonna.

| **Composizione** | **Tot. note** | **Polifonia tipica** | **Note mancanti** | | ***Ghost notes loud.*> 25%** |
|---|---|---|---|---|---|
| | | | **Non trovate** | ***Loud.* < 25%** | |
| ***Inventio 8*** | 205 | 2 | 6• | 9•• | 4 |
| ***Für Elise (I)*** | 86 | 1 | - | - | 3 |
| ***Für Elise (II)*** | 190 | half:2, half:4 | 4 | 5 | 6 |
| ***Rondo alla Turca*** | 142 | 3 (up to 5) | - | 1 | - |

**Tabella 5: Risultati delle simulazioni: trascrizione di brani musicali polifonici [Klapuri98].**

Da *Für Elise* sono stati presi due diversi estratti, di dodici battute ognuno, il secondo estratto risulta più complesso, sia per la quantità di note presenti, che per la polifonia che lo caratterizza.

Nella seconda colonna è riportato il numero di note totali del pezzo.

Nella terza colonna la polifonia del pezzo. La prima parte di *Für Elise* risulta monofonica, la seconda ha una polifonia che varia da due (nella prima metà) a quattro (nella seconda metà. Il *Rondo alla Turca* è il più complesso, con una polifonia che varia da tre a cinque note.

Quarta e quinta colonna riportano rispettivamente quantità di note non trovate e di note con *loudness* troppo basso (inferiore al 25%).

La sesta colonna, infine, riporta il numero di *ghost notes*, con *loudness* superiore al 25%.

Analizziamo i risultati più nel dettaglio:

***Inventio 8, Bach***: come ci si poteva aspettare in un pezzo così semplice, quasi tutte le note sono state trovate e correttamente trascritte, tutte le note non trovate (•) mancano a causa di errori da parte dell'*onset detection*, che risulta, in questo caso, essere

l'anello debole della catena. Infatti se non si stabilisce che in un certo *frame* temporale non è presente un *onset*, in tale lasso non si cercherà nemmeno le note. Inoltre vi sono ben nove note con *loudness* che si può considerare troppo basso (••). Questo avviene durante la sequenza in cui sono alternate due note, ed il pianista, sistematicamente, suona la seconda più dolcemente.

***Für Elise (I), Beethoven***: questo pezzo (le prime dodici battute) è, in gran parte, monofonico, tuttavia la sua trascrizione non risulta banale, dato che contiene note basse e suonate molto velocemente. Il trascrittore si comporta, molto bene.

***Für Elise (II), Beethoven***: altre dodici misure. Queste contengono sequenze molto rapide, e molti tratti polifonici di quattro note.

***Rondo alla Turca, Mozarth***: questo pezzo è stato suonato da un computer, attraverso un piano elettrico. Gli ottimi risultati ottenuti, nonostante la complessità del brano, sono attribuibili alla mancanza di variazioni dinamiche, massicciamente presenti, invece, nei pezzi precedenti, suonati da un musicista su un pianoforte acustico.

I risultati sono notevoli anche in questo caso. Rispetto all'analisi effettuata con le reti neurali, possiamo affermare che questo approccio è più vicino al problema, più a basso livello.

## 5.4 Amazing MIDI™: un software commerciale


Dopo aver studiato diversi approcci, ed esserci fatta un'idea di come affrontare noi stessi il problema, abbiamo deciso di vedere se esistesse in commercio qualche prodotto che trascrivesse automaticamente la musica polifonica. Navigando la rete, senza nemmeno cercare troppo affannosamente, abbiamo trovato un *software* commerciale, la cui versione *shareware* era scaricabile gratuitamente da [AmazingMIDI$^{T-M}$Web].

### 5.4.1 Specifiche

AmazingMidi$^{TM}$ permette, secondo quanto riportato nel sito, di trascrivere (nel senso di estrarne un *file* MIDI) musica polifonica per qualsiasi strumento musicale. Non solo, sarebbe in grado di discriminare uno strumento tra molti altri, trascrivendo solo la partitura di quel particolare strumento.

La versione *shareware* in nostro possesso (V.1.60, rilasciata nel Gennaio 2000), permette di utilizzare tutte le proprietà del sistema, senza alcuna limitazione se non il tempo: si possono, infatti, trascrivere brani musicali della durata massima di trenta secondi. La copia scade e termina di funzionare dopo trenta giorni. Questo *software* supporta *files wave* mono o stereo, in formato PCM, con una risoluzione di 16 bit e frequenza di campionamento di 22.050 o 44.100 Hz. La velocità di elaborazione è tale per cui, con il nostro sistema[21], un brano di un trenta secondi viene trascritto in circa trenta secondi, tuttavia non lavora in tempo reale.

Il creatore di questo programma è Araki, Giapponese, il suo sito, da cui si scarica il programma, è completamente in Giapponese (esclusa la pagina dedicata ad AmazingMIDI$^{TM}$, che invece è in Inglese), e quindi non c'è dato di sapere alcunché di lui.

[21] Pentium II, clock a 266 MHz, 128 MB di RAM.

Niente sappiamo di quali siano le basi su cui poggia AmazingMidi™, ma si può notare qualche similitudine con il sistema di Klapuri. Anche in questo caso, infatti, oltre al *file* da trascrivere, si deve fornire al *software* anche un ulteriore *file wave* che riproduca lo stesso strumento. Questo è un procedimento necessario, visto che, da questo *file* "modello", vengono estratte le caratteristiche dello strumento da trascrivere, in modo da poterlo riconoscere anche in mezzo ad altri. Ricordiamo, infatti, che caratteristica di questo programma, dovrebbe essere quella di distinguere i vari strumenti. Tuttavia, se il pezzo da trascrivere fosse, per esempio, per piano solo, nulla ci vieterebbe di fornire lo stesso *file* sia come modello, sia come *input* per la trascrizione. Si noti, comunque, che il processo con cui si ricava il modello è completamente diverso da quello di Klapuri, basti pensare che il *file*-modello che AmazingMIDI™ fornisce di *default* per un pianoforte, è una semplice nota della durata di alcuni secondi.

### 5.4.2 Funzionamento

Il programma risulta molto semplice da usare, con un'interfaccia molto intuitiva. Si caricano i due *files wave* citati prima, e si lanciala trascrizione. A questo punto è possibile aggiustare dei parametri, che vanno ad intervenire sulla *performance* del trascrittore, ed in particolare della sensibilità con cui considera le variazioni di energia in frequenza, o il tempo minimo che può intercorrere tra due note successive. Si sceglie quale sia lo strumento che si vuole che suoni il nostro *file* MIDI e si lancia la trascrizione.

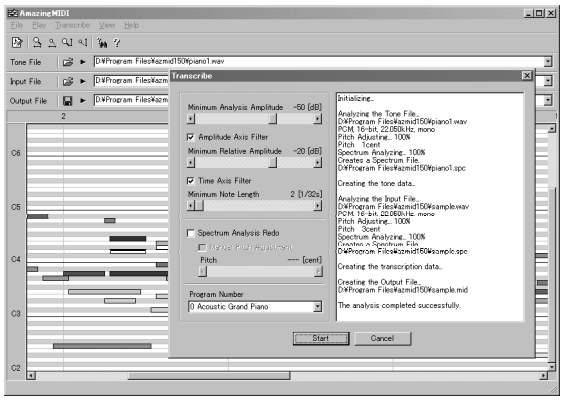


**Figura 5.23: Schermata di AmazingMIDI™**

L'uscita è un *file* MIDI, che sarà suonato automaticamente, dalla nostra scheda audio.

Una volta effettuata la trascrizione, il programma visualizza il *file* prodotto allo stesso modo di molti *sequencer*, con la classica grafica a barre. Le barre hanno un colore tanto più intenso, quanto più è alto il volume della nota ricavata.

### 5.4.3 Test

AmazingMIDI™ è stato testato in varie situazioni, le impostazioni del *software* sono state lasciate sui valori di *default*, si sono approntate due serie di test. La prima serie riguarda il semplice riconoscimento di note presenti in un *frame*, si è cominciato dalla singola nota e si è terminato con accordi complessi. La seconda serie riguarda la trascrizione di brani veri e propri, composti dalla successione di più *frames* temporali.

Per quanti riguarda il riconoscimento di note in singoli *frames*, si sono utilizzati suoni campionati da un pianoforte *Boesendorfer* da concerto. Purtroppo non sappiamo specificare meglio il modello e le misure.

| **Note** | **Note trovate** | **Errori** | | |
|---|---|---|---|---|
| | | **Note mancanti** | ***Ghost notes*** | **Ottava** |
| A#0 | G#2 | 1 (100%) | 1 | - |
| A2 | G#1 | 1 (100%) | 1 | - |
| A3 | A2 | 1 (100%) | | 1 |
| A4 | A4 | - | - | - |
| A5 | A5 | - | - | - |
| C5 | C5 | - | - | - |
| A2 A3 – ottava | A2 A3 G4 | - | 1 | - |
| A4 A5 – ottava | A4 | 1 (50%) | - | - |
| C5 C6 – ottava | C5 | 1 (50%) | - | - |
| C3 E3 – terza mag. | C3 E3 A5 | - | 1 | - |
| A3 C4 – terza min. | A3 C4 A4 C5 | - | 2 | - |
| A3 A#3 – semitono | A3 A4 A#4 F5 D6 | 1 (50%) | 3 | 1 |
| C3 D3 – tono | C3 D3 A#5 | - | 1 | - |
| A2 C3 E3 – Amin | A2 C3 G3 A#5 | 1 (33%) | 2 | - |
| A2 A3 A4 A5 | A2 A3 | 2 (50%) | - | - |
| C3 E3 G#3 – Cdim | C3 E3 G#3 G#4 | - | 1 | - |

**Tabella 6: AmazingMIDI™, riconoscimento note in *frames* temporali.**

Nella Tabella 6 si vedono i risultati dei test, che possono sembrare abbastanza sconfortanti se non si tiene conto che le combinazioni di note utilizzate sono volutamente critiche. Si noti, infatti, che gli errori nel riconoscere le note singole, avvengono quando queste appartengono alla parte bassa della tastiera, allo stesso modo il programma sbaglia spesso (forse un po' troppo spesso, a dire il vero…) quando ha a che fare con note separate da una o più ottave. La cosa che bisogna notare è la notevole presenza di *ghost notes*, che spesso si trovano al posto di note vere, e spesso, invece, oltre a queste.

Passando ora al caso tempo-variante, sono stati considerati diversi brani musicali. Per costruire gli esempi si è utilizzato il *soundfont* di cui si è già parlato, scaricabile da

[sito3]. I primi test sono estremamente semplici, si tratta di una ripetizione di un $D_4$, senza altre note presenti, e successivamente di una scala diatonica maggiore di C, che parte da $C_3$ per coprire tre ottave. Si tratta, quindi, di due casi di trascrizione di musica non polifonica.

Nel primo caso le note sono state trovate tutte, senza alcun errore di *onset*. Erano, comunque, presenti molte *ghost notes*, in particolare un $G_2$ con *onset* proprio all'inizio del brano, e che continuava per tutte le tre battute. Oltre a questo un $E_2$ che compariva qui e là, insistentemente.

Il secondo caso ha riportato anche più errori. Nelle prime due ottave della scala si sono riscontrate anche quattro *ghost notes*! Oltre a questo, sempre nella prima parte della scala (da $C_3$ ad $E_3$), le note erano sbagliate di un'ottava (errore abbastanza tipico). Non dimentichiamo, che è sempre molto critico riuscire a trascrivere la parte bassa della tastiera di un pianoforte.

L'ultimo esempio utilizzato è un estratto da *Prelude & Fugue in C major* (BWV 846) di Bach. Questo pezzo risulta ritmicamente molto semplice, è monofonico ma presenta delle difficoltà, giacché è basato principalmente su molti arpeggi, in cui le note, sono spesso tra loro in rapporto armonico. Nemmeno in questo caso il *software* si è comportato molto bene, con una presenza di addirittura tre o quattro *ghost notes*. A suo favore c'è da dire che non sbaglia a trovare le note che realmente ci sono, ma il numero di errori per quanto riguarda le *ghost notes* risulta davvero troppo elevato. Ci è sembrato inutile continuare le prove con brani più complessi.

Non molto si può dire a proposito di futuri miglioramenti di questo programma, la versione da noi considerata è datata Gennaio 2000, ed è la più recente disponibile. Nelle pagine di *download* non è specificato se e quando sarà pronta una nuova versione.

## 5.5 Confronti e conclusioni

Dobbiamo innanzi tutto precisare che solo con il modello uditivo e con AmazingMIDI™ abbiamo potuto fare delle prove, negli altri casi, non essendo disponibili né *softwares*, né listati da compilare, abbiamo dovuto accontentarci dei risultati riportati dagli autori nelle letture fatte.

Come già detto il modello uditivo non si presta, così com'è, a risolvere il nostro problema, anche se risulta essere parte integrante di altri approcci, ed è molto utile per l'analisi vocale.

Molto interessante è invece la soluzione proposta da Marolt, anche per i risultati che riesce ad ottenere [MaroltWeb]. Anche noi avevamo avuto la tentazione di usare le reti neurali, ma abbiamo abbandonato questo proposito per diversi motivi. Il primo motivo è che, con questa tesi, non si voleva solo offrire un algoritmo, un progetto che risolvesse il problema della trascrizione automatica, si voleva anche che questo sistema fosse basato sull'effettiva comprensione del problema stesso. Questo, con le reti neurali, non sempre è possibile. Le reti neurali si comportano spesso come una scatola nera, costruita a priori, a cui si fornisce un ingresso, da cui si ottiene un risultato, ma spesso risulta difficile non solo capire come correggere un errore di valutazione delle reti, ma addirittura il capire quale sia la causa di tale errore. Il secondo motivo era il tempo (molto) che avremmo dovuto impiegare per allenare e testare le reti neurali. Ciò detto, comunque, i risultati ottenuti percorrendo questa strada sembrano molto promettenti.

Non ci interessa analizzare AmazingMIDI™, visto che non sappiamo che studi ci siano alle spalle o che tipo di analisi esso effettui. I dati su questo programma sono stati riportati ad onor di completezza, essendo l'unico *software* commerciale del genere cui abbiamo avuto accesso. Lo terremo in considerazione solo per quanto riguarda i risultati, per un confronto con la nostra soluzione.

La soluzione di Klapuri, invece, oltre a riportare buoni risultati, è più vicina al tipo di approccio che volevamo. E' basata, infatti, su uno studio ed una conoscenza molto approfonditi dello spettro del suono, nonché delle relazioni che regolano i rapporti tra

le note. Questo è molto importante, perché permette di intervenire direttamente sul sistema, o su parti di esso, per correggere eventuali errori di valutazione.

Abbiamo deciso proprio per questo di usare un approccio simile a quello di Klapuri, seppure il nostro algoritmo risulti diverso.

Per questa scelta, quindi, non ci siamo basati esclusivamente su quali fossero i risultati delle simulazioni dei sistemi proposti, ma abbiamo fatto un compromesso tra risultati ottenuti (o verosimilmente ottenibili) e tipo di analisi effettuata.

# 6 La nostra proposta

**“Stupido relitto, lasciami partire per un viaggio che è cieco.**
**Le Sacre Speranze esigono il tuo didietro.”**
**P.K. Dick “*Divine Invasion*”**

Abbiamo già anticipato di come cercassimo una soluzione che fosse il più possibile alle radici del problema, che presupponesse una certa conoscenza e comprensione delle problematiche connesse, della struttura del suono e degli eventi a questo correlati. Abbiamo quindi deciso di affrontare il caso dello studio in frequenza. In Appendici 7.1 si possono trovare tutti gli *scripts* per Matlab utilizzati per implementare gli algoritmi e per le simulazioni.

## 6.1 Introduzione

L’approccio di massima è simile a quello di Klapuri, con uno studio in frequenza, che analizza il suono nella sua struttura armonica. Ma l’originalità della nostra soluzione sta nel fatto che è slegata da modelli predefiniti (almeno in una prima implementazione) e decide quali siano le note presenti non da parametri “locali”, come l’ampiezza delle singole parziali, ma da una grandezza “globale”, qual è l’energia.

Prima di tutto qualche definizione:

- $f_{j,k}$ è la ($j+1$)-*esima* parziale della nota *k-esima*, quindi $f_{0,k}$ è la prima parziale, la fondamentale della nota *k-esima.*
- $I_{j,k}$ è l’intorno di $f_{j,k}$ di raggio pari a $f_{j,k}\bullet(2^{1/24}-1)$ [1].
- $A(f)$ è l’ampiezza della riga dello spettro associata alla generica frequenza $f$.

[1] Questi intorni, davvero molto piccoli, sono tali per cui se centrati nelle armoniche prime di note che distino tra loro un solo semitono (la distanza minima nella musica occidentale) non si intersechino tra loro.

- $E(I_{j,k})$ è l'energia associata all'intorno $I_{j,k}$: $E(I_{j,k}) = \sum_{f \in I_{j,k}} A^2(f)$.

- $J = \{j \in \mathbf{N}$ che soddisfino una certa legge $L\}$, $L$ sarà definita di volta in volta.

- $E_k = \sum_{j \in J} E(I_{j,k})$ è l'energia che possiamo attribuire ad una singola nota, la *k-esima.*

Il procedimento utilizzato è iterativo, ed il suo schema di massima può essere così riassunto:

1. Delimitazione di una finestra temporale. Se tale finestra non è la prima deve essere immediatamente successiva alla precedente. Se si è arrivati alla fine del *file* da analizzare, si esce dal programma.

2. Normalizzazione del *file* sonoro e Trasformata di Fourier.

3. Se non è abbastanza probabile la presenza di almeno una nota, si torna al punto 1.

4. Estrazione della nota e memorizzazione (algoritmo *notefind*).

5. Eliminazione della nota trovata dal *file* (algoritmo *notekill*).

6. Ripetere dal punto 2.

L'idea che sta alla base della nostra soluzione è abbastanza semplice: partiamo dal presupposto che una gran parte dell'energia di una nota sia concentrata nelle parziali. Per la precisione, se noi consideriamo tutte le parziali $f_i$ di un singolo suono di pianoforte e calcoliamo l'energia associata ai loro intorni, di raggio pari a $(2^{1/24} - 1) \bullet f_i$, possiamo notare che tale energia è circa il 95% di quella totale![2].

Se sono presenti più note, con lo stesso procedimento si arriva alla medesima conclusione: attorno alle loro parziali si addensa la maggior parte dell'energia sonora. Ecco che, quindi, si può decidere se una nota sia o non sia presente in un *frame* temporale,

[2] Vedi Appendici 7.1 per lo *script* utilizzato per il calcolo, la nota considerata è un $A_4$.

calcolando l'energia associata alle sue parziali. E' su questo procedimento che si basa l'algoritmo *notefind*.

Per lo stesso principio è possibile eliminare l'energia associata ad ogni nota, per "pulire" il *file* dalla sua presenza e procedere quindi alla ricerca della nota successiva. Su questo si basa l'algoritmo *notekill*.

Ovviamente il procedimento non può e non deve iterare per sempre, è necessario decidere quando fermare la ricerca di nuove note, per evitare di voler estrarre note da un suono che, per esempio, contiene solo rumore. Non è un problema di facile soluzione, e spesso è molto difficile non incorrere in *ghost notes*.

Quando, poi, il problema si sposta dall'analisi del singolo *frame* temporale a quella di un pezzo musicale completo, si deve poter decidere anche quando inizia e quando finisce una nota, in altre parole risolvere l'*onset detection*.

## 6.2 Caso Stazionario: analisi di un unico *frame*

Analizzeremo la soluzione da noi proposta cominciando dal sottoproblema principale: riconoscere una o più note all'interno di un unico *frame* temporale. Il problema non è banale, giacché non è limitato solo al riconoscimento delle note, ma anche al decidere quante note ci siano all'interno di un *frame*, ovvero se l'energia del *frame* considerato sia sufficiente a giustificare la presenza di una o più note.

Lo suddividiamo, quindi, in due sottoproblemi: il primo si occupa di rivelare se nel *frame* ci possano essere delle note, o se l'energia contenuta in esso sia da considerarsi come rumore; il secondo, invece, partendo dal presupposto che nel *frame* sia presente almeno una nota, trova la nota più "presente". Cominceremo con l'analizzare il secondo.

Si noti, inoltre, che non è banale decidere quanto debba durare un *frame* temporale: può sembrare logico che, più piccolo è il *frame*, maggiore sarà la precisione con cui collocheremo gli eventi-nota sull'asse temporale, tuttavia non si deve scordare che la definizione, in frequenza, è inversamente proporzionale all'ampiezza della finestra di analisi. Tenendo presente che, se si considera le due note più basse di un pianoforte ($A_0$, 27.5 Hz e $A\#_0$, 29.135 Hz circa ) le frequenze fondamentali di queste distano

solo 1.5 Hz circa, si ottiene che, per riuscire a distinguerle, la finestra di analisi dovrebbe durare circa mezzo secondo. Ovviamente è una cosa improponibile. Quello che si è fatto è considerare finestre di analisi pari a circa 20 ms, e poi Fourier-trasformare dopo uno *zero-padding* che portasse la definizione in frequenza a livelli accettabili. Ecco di seguito le specifiche salienti:

- Suoni campionati a 16 bits e frequenza di campionamento pari a 22050 Hz
- Finestra di analisi di 512 punti, pari a circa 23 ms
- Ampliamento della finestra, prima della trasformata, fino a 16384 punti, pari a circa 0,7 ms, che garantisce una definizione in frequenza di circa 1,34 Hz

Ovviamente, seppure la definizione in frequenza è garantita, non è garantita la precisione nel valutare i valori delle $A(f)$, infatti questi sono solo approssimati tramite interpolazione cubica.

### 6.2.1 L'algoritmo *"notefind"*

L'algoritmo in questione è molto semplice, in pratica calcola l'energia associata ad ogni nota, e poi fornisce in uscita quella con energia maggiore.

Una prima implementazione di *notefind* funziona come segue, si consideri che $k$ è un numero che identifica un tasto del pianoforte (da 1, $A_0$ ad 88, $C_8$) e quindi una nota:

1. Sia $k = 1$
2. Si consideri $J = \{j=0, 1, 2, 3, 4\ldots jmax\}$, dove $jmax$ è tale per cui $f_{jmax,k}$ è la parziale più grande che può essere contenuta nella banda di segnale[3]
3. Si calcoli $E_k = \sum_{j \in J} E(I_{j,k})$
4. Se $k \neq 88$ si incrementi $k$ e si torni al passo 2, altrimenti si prosegua

[3] Non dimentichiamoci che, essendo il segnale campionato, la sua banda è limitata dalla frequenza di campionamento *Fs*, quindi $f_{jmax,k}$ sarà minore di *Fs/2*, mentre $f_{jmax+1,k}$ sarà maggiore di *Fs/2*.

5. Si scelga $E_i = max\{E_k\}$, $i$ rappresenta la nota trovata.

Già questa prima versione di *notefind* funziona molto bene con *frames* contenenti un'unica nota, sbagliando solo nel caso in cui le note presenti siano molto basse.

Si noti che la somma delle $E_k$ generalmente è maggiore dell'energia totale del *file*. Questo non ci deve stupire, infatti molte note hanno parziali in comune, e quindi può benissimo capitare che si calcoli due volte l'energia associata ad una medesima parziale. Non solo, sappiamo bene che se sono presenti due note tra cui vi sia un intervallo di quinta giusta, le parziali di queste coprono per il 60% le parziali della terza maggiore, e questo vuol dire, per esempio, che se in un *frame* fossero presenti un $C_4$ ed un $G_4$, noi assoceremmo un'energia non indifferente anche a $E_4$. Questo si deve evitare il più possibile.

Per risolvere, almeno in parte, il problema, ci viene in aiuto il Teorema 4, presentato in 3.2.4. Non considereremo, quindi, l'energia associata a tutte le parziali, ma solo a quelle che occupano una posizione indicata da un numero primo[4]: la prima (o fondamentale), la seconda, la terza, la quinta e così via (ovvero $f_0$, $f_1$, $f_2$, $f_4$, $f_6$, etc…). Chiameremo tale energia $E'_k$, dove $k$, al solito, è la nota considerata.

Ma i problemi non sono finiti. Consideriamo il caso in cui nel *frame* sia presente $A_5$: in questo caso l'energia di $A_4$ comprenderebbe quasi tutta quella di $A_5$, a causa del fatto che *notefind* considera tutte le tre prime parziali, e su queste è, generalmente, concentrata la maggior parte dell'energia.

Questo può sembrare un problema secondario, poiché, nell'ipotesi che si scelga la nota con energia maggiore, si sceglierebbe automaticamente $A_5$. Questo è un errore grossolano in cui è bene non incorrere. Si tenga ben presente, infatti, che noi stiamo lavorando con suoni in cui si presuppone che siano presenti più note, ed è, quindi, un'evenienza molto probabile che l'energia di $A_4$ sia incrementata da altre note presenti, o da rumore, fino ad avvicinarsi pericolosamente, se non superare, quella di $A_5$. Insomma, lasciando le cose come sono sarebbe molto facile incorrere nel classico errore delle *ghost notes*.

[4]Non si faccia confusione tra la posizione che occupa un'armonica ed il numero $j$ che identifica $f_j$.

Definiamo, quindi, una nuova grandezza, che nasce, in un certo senso, dalla definizione di energia, ma che non è un'energia vera e propria:

$$NRGk = \sum_{j \in J} E''(I_{j,k})$$

con $J = \{j \in \mathbf{N}$ *non maggiori di jmax tali che j+1 è numero primo*$\}$, dove *jmax* è tale per cui $f_{jmax,k}$ è la parziale più grande che può essere contenuta nella banda del segnale, ed $E''(I_{j,k})$ funzione che restituisce solo una parte dell'energia "condensata" attorno alla $f_{j,k}$, e la quantità di questa energia dipende proprio da *j*. Possiamo già anticipare che, tanto più *j* è grande (ovvero quanto più $f_j$ è distante dalla frequenza fondamentale), tanto minore sarà questa porzione di energia.

Ovviamente di funzioni di questo tipo ne esistono moltissime, si tratta di trovare quale sia la migliore o la più conveniente. Dopo molte prove abbiamo deciso di utilizzare una funzione che tenesse conto di come $E(I_{j,k})$ calasse all'aumentare di *j*, cosa che risulta evidente, nel pianoforte, per il fatto che l'ampiezza $A(f_j)$ delle parziali diminuisce all'aumentare di *j* (vedi Appendici 7.1). Abbiamo anche cercato di considerare il diverso comportamento dei tasti bassi rispetto agli alti. Il parametro *tastibassi* indica il numero di tasti per cui si applica un calcolo diverso. Consideriamo *tastibassi* pari a trenta (da $A_0$ a $D_3$ compreso).

Ecco quindi, come abbiamo deciso di valutare $E''(I_{j,k})$:

$$E''(I_{j,k}) = \begin{cases} E(I_{j,k})/\, j & se\ k \le tastibassi \\ E(I_{j,k})/\sqrt{j},\ j \neq 0 & se\ tastibassi < k \le tastibassi+14 \\ \begin{cases} E(I_{j,k}) \quad se\ j = 0 \\ \min\{E(I_{0,k})/\, j^3, E(I_{j,k})\},\ j \neq 0 \end{cases} & se\ k > tastibassi+14 \end{cases}$$

e quindi l'energia associata alla nota *k-esima* sarà la $NRG_k$ definita poco sopra.

Perché proprio questa scelta? Perché considera una porzione dell'energia delle parziali secondarie pari, al massimo, a quella che presupponiamo essere la quantità "ideale", rispetto all'energia della fondamentale. Se la fondamentale di una certa nota non si sovrappone a alcuna parziale, di nessun'altra nota, questo ci mette al riparo dal

sommare anche l'energia dovuta alle note che si sovrappongono alle parziali secondarie. Se, viceversa, è la fondamentale che si sovrappone, mentre le altre no, sommeremo solo l'energia "in più" aggiunta alla prima parziale, evitando, il più possibile, ulteriori sbagli sulle successive.

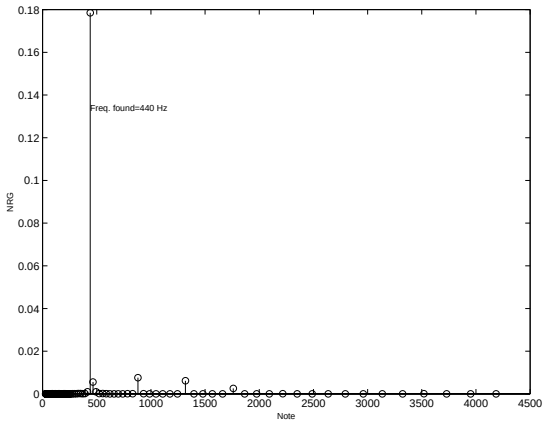


**Figura 6.1: Valutazione della funzione *NRG* per un *file* contenente solo un A4.**

In Figura 6.1 si può valutare l'efficacia dell'algoritmo. Si è applicato *notefind* ad un campione contenente un $A_4$, nell'asse delle ascisse sono riportate le frequenze fondamentali delle varie note.

Il risultato è notevole: se si considerano *files* contenenti una sola nota, *notefind* non commette alcun errore, eccezion fatta per le prime due ottave, dove gli sbagli sono molti. In particolare la tipologia di errore più frequente è lo sbaglio di un'ottava, ove, per esempio, prende un $C_2$ per un $C_3$.

Il compito di quest'algoritmo è di trovare tutte le note presenti in un *frame*, ma per farlo, ha bisogno di un altro strumento, un algoritmo che elimini le note trovate.

### 6.2.2 L'algoritmo *"notekill"*

Riprendiamo le considerazioni fatte per la soluzione di Klapuri: se noi eliminassimo tutte le parziali di una nota, corromperemmo troppo il *file*, se noi ne eliminassimo troppo poche, falseremmo la ricerca delle note successive.

Crediamo, quindi, che questo sia davvero un punto cruciale nel riconoscimento delle note presenti, e che debba essere perfezionato e personalizzato secondo lo strumento che si vuole analizzare. Non è facile trovare una funzione che si comporti bene in tutti i casi possibili, sicuramente la definizione di *notekill* sarà più problematica di quanto non lo sia stato quella di *notefind*.

Si deve considerare che la nostra è un'analisi fatta nello spazio delle frequenze, e che, inoltre, non valutiamo tutte le informazioni possibili, in quanto operiamo sempre

con il modulo della trasformata di Fourier, perdendo la parte di informazione data dalla fase. Questo è un fatto non trascurabile. Qualora, infatti, $A(f)$ sia la risultante del contributo di due parziali appartenenti a due note diverse, $f_{j1,k1}$ ed $f_{j2,k2}$, non è possibile stabilire in che misura queste contribuiscano, si può solo affermare che $A(f) = a_1 \bullet A(f_{j1,k1}) + a_2 \bullet A(f_{j2,k2})$, con $a_1$ ed $a_2$ reali. L'unica ipotesi che possiamo fare, è che, considerando che le prime parziali di una nota siano ad energia maggiore delle successive, con buona probabilità[5] ad $A(f)$ contribuiscano maggiormente le parziali più vicine alla fondamentale, quindi le parziali più vicine alla fondamentale andranno attenuate di più, rispetto a quelle più lontane.

Altra considerazione: mentre con *notefind* si analizzavano solo le $f_j$ con $j$ primo, se vogliamo eliminare una nota, dovremo eliminare tutte le sue parziali, o almeno quelle più forti, e dunque, le prime.

Detto questo, si deve sottolineare che le parziali appartenenti ad una nota che vogliamo eliminare, non vanno annullate completamente, l'alternativa sarà, quindi, un'attenuazione. Tale attenuazione dovrà tener conto della posizione delle parziali, visto che mentre l'energia di $f_{0,k}$ si può imputare per la maggior parte alla nota $k$, così non si può dire, ad esempio, per $f_{23,k}$.

L'algoritmo *notekill*, purtroppo, è da intendersi empirico, nel senso che siamo arrivati alla sua definizione partendo dalle considerazioni fatte fin'ora, ed affinandolo attraverso prove. Queste prove non hanno coperto tutti i casi possibili, e quindi potrebbe essere che *notekill* non funzioni a dovere con combinazioni di note non considerate da noi.

Ecco come funziona *notekill*: dato il modulo della FFT di un generico *file* contenente un insieme di note, e data la frequenza della fondamentale della nota che si vuole eliminare, *notekill* attenua le frequenze contenute negli intorni $I_{j,k}$ di raggio $(2^{1/24} - 1) \bullet f_j$:

1. Sia $k$ il numero identificativo della nota da eliminare.

2. sia $J = \{ j=0, 1, 2, 3, 4\ldots jmax' \}$, con $jmax' < jmax$.

[5] Poniamo l'accento sul fatto che questo è vero "con buona probabilità".

3. Si attenui lo spettro nel seguente modo: per tutti gli elementi di $J$

$$A(f) = \begin{cases} A(f)/8, f \in I_{0,k} \\ A(f) \cdot [1 - (1/\sqrt{n+1})], f \in I_{j,k}, j \geq 1 \end{cases}$$

Vogliamo sottolineare il fatto che questi algoritmi hanno una forte base empirica.

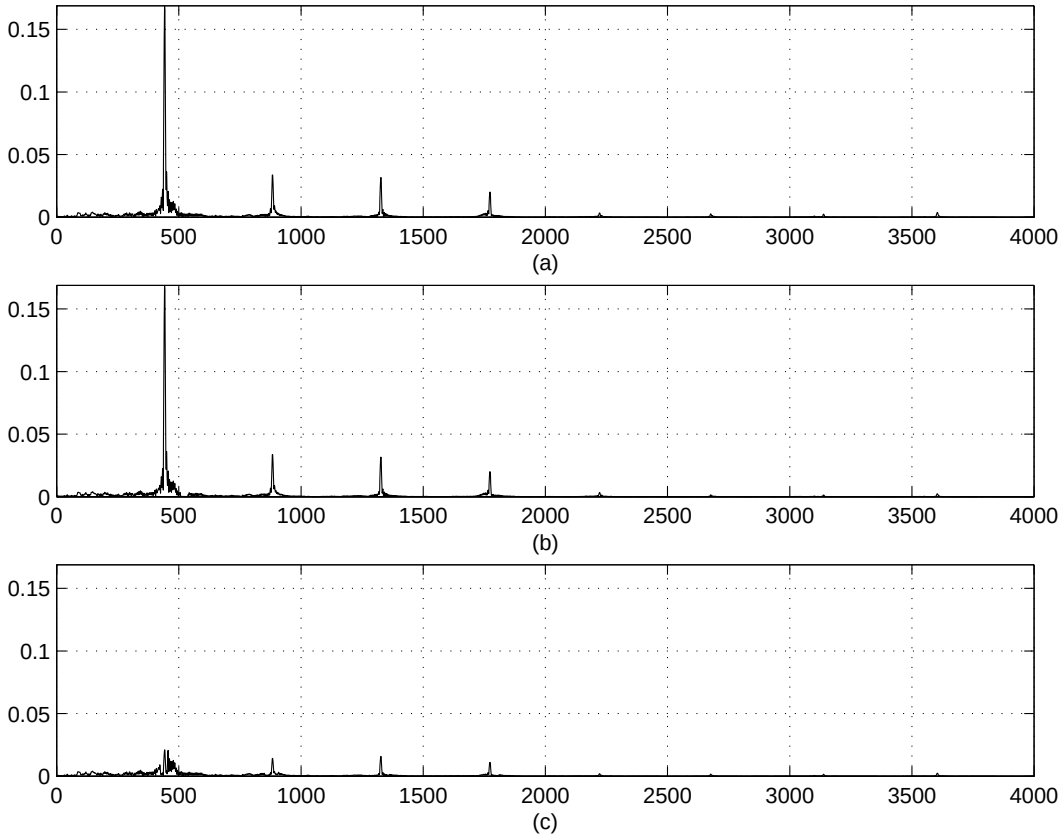


**Figura 6.2: (a) lo spettro di $A_4$, (b) dopo l'eliminazione di una nota diversa da $A_4$, (c) dopo l'eliminazione di $A_4$.**

In Figura 6.2 si vede lo spettro del suono di $A_4$ dopo l'eliminazione di $A_4$ (c) e dopo l'eliminazione di un C5, nota diversa da $A_4$ (b). Dalla stessa immagine sembra che l'eliminazione di una nota diversa da $A_4$ non intacca più di tanto lo spettro di $A_4$. Questo è vero, ma si deve considerare che la nota eliminata è un $C_4$, ed è pur vero che questa è la terza minore, ma non ha tanta energia in comune con $A_4$ quanta potrebbe averne, per esempio, una nota che dista una o più ottave da $A_4$, come $A_3$ o $A_5$. La cosa sembra funzionare bene, ma il caso polifonico presenterà ulteriori problemi…

In Figura 6.3, invece, vediamo un la rappresentazione di un test effettuato. Nel *file* erano presenti un $A_4$ (440 Hz) ed un $A_5$ (880 Hz): *notefind* trova prima l'$A_4$, *notekill*

lo elimina, e successivamente si trova ed elimina anche l'$A_5$. L'energia posseduta dal *file* finale è circa il 5% di quella posseduta dal *file* originale.

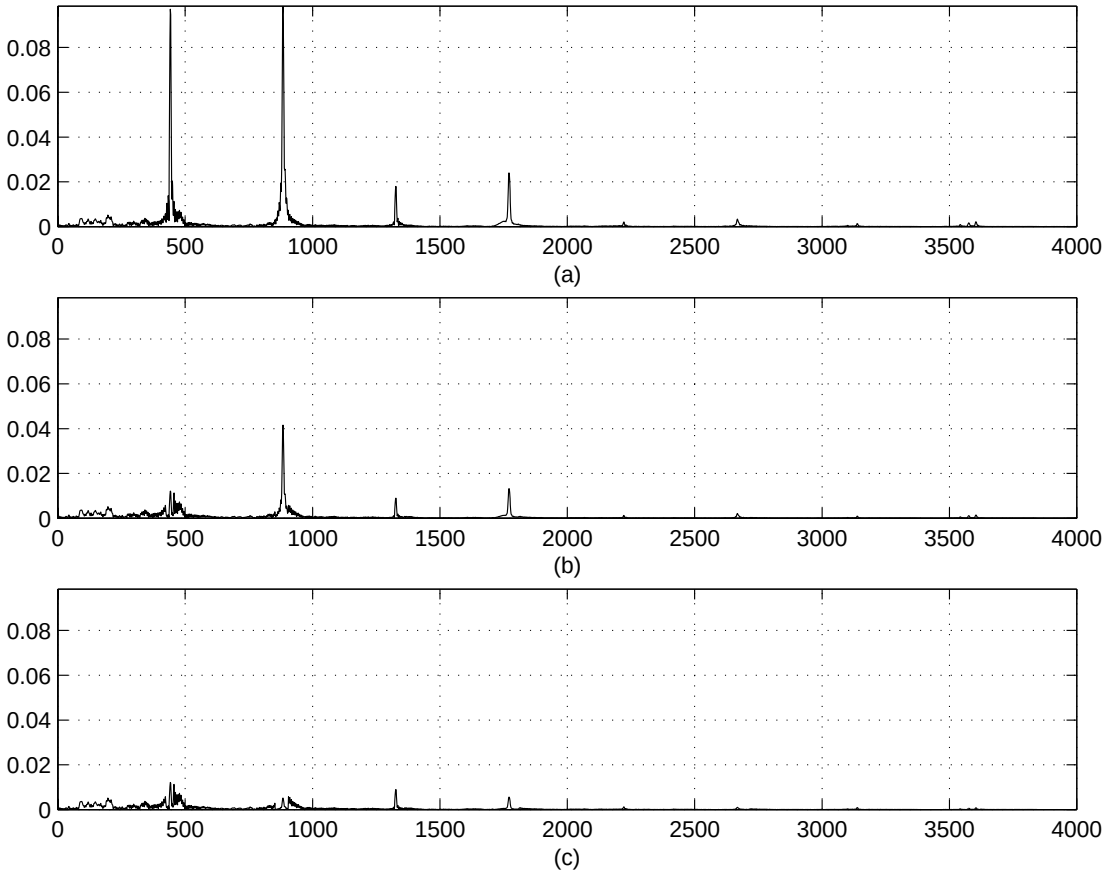


**Figura 6.3: Test. (a) *file* originale contenente $A_4$ ed $A_5$ (b) dopo aver eliminato $A_4$ (c) dopo aver eliminato $A_5$.**

### 6.2.3 Quante note sono presenti in un *frame*?

Questo è un grosso problema, risolverlo equivale a stabilire quanta energia deve avere un *file* perché sia abbastanza probabile la presenza di una nota ulteriore. Questo perché noi andiamo a cercare la *n-esima* nota dopo averne eliminate *n*-1. E' importante stabilire il passaggio giusto in cui fermarsi, onde evitare di trovare *ghost notes*, o, al contrario, non trovare note realmente presenti.

Tuttavia nel paragrafo precedente abbiamo stabilito che, dopo aver eliminato le note presenti, l'energia totale si riduceva a circa il 5%, quindi, per esempio, potremmo stabilire che, se l'energia rimasta dopo l'eliminazione di una nota è inferiore al 5% rispetto a quella iniziale, non ci sono altre note da trovare. Ma questa ipotesi si può considerare plausibile? No, ripetendo la simulazione con altre due note, le percentua-

li variano di molto. Considerando, infatti, un bicordo composto da $C_5$ e $C_6$, l'energia finale è il 14% di quella iniziale, con $A_3$ e $C_4$ è il 19%, con $C_3$ ed $E_3$ il 26% e così via.

Sicuramente l'energia finale di un *file* non dipende solamente da quante note siano state trovate, ma anche da quali siano queste note e quali intervalli le separino, tuttavia non siamo riusciti a trovare una relazione matematica, che potesse aiutarci.

Dopo aver effettuato alcune prove abbiamo deciso di fermarci qualora l'energia residua sia inferiore al 20%, e, comunque, dopo aver trovato al massimo cinque note (per impedire indesiderabili *loops* infiniti).

### 6.2.4 Risultati e conclusioni

Abbiamo effettuato dei test per vedere come il nostro *software* funzionasse, in Tabella 7 sono inseriti i risultati di tali test.

| **Note presenti** | **Note trovate** | **Errori** | | |
|---|---|---|---|---|
| | | **Note mancanti** | ***Ghost notes*** | **Ottava** |
| A2, A3 ottava | A3+A4 | 1 | - | 1 |
| A4, A5 ottava | A4+A5 | - | - | - |
| C5, C6 ottava | C5, C6 | - | - | - |
| A3, A#3 semitono | A3, A4, A#4 | 2 | 1 | 1 |
| A4, A#4 semitono | A4, A#4 | - | - | - |
| C3, D3 tono | A4, C3, D3 | - | 1 | - |
| A3, C4 terza minore | A3, C4 | - | - | - |
| C3, E3 terza | C3, E3, F3 | - | 1 | - |
| C3, G#3 quinta aumentata | G#4, C3, G#3 | - | 1 | - |
| A2, C3, E3 Amin | C3, E3, A3, F3 | 1 | 1 | 1 |
| C3, E3, G#3 Cdim | G#4, C3, E3 | 1 | - | 1 |
| D#1, D#3, F#3, A#3 D#min8 | D#3, F#3, A#3, G3 | 1 | 1 | - |
| A2, C3, E3, D5 Amin | D5, C3, E3 | 1 | - | - |
| G#3, A3, A#3, C4, D4 | G#4, A3, D5, C4 | 3 | - | 2 |
| A2, C3, E3, D5, G#5 | G#5, D5, C3 | 2 | - | - |

**Tabella 7: Risultati dei test. Si sono evidenziati i casi che non hanno dato errori.**

Per note mancanti s'intende ovviamente le note che non sono state trovate, mentre gli errori di ottava sono costituiti dalle note identificate di un'ottava più alte del loro *pitch* reale. Per sapere quanti errori sono stati effettivamente fatti, non si devono sommare i valori delle tre celle, ma solo quelli delle prime due. Inoltre, i test che hanno dato come unico errore delle *ghost notes*, o delle note mancanti, sono rivelatori del fatto che si dovrebbe approfondire ulteriormente il problema dell'energia residua.

Le nostre prove, comunque, hanno dimostrato che se si abbassa il limite del 20% tra energia residua ed energia iniziale, si trovano più *ghost notes*, viceversa se si impone maggiore, si incrementa il numero di note non trovate.

Un'analisi quantitativa dei risultati, ci porta ad affermare che ci sia un numero notevole di errori di ottava. Questo tipo di errore, se si vuole, influisce due volte sulla *performance* della trascrizione, in quanto comporta di non trovare una nota sbagliata ad posto di quella realmente presente, e quindi si ha sia un errore di "nota mancante", che una sorta di *ghost note*. Abbiamo pensato a quale sia la causa di questi errori, ed abbiamo, di conseguenza, cambiato il programma *notefind*.

Si notano due cose:

- Gli errori di ottava si verificano sempre "verso l'alto", nel senso che si troverà una nota di un'ottava più alta di quella realmente presente, mai di un'ottava più bassa.

- Gli errori di ottava, non si verificano mai nella parte centrale o alta della tastiera, ma sempre in quella bassa. Tanto per fare un esempio, si troverà un $A_4$ al posto di un $A_3$, o un $G\#_4$ al posto di un $G\#_3$, ma mai un $A_6$ per un $A_5$.

Questo ci porta ad individuare la causa di tali errori principalmente nella distribuzione dell'energia spettrale delle note basse del pianoforte. Come già detto, le parziali fondamentali di queste note, non sono fortemente predominanti, come invece capita per le note più alte, e questo fa in modo che, quando *notefind* associa l'energia alle varie note, ne associ troppa a quelle di un'ottava più alta.

La soluzione che proponiamo al problema è molto semplice, e, sicuramente, migliorabile. Si tratta di sottrarre ad ogni nota, una frazione dell'energia associata a quella posta un'ottava più in basso. Questa frazione, dopo qualche simulazione, è stata posta pari a $^1/_5$ per le note alte ed $^1/_2$ per quelle basse. In Tabella 8 si possono considerare i notevoli miglioramenti introdotti da questa semplice soluzione.

I miglioramenti sono notevoli, basta pensare ad un semplice esempio: se eliminiamo una nota che, in realtà non esiste, noi andiamo ad intaccare l'energia delle altre note, compromettendo la ricerca, e, quindi, incrementando gli errori, che non si limitano più solo a quelli di ottava, ma faranno comparire anche *ghost notes*, faranno diminui-

re l'energia complessiva del *file* e, quindi, fermare la ricerca troppo presto, aumentando anche il numero delle note mancanti.

Con l'introduzione di questo accorgimento abbiamo compiuto un solo errore d'ottava, peraltro in un accordo formato da note molto basse, notoriamente problematiche, con un miglioramento, sotto quest'aspetto, dell'83%!

| **Note presenti** | **Note trovate** | **Errori** | | |
|---|---|---|---|---|
| | | **Note mancanti** | ***Ghost notes*** | **Ottava** |
| A2, A3 ottava | A3+A3 | 1 | 1 | - |
| A4, A5 ottava | A5+A4 | - | - | - |
| C5, C6 ottava | C5, C6 | - | - | - |
| **A3, A#3 semitono** | **A3, A#3** | **-** | **-** | **-** |
| A4, A#4 semitono | A4, A#4 | - | - | - |
| C3, D3 tono | A4, C3, D3 | - | 1 | - |
| A3, C4 terza minore | A3, C4 | - | - | - |
| C3, E3 terza | C3, E3, F3 | - | 1 | - |
| C3, G#3 quinta aumentata | G#4, C3, G#3 | - | 1 | - |
| **A2, C3, E3 Amin** | **C3, E3, A2, F3** | **-** | **1** | **-** |
| **C3, E3, G#3 Cdim** | **G#3, C3, E3, G#4** | **-** | **1** | **-** |
| A2, C3, E3, D5 Amin | D5, C3, E3 | 1 | - | - |
| D#1, D#3, F#3, A#3 D#min8 | D#3, F#3, A#3, G3, A#2 | 1 | 2 | 1 |
| **G#3, A3, A#3, C4, D4** | **G#3, A3, D4, C4, A#3** | **-** | **-** | **-** |
| A2, C3, E3, D5, G#5 | D5, G#5, C3 | 2 | - | - |
| A2, A3, A4, A5 | A3, A5 (A4, A3) | 2 (1) | - | - |

**Tabella 8: Risultati dei test. Si sono evidenziati i test che non hanno dato errori. In grassetto risaltano i casi che hanno portato ad un miglioramento rispetto ai test di Tabella 7.**

Per finire, nell'ultima riga di Tabella 8, è presente il risultato del test per un caso molto particolare, la presenza di quattro note distanti un'ottava l'una dall'altra. Di tali note ne sono state trovate solo due, visto che già dopo aver eliminato queste l'energia residua era scesa ad appena il 15,5% di quella iniziale, ma se si toglie la condizione di "stop" sull'energia residua, si trovano, successivamente, anche $A_2$, ed ancora $A_3$. In questo caso (i risultati sono tra parentesi), visto che si trova due volte la stessa nota, l'unico errore che si verifica è quello di nota mancante ($A_4$). Si noti che dopo aver eliminato la terza nota trovata ($A_3$) l'energia residua è pari al 6% circa.

A questo punto, potendoci considerare abbastanza soddisfatti dei risultati ottenuti con *notefind* e *notekill*, possiamo applicarli al caso tempo variante, in altre parole, all'analisi ed alla trascrizione di un vero brano musicale.

## 6.3 Caso tempo-variante: analisi di un brano musicale

Passare dal caso stazionario a quello tempo-variante, non è banale: oltre al riconoscimento delle note nei *frames*, si pone il problema dell'*onset detection*, e, se tale problema viene risolto in maniera indipendente dal riconoscimento delle note, si deve anche trovare un sistema efficace per "sincronizzare" i risultati dell'*onset detection* e del *pitch detection*.

### 6.3.1 *Onset Detection*

Abbiamo risolto in modo abbastanza indolore il problema dell'*onset detection*. Si deve notare, comunque, che anche questo passaggio genera errori, che si sommeranno a quelli del *pitch detection*, andando, a deteriorare ulteriormente il risultato finale.

Per calcolare gli *onsets*, utilizziamo il seguente algoritmo:

1. Si memorizza il brano da analizzare nel vettore *x*.
2. $x' = \left| \frac{dx}{dt} \right|$
3. Si applica una convoluzione tra una finestra di Hanning di 5 ms ed *x*'. Si ottiene il nuovo segnale *x*".
4. Si applica una nuova trasformazione $y = \frac{d}{dt}[\log_{10}(x''+10)]$.
5. Si divide *y* in tante finestre temporali di ampiezza pari ad *hop* (circa 23 ms), per ogni finestra si sceglie il valore massimo in essa contenuto. In ***os***(*t*) è memorizzato il valore massimo contenuto nella *t-esima* finestra. Questa operazione è equivalente ad una decimazione.
6. Si annullato gli ***os***(*t*) inferiori alla media di tutti i valori di ***os***.
7. A questo punto se ***os***(*t*) ed ***os***(*t+1*) sono entrambi non nulli, si sceglie il maggiore e si annulla l'altro.

L'ultimo punto sembra una forzatura, ma non è così. Noi siamo interessati a che il valore di *hop* sia il più piccolo possibile, in modo che la risoluzione nel tempo sia la migliore possibile, ma *onsets* distanziati di pochi millesimi di secondo, possono considerarsi coincidenti. In Figura 6.4 è visualizzato un esempio. Si tratta di una ripetizione di un'unica note ($D_4$), in questo caso, molto semplice in realtà, non sono stati commessi errori.

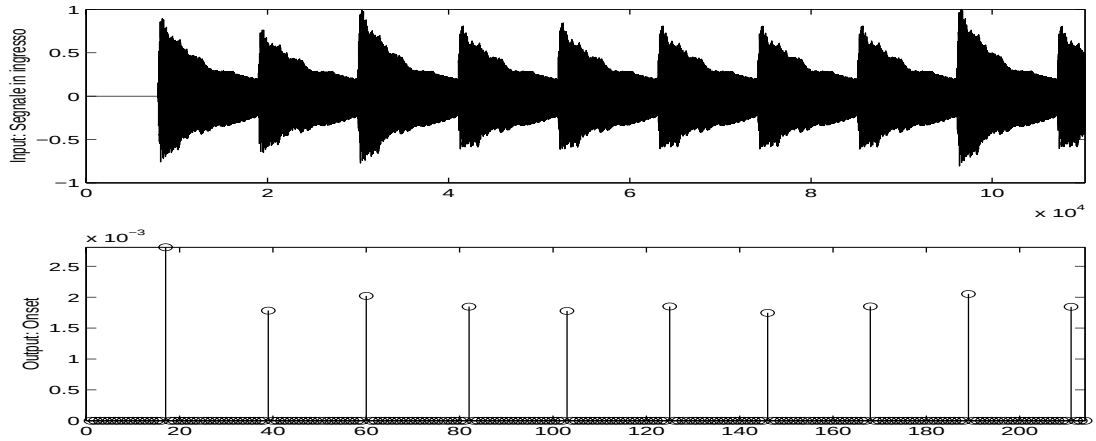


**Figura 6.4: Visualizzazione degli *onsets*. L'esempio considerato è un $D_4$ suonato ripetutamente. Le note sono tutte di durata pari ad $^1/_4$, il *bpm* è pari a 120.**

E' importante porre ancora una volta l'accento sul fatto che nel caso di musica polifonica, qualora si suonassero due note contemporaneamente, l'*onset detection* non consente di determinare a quale delle due note sia imputabile l'*onset*, questo si dovrà fare in una fase successiva, quando si valuteranno assieme sia i risultati dell'*onset detection* che del *pitch detection*.

### 6.3.2 *Pitch Detection* nel caso tempo-variante

Non si discosta molto dal caso stazionario. L'intero brano musicale è diviso in più finestre, di larghezza pari a *wind*, e le note vengono trovate all'interno di tali finestre. Queste finestre temporali non sono solamente contigue bensì si sovrappongono, sicché una frazione della finestra *t-esima* si sovrappone con una frazione della finestra (*t+1*)-*esima*. Ecco quindi che la distanza tra due finestre successive sarà pari ad *hop*, ma la loro durata sarà pari a *wind*. Per *hop* e *wind* si potrebbero scegliere dei valori arbitrari, in realtà si devono scegliere valori di compromesso, in modo da migliorare il più possibile la *performance*.

Per *wind* non si è scelto lo stesso valore considerato per l'analisi nel caso stazionario, ovvero 1024 punti (pari a circa 93 ms), ma il doppio di tale valore, mentre per *hop* si è scelto un valore pari a 512 punti (circa 46 ms). Questo ci garantisce una sufficiente discrezione nel tempo, ed una finestratura sufficiente a dare una buona risoluzione in frequenza. La FFT, inoltre, è fatta su 16384 punti (si passa dalla lunghezza di *wind* a questo valore attraverso un'operazione di *zero padding*), garantendo una risoluzione in frequenza di circa 1,34 Hz.

Ecco come funziona l'algoritmo:

1. Si divide il brano in finestre temporali di durata pari a *wind*. La distanza tra i punti d'inizio di due finestre successive è pari a *hop*. Si consideri la prima finestra.

2. Se l'energia del segnale contenuto nella finestra è inferiore ad una certa soglia, si passa alla finestra successiva. Se anche questa non contenesse energia sufficiente si passerebbe alla successiva[6] ancora e così via sino a che non si trovi una finestra con sufficiente energia.

3. Si applica l'algoritmo *notefind*. Si memorizza la nota trovata, ed una stima dell'energia a lei imputabile.

4. Attraverso *notekill* si elimina la nota trovata.

5. Se il segnale residuo dopo l'eliminazione della nota possiede un'energia superiore ad una certa soglia, si torna al punto 3. Altrimenti si passa al successivo.

6. Se la finestra appena analizzata non era l'ultima, si considera la successiva e si torna la punto 2, altrimenti si è giunti alla fine dell'analisi.

In Figura 6.5 sono visualizzati i risultati del *pitch detection*, la figura è la rappresentazione grafica della matrice ***note***, che fornisce semplicemente l'indicazione della presenza di una certa nota in un certo istante. In ***note***(*tast, t*) è memorizzata l'energia

[6] Significa che nella finestra era contenuto solo silenzio, o per lo meno un segnale che noi consideriamo troppo basso perché rappresenti un suono utile.

che si stima per la nota *tast* nell'istante *t*, se tale valore è nullo, significa che tale nota non è presente nell'istante *t* (ovvero che tale nota non è stata trovata da *notefind*). Si notino i molti errori di ottava, avvenuti particolarmente nella parte bassa della tastiera.

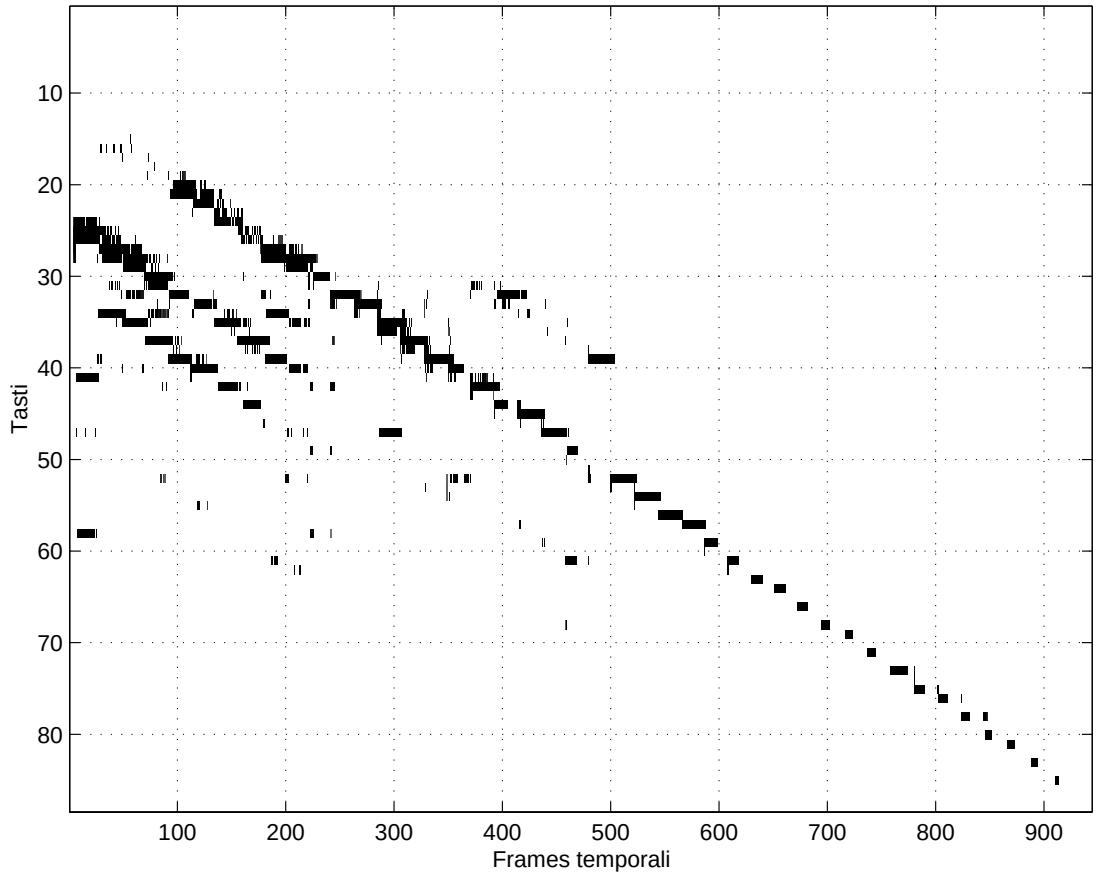


**Figura 6.5: Visualizzazione dei *pitch*. L'esempio considerato è una scala minore armonica di A che copre sei ottave.**

A questo punto si devono mettere assieme i risultati dell'*onset detection* e del *pitch detection*. La cosa è tutt'altro che banale. Gli *onsets* sono rappresentati in un vettore di lunghezza pari al numero delle finestre analizzate. Se all'istante *t* è presente un *onset*, il valore ***onsets***(*t*) è non nullo, tuttavia non possiamo sapere a quale nota sia associabile quell'*onset*!

L'algoritmo che mette assieme i risultati dei due processi consiste nel vedere se in un istante *t* sia stato rilevato un *onset* e, attraverso opportune ipotesi, se questo *onset* sia imputabile alla nota che stiamo considerando.

L'algoritmo che risolve in maniera abbastanza soddisfacente la questione è piuttosto complesso da spiegare, si rimanda agli *script* in appendice 7.2, crediamo sia più faci-

le comprenderlo direttamente da questi. Si faccia presente che un valore non nullo di ***onsets***(*t*) rappresenta un *onset*, mentre tutto il lavoro del *pitch detection* è memorizzato in una matrice ***note*** con 88 righe (una per ogni tasto) e tante colonne quante sono le finestre analizzate. Ora abbiamo un file MIDI che rappresenta la trascrizione del brano.

### 6.3.3 Test e risultati

Siamo giunti al momento topico: dobbiamo testare il nostro trascrittore[7]. Per farlo siamo partiti da brani musicali molto semplici.

Il **primo test** è stato effettuato su una sequenza di $D_4$, suonati tutti alla stessa intensità. Non è stato rilevato alcun errore, né nell'*onset detection*, né nel *pitch detection*.

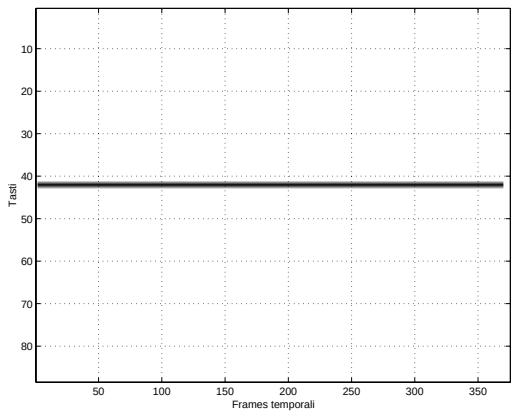


**Figura 6.6: *Pitch detection* del primo test**

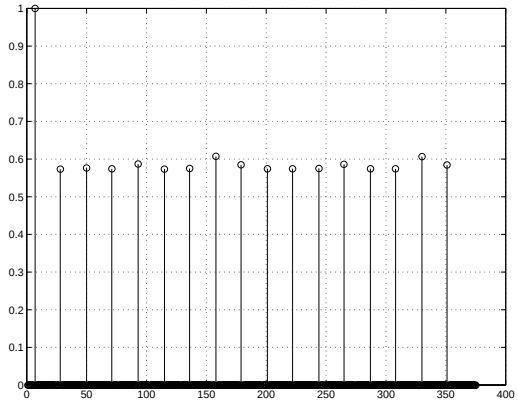

**Figura 6.7: *Onset detection* del primo test**

Il **secondo test** consiste nella trascrizione di una scala, minore armonica, che copre sei ottave. Gli errori nel riconoscimento delle note sono molti, specialmente nella parte bassa della tastiera, e sono per la maggior parte errori di ottava. Nella parte alta della tastiera, invece, si hanno soprattutto errori di *onset detection*, che, però, siamo riusciti a limitare grazie ad un accurato confronto tra i risultati di *onset* e *pitch detection*.

[7] I test sono stati tutti effettuati su campioni di *files* MIDI fatti suonare da un *soundfont* di un pianoforte "*Steinway Model-C* (*7 foot, 5 inch*) *grand piano*" del 1897. Tali *soundfonts* sono di ottima qualità e si possono scaricare da [sito3].

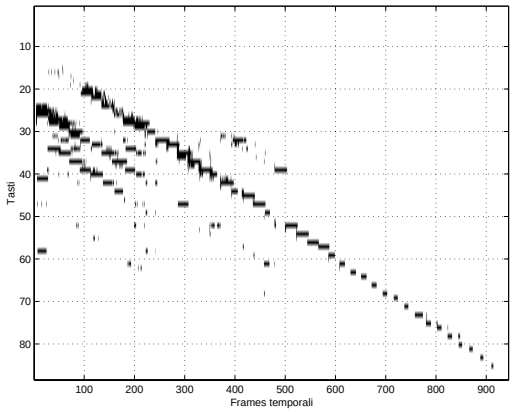


**Figura 6.8: *Pitch detection* nel secondo test**

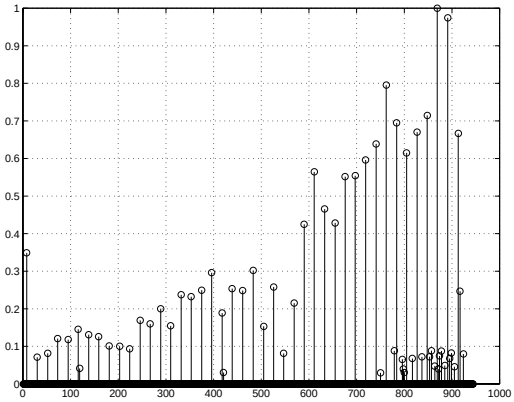

**Figura 6.9: *Onset detection* nel secondo test**

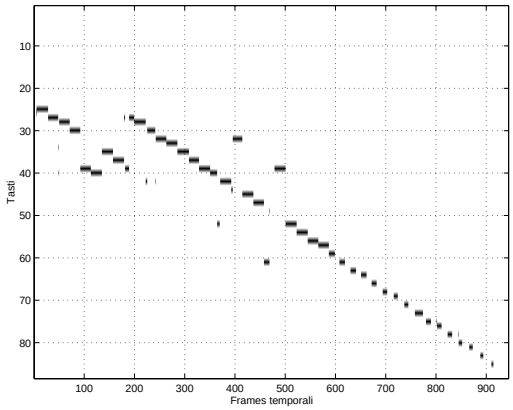


**Figura 6.10: *Pitch detection* nel secondo test. Si forza il programma a trovare una sola nota per *frame*.**

In questo test abbiamo provato a forzare l'algoritmo a trovare una sola nota per ogni *frame*. Come si vede in Figura 6.10 le cose migliorano di parecchio, e questo esempio ci servirà più avanti, per delle conclusioni importanti.

Il **terzo test** consiste in una semplice composizione, con una polifonia massima di tre note, ma con sole due suonate contemporaneamente. In Figura 6.11 e Figura 6.12 si possono vedere separatamente i risultati selle sezioni rispettivamente di *pitch* ed *onset detection*. Il *pitch detection* riporta circa il 96% di note presenti, con circa il 15% di falsi *onsets*. Il *pitch detection*, invece, sembra riportare molti più errori. Tali errori in parte si sommeranno a quelli dell'*onset detection*, degradando la *performance*, in parte saranno "corretti" dal confronto tra *onset* e *pitch detection*.

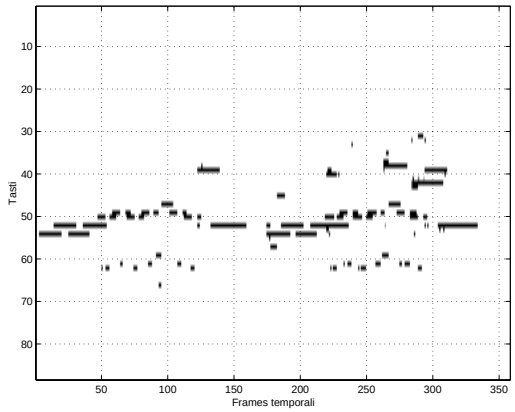


**Figura 6.11: *Pitch detection* nel terzo test.**

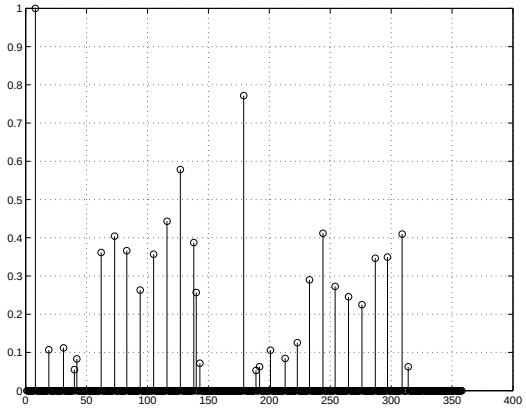

**Figura 6.12: *Onsets* del terzo test.**

Il **quarto test** presenta grossomodo le stesse difficoltà del primo. Viene trovato il 100% degli *onsets*, sempre con circa il 15% di falsi *onsets*.

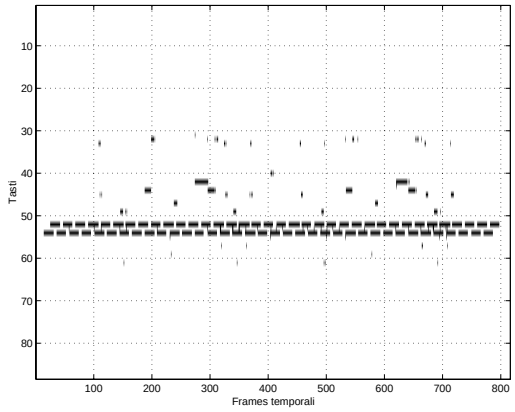


**Figura 6.13: *Pitch detection* nel quarto test.**

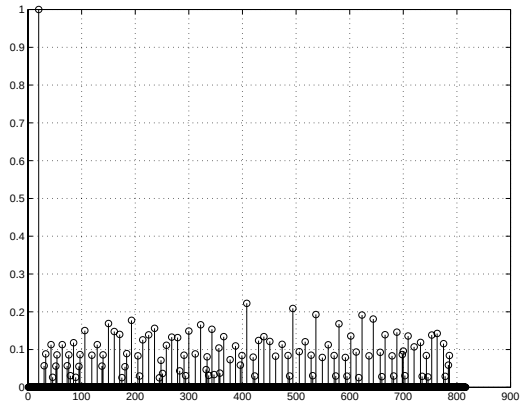

**Figura 6.14: *Onset detection* nel quarto test.**

Il **quinto ed ultimo test** consiste nella trascrizione di un brano musicale tratto da un CD. Si tratta di un frammento di "Days", dall'album "Days" (Caligola Records 1995) di Marcello Tonolo Music Poetry. Non avendo a disposizione una partitura con cui confrontare il file MIDI prodotto, cercheremo di valutare i risultati "a orecchio". Il brano in esame è molto complesso, presentando molti accordi "pieni"[8] nell'accompagnamento, e, quindi, nelle note basse. Per quanto riguarda l'*onset detection*, le prestazioni, sembrano rimanere vicine a quelle riscontrate nei casi precedenti. Potrebbe stupire che, la trascrizione di questo pezzo, sembri "suonare" meglio degli altri brani presi in esame, ma la cosa non è così strana. Questo brano ha una polifonia molto elevata, ed il nostro trascrittore funziona meglio in questi casi

[8] Intendiamo con questo un'alta polifonia, ed una forte presenza di note basse.

(così come abbiamo costatato per AmazingMIDI$^{TM}$), piuttosto che nei casi in cui suonino meno note contemporaneamente[9].

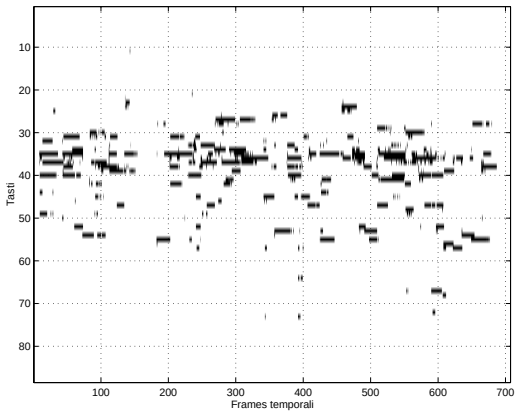


**Figura 6.15:** ***Pitch detection*** **del quinto test.**

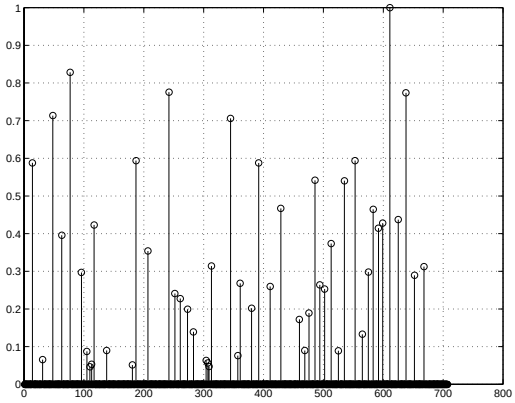

**Figura 6.16:** ***Onset detection*** **del quinto test.**

### 6.3.4 Considerazioni finali

Gli errori presenti nei test sono suddivisibili in tre categorie, vediamole nel dettaglio:

1. **Errori d'ottava**: avvengono soprattutto nella parte bassa della tastiera, sono dovuti a due motivi principalmente: la forte inarmonia del pianoforte nella parte bassa della tastiera, e la poca definizione che la FFT ci permette di avere nelle basse frequenze[10].

2. **Errori di *onset***: Per ovviare a questi si possono percorrere molte strade. Da una parte si può pensare di migliorare la funzione che estrae gli *onsets*, ma questo può non essere sufficiente. Si ricorda, infatti, che tale funzione non fornisce informazioni su quale sia la nota cui l'*onset* estratto è riferito, tale indicazione si deduce solo facendo un confronto tra il vettore ***onsets*** e la matrice ***note***.

3. ***Ghost notes***: l'indicazione di note non realmente presenti (se escludiamo gli errori riconducibili al punto 1) è principalmente dovuta al fatto che l'algoritmo "sbaglia a fermarsi" nella ricerca delle note *frame by frame*. Que-

[9] A meno che non si forzi il trascrittore a lavorare monofonicamente, ovvero trovando una sola nota per ogni *frame*.

[10] Basti pensare che $I_{0,k}$ (cfr. 6.1) per $A_0$ è limitato ad un solo punto.

sto comporta, come abbiamo già avuto modo di notare, sia il fatto di trovare la nota sbagliata, sia di attenuare (attraverso *notekill*) frequenze appartenenti a note realmente presenti e che non sono state ancora rilevate. Un'altra motivazione, ci riconduce ancora alla scarsa risoluzione in bassa frequenza, rimandando alle considerazioni del punto 1. Ma tale risoluzione insufficiente, porta anche ulteriori problemi. Infatti, a causa di questa, le note basse possiedono meno energia, pur avendo un maggior numero di armoniche all'interno della banda limitata dalla frequenza di campionamento (22050 Hz), eliminandole, quindi, si elimina meno energia del "previsto", incontrando problemi di soglia che riconducono al punto 2.

Se prendiamo in come esempio il quinto test, memori delle considerazioni appena fatte, capiamo perché questo suoni meglio, per esempio, del secondo test. Essendo molto alta la polifonia, è molto più facile che l'algoritmo trovi qualche nota in meno, piuttosto che qualcuna in più, evitando, quindi, stonature.

Per concludere, possiamo affermare che i risultati siano soddisfacenti, soprattutto se si considera che l'idea di individuare una nota attraverso una stima della sua energia, è completamente originale. Ulteriori miglioramenti del sistema, dovranno necessariamente passare per:

- **Miglioramento dell'*onset detection***, magari attraverso un algoritmo che ci permetta di imputare con precisione un *onset* rivelato ad una certa nota o ad un gruppo di note.

- **Miglioramento dell'algoritmo di "stop" durante la ricerca nei *frames***. Questo necessariamente passa attraverso uno studio empirico che ci permetta di valutare meglio i problemi di soglia che inevitabilmente incontreremo.

- **Miglioramento della definizione in bassa frequenza**, che porterebbe vantaggi non solo al riconoscimento delle note nella parte bassa della tastiera, ma anche all'analisi delle energie del *frame*, che è così importante per i miglioramenti auspicati nel punto precedente.

# 7 Appendici

## 7.1 Modello del suono di un pianoforte

[Askenfelt90], [Bank00], [Fletcher-Rossing98], danno diversi modelli di quale sia la rappresentazione del modulo della frequenza di una nota di pianoforte, e rimandiamo a questi scritti per una trattazione teorica approfondita. Ovviamente, sia per *notefind* sia per *notekill* è molto importante sapere come si distribuisce l'energia nello spettro, in altre parole come questa si addensa attorno alle parziali.

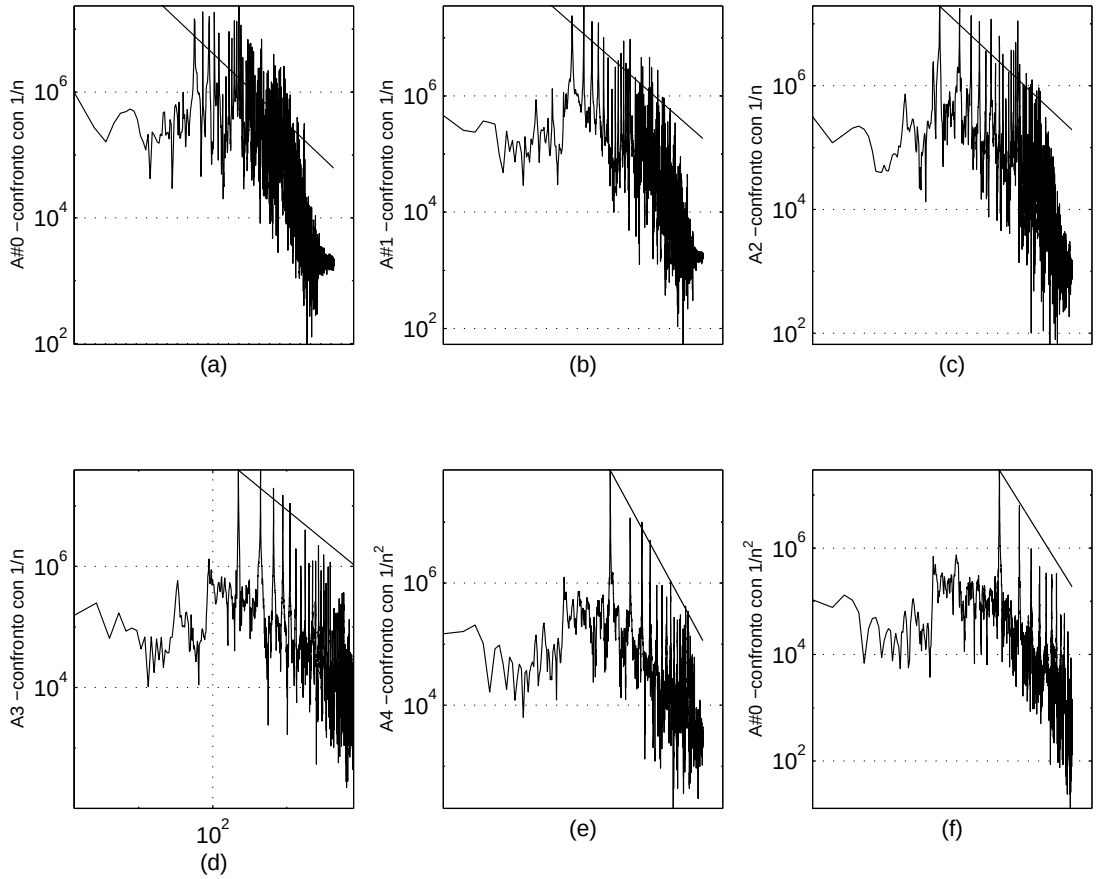


**Figura 7.1: Andamento delle parziali delle note in vari punti della tastiera.**

Abbiamo rilevato che, nel caso sia presente un'unica nota, considerando un intorno della parziale $f_j$ di raggio pari a $f_k \bullet (2^{1/24} - 1)$, nell'unione di tali intorni si concentra circa il 95% dell'energia totale.

Inoltre, l'ampiezza delle parziali $f_j$ cala all'aumentare di $j$. Il comportamento si può osservare nella Figura 7.1. Come si vede, si può dividere la tastiera in tre parti:

- Nella prima parte (Figura 7.1a,b), che comprende approssimativamente le prime due ottave, lo spettro delle note è carente delle prime parziali, e risente molto di quanto riportato nel capitolo 4.2, sulla relazione tra martelletti e corde, inoltre l'ampiezza $A(f_j)$ delle parziali, pur calando con l'aumentare di $j$, non segue una relazione abbastanza regolare per essere utile ai nostri scopi.

- Nella seconda parte (Figura 7.1c,d), che corrisponde all'incirca alle successive due o tre ottave, l'ampiezza $A(f_j)$ decresce abbastanza regolarmente come $1/j$.

- Nella terza parte (Figura 7.1e,f), il resto della tastiera, l'ampiezza $A(f_j)$ decresce come $1/j^2$.

I nostri algoritmi partono da queste considerazioni, ma si evolvono attraverso successive prove pratiche.

## 7.2 *Scripts* per MatLab™

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Analisi energie
% Procedimento iterativo
% Calcola l'energia relativa agli intorni delle armoniche e l’energia totale del file.
% Fornisce il rapporto tra le due energie
% Luca Tiengo: lucatiengo@libero.it http://digilander.iol.it/lucatiengo
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

close all;
clear all;

warning ('--------------Start-------------------')

[s,Fs,bit]=wavread ('C:\..\nomefile.wav');
s=s/max(s);
range=21:21+4096; %delimitazione di una finestra di analisi
sanalisi=s(range);

T=1/Fs;
N=4096•4;


S=fftshift(fft(sanalisi,N)/(N/2));
f=(-N/2:N/2-1)/(N•T);
file = abs(S(length(S)/2+1:length(S)));
NRGTOT = sum(file(1:length(file)).^2)

NRG=0;
pitch=440;
nota=pitch;
for (n = 1:100)
  nota1 = nota/2^(1/24);
          nota2 = nota•2^(1/24);
  delta=round((((nota2)•T•(N+1)) - round((nota1)•T•(N+1)))/2);
  center = round((nota)•T•(N+1));
  ind1 = center - delta;
  ind2 = center + delta;
  if (ind2 < length(file))
    NRG = NRG + sum(file(ind1:ind2).^2);
    nota = (n+1)•pitch;
  end
end

NRG
NRGTOT
NRGRate = NRG/NRGTOT
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Transcriber
% Luca Tiengo: lucatiengo@libero.it http://digilander.iol.it/lucatiengo
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

warning ('--------------Start-------------------')

[sload,Fs,bit]=wavread('C:\Documenti\PUPPET\Tesi\samples\tempovariante\steinway\prova3.wav');
len_song = 22050*1;
s=sload;%(1:len_song+1);
s=s/max(s);
wind=round(4096/2);
hop=4096/8;
 len_rate=round(length(s)/hop);
nnote=5;
note=zeros(88,len_rate);

T=1/Fs;
N=4096*4.

onsets=onset1(s,Fs,hop);

% Adesso devo cominciare a fare l'analisi

for(w=1:len_rate-6)
% Qui cominciamo a scorrere tutto il file
  perc = w/len_rate*100                              % Percenuale di lavoro compiuto
  sanalisi=s(1+(w-1)*hop:(w-1)*hop+wind);            % Delimitazione di una finestra di analisi

  S=fftshift(fft(sanalisi,N)/(N/2));
  file = abs(S(length(S)/2+1:length(S)));
  integro=file;
  f=(-N/2:N/2-1)/(N*T);
  f1=(0:length(file)-1)/(length(file)*2*T);
 n = 0;
  Ei = sum(file.^2);                                 % Energia iniziale
  Eres = Ei;                                         % Energia residua
  Esum = 0;                                          % Energia eliminata in totale per ogni frame

  while((Eres>10e-4)*(Esum/Ei<1)*(n<nnote)*(Eres/Ei>0.1))
    [notanota, tastonoto, Enrg] = notefind (file, Fs, N);
    note(tastonoto, w) = energia (notanota, integro, Fs, N);
    file = notekill (file, Fs, N, notanota);
    Eres = sum(file.^2);
    Esum = Esum + Enrg;
    n=n+1;
  end
end


% Ora ho a disposizione la matrice "note" che ha nelle righe gli eventi-tempo, nelle colonne i tasti.
% Praticamente la casella note(i,j) ci dice se il tasto j sta suonando nell'istante i.
% Da questa dobbiamo ricavarci una matrice, il cui numero di righe non è fissato, ma le cui colonne
% hanno questi valori:

% 1^ Colonna     2^ Colonna      3^ Colonna        4^ Colonna      5^ Colonna
% Onset time     Durata nota     Pitch (C4=60)     Velocity        Midi Channel

% Definiamo alcune variabili:
% Offset Time = Onset time + Durata Nota
```

```
% Il pitch si calcola considerando che al valore 60 corrisponde C4 -> Pitch = n° tasto +20

[row, columns] = size (note);

D = zeros(5, 5);

t=0;       % Rappresenta la durata della nota.
           %Quando si trova la nota per la prima volta, viene messo pari a 1,
           % e si incrementa se la nota è presente anche nei frames successivi.
d=1;       % Tiene il conto degli eventi già accaduti. Per eventi intendiamo l'inizio di una nota.


for (tast=1:88)
  for (temp=2:columns-2)
    if (onsets(temp+2))
      if (note(tast,temp)>1*note(tast,temp-1))
        if(t)
          D(d,1)=(temp-t)*hop/Fs;   % Onset time
          D(d,2)=t*hop/Fs;                                  % Durata nota
          D(d,3)=tast+20;                                   % Pitch
          %           D(d,4)=RES(tast,temp-t);              % Velocity
          t=1;
          d=d+1;
        else
          t=1;
        end
      else if (note(tast,temp))
          t=t+1;
        else if (t)
            D(d,1)=(temp-t)*hop/Fs;                         % Onset time
            D(d,2)=t*hop/Fs;                                % Durata nota
            D(d,3)=tast+20;                                 % Pitch
            %           D(d,4)=RES(tast,temp);              % Velocity
            t=0;
            d=d+1;
          end
        end
      end
    else if (note(tast,temp))
        t=t+1*(t>0);
      else if (t)
          D(d,1)=(temp-t)*hop/Fs;   % Onset time
          D(d,2)=t*hop/Fs;                                  % Durata nota
          D(d,3)=tast+20;                                   % Pitch
          %           D(d,4)=RES(tast,temp);                % Velocity
          t=0;
          d=d+1;
        end
      end
    end
  end
end

D(d,1)=(len_rate)*hop/Fs;                                   % Onset time
D(d,2)=hop/Fs;                                              % Durata nota
D(d,3)=0;                                                   % Pitch
          %           D(d,4)=RES(tast,temp);                % Velocity
```

```
D(:,4)=125;
nt = notetrack(D);

save (nt, 'C:\..\nomefile.mid',120);

warning ('----------The End----------')

figure
stem (onsets)
grid

figure
map = (0:63)'/63;
colormap([1-map 1-map 1-map]);
image(min(note/.0004*length(colormap),64))
ylabel('Tasti')
xlabel('Frames temporali')
grid

sound(s,22050)
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% energia
% Stima l’energia di una certa nota
% Luca Tiengo: lucatiengo@libero.it http://digilander.iol.it/lucatiengo
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function Energia = energia (pitch, file, Fs, N)

ind = round(12*log2(pitch/27.5)+1);
T=1/Fs;
Energia=0;
E=0;

%per le prime due ottave cerchiamo di trovare un algoritmo che enfatizzi di più le armoniche basse

if (ind<31)
  nota=pitch;
  for (n = [1:100])
    nota = (n)*pitch;
    nota1 = nota/2^(1/24);
    nota2 = nota*2^(1/24);
    ind1 = round((nota1)*T*(N+1));
    ind2 = round((nota2)*T*(N+1));
    if (ind2< length(file))
      Energia = Energia + sum((file(ind1:ind2).^2)/n);
    end
  end

elseif (ind<40)
  nota=pitch;
  E=0;
  for (n = [1:100])
    nota = (n)*pitch;
    nota1 = nota/2^(1/24);
    nota2 = nota*2^(1/24);
    delta=round((((nota2)*T*(N+1)) - round((nota1)*T*(N+1)))/2);
    center = round((nota)*T*(N+1));
    ind1 = center - delta;
    ind2 = center + delta;
    if (ind2< length(file))
      Energia = Energia + sum((file(ind1:ind2).^2/(n^0.5)));
    end
  end

else
  nota=pitch;
  for (n = [1:100])
    nota = (n)*pitch;
    nota1 = nota/2^(1/24);
    nota2 = nota*2^(1/24);
    delta=round((((nota2)*T*(N+1)) - round((nota1)*T*(N+1)))/2);
    center = round((nota)*T*(N+1));
    ind1 = center - delta;
    ind2 = center + delta;
    if (ind2< length(file))
      E = E + sum (file(ind1:ind2).^2);
      if (n==1)
        Energia = E;
        main = E;
      else
```

```
          Energia = min(E, main/n^3) + Energia;
        end
      end
    end
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% onsets1
% Trova gli onsets delle note
% Luca Tiengo: lucatiengo@libero.it http://digilander.iol.it/lucatiengo
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function out1=onset1(x,Fs,hop)

x=x/max(x);
len = length(x);

T=1/Fs;
len1= round(len/hop);
h=hanning(0.05/T);

y = diff(log10(conv(h, abs(diff(x)))+10));

for(j=1:len1)
  os(j) = max(y(1+(j-1)*hop:1+j*hop));
end

os = os/max(os);

ave=(sum(os))/length(os);
os(:)=os(:).*(os(:)>ave);
out1=os;


for(j=2:length(os)-1)
  out1(j)=(os(j)>=os(j-1))*(os(j)>os(j+1))*os(j);
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Funzione notefind
% trova la nota più presente in un file
% input: file, Fs (frequenza di campionamento di file), N (numero di punti nella finestra di analisi
% output: notanota (il pitch della nota trovata), tastonoto (il relativo tasto sul pianoforte)
% Luca Tiengo: lucatiengo@libero.it http://digilander.iol.it/lucatiengo
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [notanota, tastonoto, Enrg] = notefind (file, Fs, N)

T=1/Fs;
pitch= 27.5;        %Questa è la nota più bassa che cercheremo, corrispondente ad un A0.
NRG = zeros (2,88);
for(ind =1:88)

  %per le prime due ottave cerchiamo di trovare un algoritmo che enfatizzi di più le armoniche basse
  if (ind<31)
    nota=pitch;
    NRG(1,ind)=nota;
    for (n = [1 2 3 5 7 11 13 17 19 23 29 31 37])
      nota = (n)*pitch;   % Attenzione! Non devo trovare la nota successiva, ma l'armonica successiva!!!
      nota1 = nota/2^(1/24);
      nota2 = nota*2^(1/24);
      ind1 = round((nota1)*T*(N+1));
      ind2 = round((nota2)*T*(N+1));
      if (ind2< length(file))
        NRG(2,ind) = NRG(2,ind) + sum((file(ind1:ind2).^2)/n);          % Bisogna metterci bene le mani su questa funzione...
      end
    end
    if (ind>12)
      NRG(2,ind)=NRG(2,ind) - NRG(2,ind-12)/5;
    end
    pitch = pitch*2^(1/12);

  elseif (ind<40)
    nota=pitch;
    E=0;
    NRG(1,ind)=nota;
    for (n = [1 2 3 5 7 11 13 17 19 23 29 31 37])
      nota = (n)*pitch;   % Attenzione! Non devo trovare la nota successiva, ma l'armonica successiva!!!
      nota1 = nota/2^(1/24);
      nota2 = nota*2^(1/24);
      delta=round((((nota2)*T*(N+1)) - round((nota1)*T*(N+1)))/2);
      center = round((nota)*T*(N+1));
      ind1 = center - delta;
      ind2 = center + delta;
      if (ind2< length(file))
        NRG(2,ind) = NRG(2,ind) + sum((file(ind1:ind2).^2/(n^0.5)));
      end
    end
    NRG(2,ind)=NRG(2,ind) - NRG(2,ind-12)/5; % qui non è necessario fare l'if
    pitch = pitch*2^(1/12);
  else
    nota=pitch;
    E=0;
    NRG(1,ind)=nota;
    for (n = [1 2 3 5 7 11 13 17 19 23 29 31 37])
```

```
        nota = (n)*pitch;
        nota1 = nota/2^(1/24);
        nota2 = nota*2^(1/24);
        delta=round((((nota2)*T*(N+1)) - round((nota1)*T*(N+1)))/2);
        center = round((nota)*T*(N+1));
        ind1 = center - delta;
        ind2 = center + delta;
        if (ind2< length(file))
          E = sum (file(ind1:ind2).^2);
          if (n==1)
            NRG(2,ind) = E;
            main = E;
          else
            NRG(2,ind) = min(E, main/n^2) + NRG(2,ind);
          end
        end
      end
      NRG(2,ind)=NRG(2,ind) - NRG(2,ind-12)/2; % qui non è necessario fare l'if
      pitch = pitch*2^(1/12);
    end
end

% Qui bisogna trovare il massimo

notanota = NRG(1, find(NRG(2,:)==max(NRG(2,:))));
tastonoto=round(12*log2(notanota/27.5)+1);
Enrg= max(NRG(2,:));
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Funzione notekill
% trova una nota dal file
% input: file, Fs (frequenza di campionamento di file),
% N (numero di punti nella finestra di analisi notanota (la nota da eliminare)
% output: killedfile (il file da cui è stata eliminata la nota.
% Non è distruttivo, non influenza l'input che gli viene passato
% Luca Tiengo: lucatiengo@libero.it http://digilander.iol.it/lucatiengo
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [killedfile] = notekill (file, Fs, N, notanota)

T=1/Fs;
killedfile=file;

for (n = 1:100)
  nota = n*notanota;
  nota1 = nota/2^(1/24);
  nota2 = nota*2^(1/24);
  ind1 = round((nota1)*T*(N+1));
  ind2 = round((nota2)*T*(N+1));
  if (ind2< length(killedfile))
    killedfile(ind1:ind2) = (killedfile(ind1:ind2))*(1/20*(n==1)+(1-(1/(n+1)^0.5))*(n>1));
  end
end
```

# 8 Bibliografia

## 8.1 Libri ed Articoli


**[Askenfelt90]** Askenfelt (a cura di), Five lectures on the acoustics of the piano Stockholm, Royal Swedish Academy of music, 1990.

**[Bank00]** Bank, Physics-based Sound Synthesis of the Piano, Helsinky, University of Technology (Laboratory of Acoustic and Audio Signal Processing, Espoo2000, Report 54, pp.17-32

**[Bregman90]** Bregman, Auditory Scene Analysis: the Perceptual Organization of Sound, Cambridge, MIT Press, 1990.

**[Fausett94]** Fausett L.V., Fundamentals of Neural Networks: Architectures, algorithms, and applications, Englewood Cliffs, 1994, Prentice Hall International Editions.

**[Fletcher-Rossing98]** Fletcher, Rossing, The Physics of Musical Instruments, Springer-Verlag, New York, 1998, pp. 345-344.

**[Klapuri98]** Klapuri A.P., Automatic Transcription of Music, Master of Science Thesis, Tampere, Department of Information Technology, Tampere University, 1998.

**[Klapuri01]** Klapuri, A.P., "Multipitch estimation and sound separation by the spectral smoothness principle" IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2001.

**[Klapuri-Virtanen-Holm00]** Klapuri A.P., Virtanen T.O., Holm J.M., "Robust multipitch estimation for the analysis and manipulation of polyphonic musical signals" Proceedings COST-G6, Conference on Digital Audio Effects, Verona, Italy, 2000.

**[Lyon-Slaney90]** Lyon R.F., Slaney M. "A Perceptual Pitch Detector", Proceedings of the 1990 International Conference on Acoustics Speech, pp. 357-360, vol.1.

**[Marolt99]** Marolt M., "A comparison of feed forward neural network architectures for piano music transcription" Proceedings of the 1999 International Computer Music Conference, Bijng, China.

**[Marolt00]** Marolt M., "Adaptive oscillator networks for partial tracking and piano music transcription", Proceedings of the 2000 International Computer Music Conference, Berlin, Germany.

**[Martin96]** Martin, "A Blackboard System for Automatic Transcription of Simple Polyphonic Music". MIT (1996) Media Laboratory Perceptual Computing Section Technical Report n°385.

**[Moorer75]** Moorer, On the Segmentation and Analysis of Continuous Musical Sound by Digital Computer, Ph.D.Thesis, Dept. of Music, Stansford University, 1975.

**[Slaney88]** Slaney, M., Lyon's Cochlear Model, Advanced Technology Group, Apple Technical Report #13, 1988.

**[Slaney98]** Slaney, M., Auditory Toolbox: A MATLAB Toolbox for Auditory Modeling Work, Version 2, Technical Report #1998-010 Interval Research Corporation, 1998.

# 8.2 Siti Internet

## 8.2.1 Siti correlati alla Trascrizione Automatica della Musica

**[MaroltWeb]** http://lgm.fri.uni-lj.si/~matic/research.html sito di Matija Marolt: trascrizione automatica della musica affrontata con le reti neurali.

**[SlaneyWeb]** http://www.slaney.org/malcolm/pubs.html sito di Malcolm Slaney: da questo sito si possono scaricare *l'Auditory Toolbox* ed il relativo manuale.

**[AmazingMIDI™Web]** http://www.pluto.dti.ne.jp/~araki/amazingmidi/ *Software* commerciale. "*…AmazingMIDI™ automatically transcribes music, converting WAV* files *into MIDI* files*. It can recognize single-instrument polyphonic music. It's a powerful tool to help you to transcribe music, to practice musical instruments and to make MIDI* files*, and so on…"* (sic)

**[KlapuriWeb]** http://www.cs.tut.fi/~klap/ sito di Annsi Klapuri.

## 8.2.2 Altri Siti

**[sito1]** http://www.mbfys.kun.nl/~cemgil/software.html per scaricare un *toolkit* per Matlab utile a lavorare con e sui files MIDI.

**[sito2]** http://www.piano-midi.de/midicoll.htm per scaticare files MIDI di musica per solo pianoforte.

**[sito3]** http://www.wstco.com/pianosounds/freesoundfont.htm per scaticare i *soundfonts* di un pianoforte "*Steinway Model-C* (*7foot,5inch*) *grand piano*" del 1897. I *soundfonts* in questione sono di ottima qualità (44100Hz, 16 bits, un file sf2 di 25MB complessivi).

**[sito4]** http://elj.warwick.ac.uk: sito di Dan Hunter: reti neurali applicate alla pratica giudiziaria.

**[sito5]** http://www-dsp.rice.edu/splib/: *Signal processing library***.**

**[sito6]** http://www-isis.enst.fr/Applications/tftb/iutsn.univ-nantes.fr/auger/tftb.html *Time-frequency toolbox* per Matlab.

**[sito7]** http://digilander.iol.it/lucatiengo: il sito del sottoscritto, da cui potrete scaricare la tesi in formato PDF e tutti gli *script* per MatLab.

**[sito8]** http://www.mtlc.net *"The Music Technology Learning Center Providing hobbyists, educators, and students with up-to- date information on the world of computers and music."* (sic)

**[sito9]** http://www.rev.net/~aloe/music/ interessante per rinfrescare la memoria sulla teoria musicale.

**[sito10]** http://www.musicarrangers.com/star-theory/ come sopra.

**[sito11]** http://www.stolaf.edu/people/hamlin/acoustics%20short.htm?tm come sopra.

**[Sito12]** http://csunix1.lvc.edu/~snyder/em/varese.html: Edgard Varése e musica elettronica contemporanea

# Titoli di Coda

**“Scommetti solo quando puoi perdere (…)**
**l’importante è pagare l’affitto prima.”**
**Charles Bukowsky *“Storie di Ordinaria Follia”***

Ecco giunto il momento più impegnativo di tutta la stesura della tesi! Come fare a ricordare tutti? Da bravi ‘*gnegnèri* seguiremo un rigoroso ordine logico.

Ovviamente grazie al Prof. De Poli, ed all’Ing. Drioli per l’aiuto e l’assistenza. Grazie anche all’Ing. Borin, che non ha mai mancato di ricordarmi, con occhiate maliziose e sghignazzate benevole, quanto gravoso fosse l’impegno intrapreso affrontando questa tesi di laurea.

## Cose serie

Grazie a Dio, se c’è, e se non c’è lo ringrazierò quando torna.

Grazie alla Vita, l’unica cosa da cui non si può uscire vivi.

Un ringraziamento particolare ai miei genitori, che mi hanno mantenuto, supportato e sopportato in questi anni. Vi ricompenserò togliendo il disturbo il prima possibile! ;-)

Un ricordo per Ivan ed Enzo, avete lasciato un segno indelebile.

Un bacio a nonna Tella, nonna Giolle e nonno Ettore. Dateci un’occhiata che qua va tutto a cartoni!

Un abbraccio forte a nonno Gigi, voglio ancora sentirti ridere, e raccontare la storia dello *“spiciaffete”*, del fantasma sull’albero e del bisnonno che andava a liberare i tuoi fratelli (ubriachi stonfi) dalla galera.

Un saluto affettuoso al resto del parentame, tutto quello che sentirete durante il dopo-laurea è falso, fazioso e calunnioso. Fate conto che non sia vero, anche nel vostro bene…

Grazie alle commesse di “Tommasini Calzature” che mi hanno pazientemente aiutato a scegliere il mio primo paio si scarpe “serio”.

## Cose mie

Grazie Francesca, per quello che sei, che mi hai dato, che mi dai e mi darai. Ti amo anche se odi il mio umorismo, e sei più gelosa di un siciliano in gita con la moglie in un campo nudisti (“*In joy and in sorrow my home is in your arms”*).

Impossibile non ricordare i Ledel: Marko (i cavei te te i si tajai… manca ea fameja…), Maurizio “Predator” (tu vo’ fa’ l’ammmmericano, mmmericano, mmmericano… ma se mejo che te i bevi!), Ox (putei picuiiiiiii), Teo (quando metareto ea testa a posto?) e Stefano (in un certo senso gavemo ciavà insieme, no?). Potrò dimenticarmi qualche nota ma non dimenticherò mai nemmeno un singolo giorno passato a suonare con voi. Un salutone anche a tutti i fans dei Ledel, Gigio e la Mara, l’Animale e la Roberta, Giacomo San, il Babi, Lino e Coorsal Dischi, Testo, e tutti gli altri che non posso ricordare ora.

Addio Pocioci, sei stato un buon gatto. Ci mancherai.

Una sambuca con ghiaccio e mosca al mio caro fratellino Matteo (moea e femene che e te fa mae, sta tento ai gati che te traversa ea strada, bevi poco e torna casa presto, e se te vè in leto par ultimo, ricordate de metare l’alarme) e tutti i suoi amici.

VaLENtina, se hai bisogno di un amico telefonami. Se hai bisogno di un uomo, gridalo forte dalla finestra.

Un ringraziamento sparso per tutti quelli che hanno a che fare con Puppet: io, Massi, Maurizio, Cece, Silvan, Seren(a)rea, Nicola Loverre, Scarlet, Herman Medrano, Dimo.

Un ringraziamento di tutto cuore a Donpa, Lorenzo, Sandro ('zzo! avrei dovuto pagare per lavorare con te!), Simone, la Fabiola, la Lisa ed a tutti i bislacchi della Usiogope (cavéghe el vin!).

Un grazie enorme e parecchio alcolico a Lele (#1, uno dei miei Maestri di vita).

Vi ringrazio per aver lavorato con me: Checca, Vale, Matteo e tutti quelli del Rococò, L'Altro Fragile, Elisa, Juri (el Director del bar) sua moglie e le loro splendide bimbe. Voglio ricordare anche tutta la gentaglia del Pedrocchi, con loro mi sono anche divertito. Un salutone a Nello (il VERO sindaco di Padova), Samuele Franchi, la sua dolce mogliettina Elena ed il bimbo che sta per arrivare (e che se nasce maschio si chiamerà come me!).

Ciao a tutti i tipacci da Feltre e dintorni: Emanuele e Sandra, Claudio e Mara, Enrico, Elena e Vania.

In ordine sparso: Maria Giovanna, i funghetti, Avana Club 7yrs, Panda 750cl, Bjork (ah, se io non avessi già la morosa…), Tricky, John Coltrane, la mia meravigliosa Jackson, il mio Team di Kick Boxing, in particolare il Maestro Radames, Andrea Fabbro (ricordi l'*American Party* ad Ios?), Simone Gatto, Andrea Marcato, Mauro Pattarello (Marulo), Stefano Angi, Matteo Poletto (R.I.P.), Giulio Fattoretto (hai lavato le tende?), e tutti i rompi coglioni del liceo.

Il ricordo vola dolorosamente anche ai compagni di sventura dell'Obitorio di Coscienze. Ne siamo usciti sani, ed abbiamo imparato una grossa lezione…

## Jap

Grazie a Franco, a Maida ed alla Susy del Bar Paradise (se non ci foste stati voi mi sarei laureato prima), Baga (sei un grande, spero di non perdere i contatti), Angelo (eccheçça§§o! Adoro dormire a cavalcioni sul poggiolo della tua finestra!) e Lea, Teo "Canareggio" (te vojo ben, can de un venexian!), Marco Sartor, Gaia, Alessia, Eros, Biofa, Pippo (moto del cazzo…), Michele (BI!), Ghego e Roana (che per fortuna ha trovato un manico ed ora è più trattabile…), Andrea Biondo e Angelica, Fede + Speranza + Carità, Silvia (1, 2, 3, unica etc…), il Catta e sua morosa Eva, Enrico, Manuelito, Fabietto, Kia, e tutti quelli che mi sono dimenticato, che tanto è lo stesso perché non leggeranno mai questa tesi.

## F#§k Off

Generalmente non odio nessuno, ma mi sento di mandare a quel paese diversa gente, almeno una volta nella mia vita: Alessia (sono un elefante…), Manuel (se te ciapo…), le donne dei miei primi vent'anni (che non mi cagavano mai), la stragrande maggioranza della musica italiana moderna, gli spritz che fanno a Padova (che costano un casino e fanno schifo, ma ti mandano in ciucca subito!), la TV che fanno di pomeriggio, i cartoni animati moderni (escluso Dragon Ball), il Cinema d'Autore Europeo (quasi tutto), chi pensa che gli ingegneri siano tutti sfigati (perché ha ragione…), Fisica Tecnica (esame di M39D@!), Mc Donald's ed i fast food in generale (MANGIARE IN PIEDI E' UNA BESTEMMIA!).

## Colonna Sonora

Ritengo doveroso elencare dischi ed artisti che mi hanno tenuto compagnia durante la stesura della tesi. Noi siamo quello che ascoltiamo: Alice In Chains, Aphex Twin, Archie Shepp, Area, Asian Dub Foundation, Beastie Boys, Beck, Bill Laswell, Bjork, Blur, Bob Marley, Cannonball Adderly, Carcass, Casino Royale, Charles Mingus, Charlie Parker, Chemical Brothers, Chet Baker, David Holmes, Deftones, Depeche Mode, Emiliana Torrini, Entombed, EyeHateGod, Fabrizio De Andrè, Faith No More, Faithless, Fantomas, Fear Factory, Franco Battiato, Frank Zappa, God Machine, Goldie, Gong, Gorillaz, Hardware, Henry Threadgill, House Of Pain, Jaco Pastorius, Jamiroquai, Jeff Buckley, Joe Zawinul, John Coltrane, John Zorn, Korn, Kyuss, Bernocchi, Kondo, Bullen, Ledel, Leftfield, Liquid Jesus, Living Colour, Man or Astroman?, Manu Chao, Masada, Massive Attack, Mad Professor,

Master Musician of Jajouka, Material, Max Gazzè, Megadeth, Metallica, Miles Davis, Ministry, Moby, Mogway, Moloko, Morcheeba, Mordred, Mr Bungle, Naked City, Napalm Death, Nearly God, Nick Drake, Nick Scopelitis, Nine Inch Nails, Nusrat Fateh Ali Khan, Ornette Coleman, Ozric Tentacles, Painkiller, Pat Metheny, Pestilence, Pink Floyd, Placebo, Portishead, Praxis, Primus, Prodigy, Propellerheads, Puppet, Rabih Abou-Khalil, Radiohead, Red Hot Chili Peppers, Red Snapper, Refused, Roni Size, Royalize, Sacred System, Scorn, Shoji Yamashiro, Shout, Slipknot, Smashing Pumpkins, Sonic Youth, Soundgarden, Spoladore, Squarepusher, St Germain, Stan Getz, Steve Vai, Subsonica, Talvin Singh, The Drummers Of Burundi, Tortoise, Transglobal Underground, Trent Reznor, Tricky, Trilok Gurtu, U2, Uncle, UNO, Voivod, Weather Report, Wu-Tang Clan, Xenia, Yoko Ono.

Chiudo questi scritti con un'ultima perla di saggezza:

**"In pignatea picoea, poca papa ghè"**

una massima di Nonna Noemi. Come dire: chi se ne frega se nella botte piccola c'è il vino buono se è troppo poco per far festa?

Grazie per l'attenzione.

**Luca "Puppet" Tiengo**

lucatiengo@libero.it
http://digilander.iol.it/lucatiengo
http://digilander.iol.it/puppetweb

**… s w i t c h o f f …**