View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by Electronic Thesis and Dissertation Archive - Università di Pisa

Università degli Studi di Pisa

Dipartimento di Ingegneria dell'Informazione
Corso di Laurea Magistrale in Ingegneria Informatica

Riconoscimento automatico di accordi e note eseguiti da strumenti musicali

Relatore:
Prof. Giorgio Buttazzo

Candidato:
Pietro Fara

Anno Accademico 2014/2015

Indice

Ringraziamenti

Un doveroso ringraziamento va al Prof. Giorgio Buttazzo per la passione che mi ha trasmesso per le materie di cui si occupa, per le numerose ore dedicate alla mia tesi e per la pazienza dimostrata. Voglio inoltre ringraziare il Dott. Ge Wang e il Dott. Matthias Mauch perché, anche se non ci siamo mai incontrati, senza il loro duro lavoro di ricerca svolto e la loro voglia di condivederlo non sarebbe stato possibile svolgere questa tesi. Inoltre, ringrazio i miei genitori e i miei fratelli per il sostegno dato anche nei momenti più difficili che mi ha permesso di non perdere mai di vista l'obiettivo e la fiducia in me stesso durante la mia esperienza universitaria. Un ringraziamento particolare va a Costanza che mi ha aiutato nel difficilissimo compito di descrivere il mio lavoro. Ringrazio infine i miei invincibili compagni di viaggio Alessio, Paolo e Indri, con cui ho condiviso gioie e sofferenze, sempre con in testa l'obiettivo di ottenere il miglior risultato possibile dal nostro lavoro.

Capitolo 1

Introduzione

Prima di immergersi nell'esposizione inerente alla progettazione ed allo sviluppo del sistema ivi trattato, si ritiene opportuno analizzare gli aspetti principali relativi al problema del riconoscimento degli accordi, della tonalità e delle melodie. A tal fine risulta appropriato illustrare lo stato dell'arte delle tecnologie attuali all'interno del panorama dei sistemi atti a costituire un ausilio al musicista durante la fase dell'esecuzione e composizione musicale. L'idea che ha guidato la creazione di tale supporto nasce dall'esperienza maturata nel campo musicale, essa difatti ha evidenziato come il problema del riconoscimento delle note eseguite, per chi si approccia allo studio di uno strumento musicale, sia quanto mai attuale e diffuso. Infine ci si appresta a descrivere la struttura generale del sistema ideato con l'ausilio di diagrammi.

1.1 Presentazione del progetto e dell'ambito nel quale si colloca

1.1.1 Informatica Musicale

Il presente lavoro di tesi si colloca nell'ambito dell'Informatica Musicale, settore dell'IT (Information Technology) che si occupa di produrre software e hardware di ausilio alla composizione, produzione e ottimizzazione della musica. L'Informatica Musicale si snoda in quattro microsettori principali che si dedicano alla ricerca e allo sviluppo di diversi aspetti della materia.

Audio signal processing

Uno di questi settori di ricerca è l'Audio Signal Processing, il quale si concentra sullo studio dell'analisi sonora, sugli algoritmi per campionare e pro-

cessare segnali audio e sulle tecniche di trasformazione e sintesi sonora. La sintesi sonora consiste nella creazione di suoni e timbriche mediante l'utilizzo di forme d'onda base (onde sinusoidali, triangolari, quadrate, dente di sega etc.) emesse dai c.d. oscillatori, allo scopo di produrre un segnale audio derivante dalla somma di esse. Nel presente lavoro viene utilizzato uno degli strumenti di analisi più diffusi nell'ambito musicale, la **Trasformata di Fourier**, che permette di scomporre il segnale audio proveniente da uno strumento musicale in una somma di forme d'onda più semplici rispetto al segnale di partenza, come le sinusoidi. Ciò permette di analizzare il contributo in ampiezza delle varie sinusoidi allo scopo di ottenere informazioni frequenziali sul segnale originario.

Riconoscimento e modellazione del suono

Questa sezione dell'Informatica Musicale si concentra sul capire e modellare la musica attraverso l'uso del calcolatore. Essa include la branca della **Music Information Retrieval** (MIR), la quale ha lo scopo, analogamente alla Image information retrieval per le immagini, di indicizzare le informazioni contenute all'interno di sequenze o brani musicali, in modo da poter sviluppare svariate applicazioni basate su di esse. Il MIR oggi ha acquisito molta rilevanza a livello internazionale, infatti tuttora migliaia di sviluppatori partecipano al MIREX (Music Information Retrieval Evaluation eXchange), concorso organizzato dall'International Music Information Retrieval Systems Evaluation Laboratory dell'Università dell'Illinois, che ogni anno pone degli obiettivi da raggiungere facendo competere i propri partecipanti. Tra i vari task proposti ogni anno vi è l'**Audio Chord Estimation**, cioè il riconoscimento accordi e l'**Audio Key Estimation**, il riconoscimento della tonalità, che sono due delle funzionalità trattate in questa tesi.

Sviluppo di interfacce

Lo sviluppo di interfacce si dedica principalmente alla progettazione di nuovi modi per interagire con la musica, creando nuove interfacce e strumenti che permettano al musicista di creare musica in maniera del tutto innovativa. In questa branca dell'Informatica Musicale sono compresi gli Iper-Strumenti, i quali possono essere definiti come "interfacce uomo-macchina basate sulle tecnologie 'touchless' dei raggi infrarossi e dell'analisi in tempo reale di immagini acquisite da telecamera. [...] L'idea consiste nel rilevare la gestualità delle mani e più in generale del corpo umano senza alcun collegamento fisico con il sistema in modo che il corpo umano diventi l'interfaccia naturale per dare espressività a performance artistiche basate su tecnologia informatica.

[...] I sensori che costituiscono l'interfaccia tra il performer e il sistema stesso non vengono collegati direttamente ai generatori sonori elettronici: in mezzo c'è il computer che interpreta i segnali provenienti dai sensori ed attiva i generatori sonori in funzione di programmi opportuni; **il programma attivo sul computer**, legge i dati relativi ai sensori e li mappa (e cioè li trasforma) in insiemi complessi di informazioni per controllare i dispositivi di generazione sonora." [Leonello Tarabella, 2010]
Questo tipo di strumenti, ancora in fase di ricerca e sviluppo, si trova a stretto contatto con il software attivo sul computer e il lavoro sviluppato in questa tesi si colloca a supporto sia degli strumenti tradizionali che degli Iper-strumenti.

Creazione di musica assisitita

È quella branca dell'Informatica Musicale orientata allo sviluppo di software in grado di assistere il musicista studente/compositore durante l'esecuzione o la creazione musicale. L'interfaccia utente creata in questa tesi offre diverse funzionalità al servizio del musicista.

1.1.2 Problematiche generali

Il riconoscimento e l'indicizzazione dell'audio e della musica in generale è da tempo uno dei più importanti e difficili obiettivi che la comunità internazionale sta cercando di raggiungere. Tale campo comprende tantissime problematiche e in questa sede ci si è soffermati sul riconoscimento degli accordi, sul riconoscimento della tonalità e sul riconoscimento delle note singole suonate da uno strumento solista.

Riconoscimento accordi

In musica si definisce accordo l'esecuzione contemporanea di tre o più note aventi altezze (frequenze) diverse. Le problematiche riscontrate durante la progettazione del riconoscitore di accordi sono state principalmente due: rilevare quando un nuovo accordo viene suonato e individuarne il tipo.
Per risolvere il primo punto si è reso necessario analizzare come il suono prodotto da uno strumento musicale si evolve nel tempo, accertamento che è stato possibile realizzare grazie al supporto del software di analisi Sonic Visualiser sviluppato dall'Università Queen's Mary di Londra. Come si può vedere in Figura 1.1, il suono nel tempo affronta diverse fasi riassumibili in quattro principali: l'attacco, il decadimento, il sustain e il rilascio. Durante

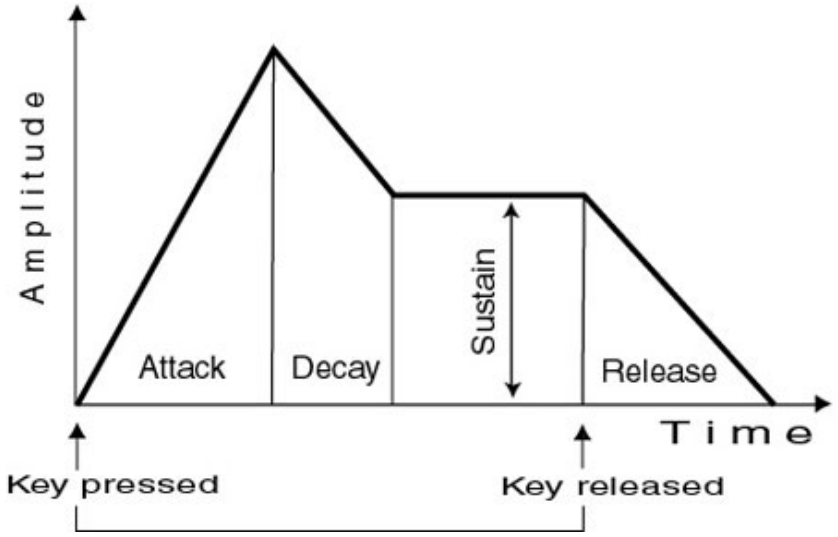

Figura 1.1: Grafico che descrive lo schema generale dell'evoluzione del suono, diviso in quattro fasi: Attacco, Decadimento, Sustain, Rilascio

la fase dell'attacco il suono ha un incremento repentino dell'intensità sonora fino a raggiungere un picco massimo, dopo il quale si entra nella fase successiva. Questa caratteristica viene sfruttata per riconoscere l'esecuzione di un nuovo accordo, infatti il sistema, prima di effettuare l'analisi, attende che il suono superi una certa soglia di intensità e che raggiunga il picco massimo. Solo il suono che supera il limite minimo di sensibilità viene riconosciuto dal sistema come un nuovo accordo. Tale soglia si può impostare a seconda della rumorosità dell'ambiente.
Una volta rilevata la presenza di un nuovo accordo occorre individuarne il tipo. Durante la fase dell'attacco una buona parte degli strumenti musicali presenta del rumore dovuto alla conformazione dello strumento stesso e al modo con cui viene suonato (ad esempio: il plettro della chitarra quando pizzica le corde o i martelletti del pianoforte quando percuotono le corde introducono rumore). Con l'ausilio di Sonic Visualiser e dell'interfaccia sviluppata in questa tesi è stato possibile individuare tale problema e la sua soluzione: il rumore, molto presente durante la fase di attacco, nella fase di decadimento si attenua molto più velocemente rispetto alle armoniche che compongono l'accordo. Alla luce di tale comportamento è stata individuata la finestra temporale più adatta ad effettuare l'analisi alla ricerca dell'accordo.

Riconoscimento tonalità

Nell'ambito del riconoscimento della tonalità sono stati svolti degli studi di teoria musicale allo scopo di acquisire le conoscenze indispensabili per risolvere il problema. In particolare ci si è concentrati sulla nozione di accordo, sulle tipologie di accordo esistenti e il loro ruolo all'interno della tonalità.
Come si è detto in precedenza un accordo è costituito dall'esecuzione simultanea di tre o più note aventi altezze diverse. La distanza in termini di altezza che esiste tra le note dell'accordo ne determina il tipo. Ad esempio: l'accordo di C maggiore (Do maggiore) è caratterizzato dalle note C, E e G (Do, Mi e Sol) come mostrato in Figura 1.2. Cambiando la distanza tra la prima e

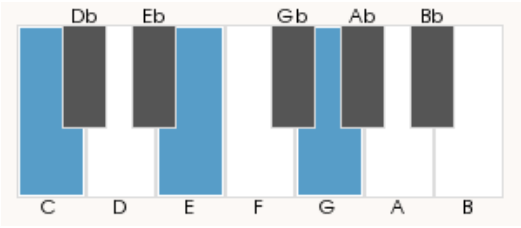

Figura 1.2: Accordo di C maggiore (Do maggiore)

la seconda nota del C maggiore si può ottenere un nuovo tipo di accordo, il C minore, formato dalle note C, Eb e G (cfr. Figura 1.3). Ne risulta che

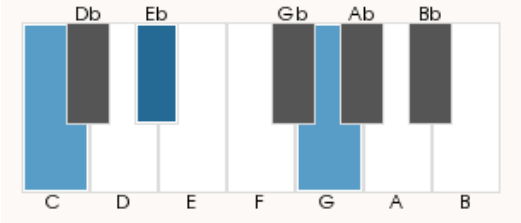

Figura 1.3: Accordo di C minore (Do minore)

cambiando anche solo una delle note di un dato accordo se ne otterrà uno diverso.
Mettendo in relazione una successione di accordi differenti, si può introdurre la nozione di **tonalità**, concetto paragonabile, facendo un'analogia con il mondo della matematica, ad un dominio nel quale gli arrangiamenti e le melodie si possono muovere (quasi) liberamente. Per spiegare meglio questo concetto si prenda in considerazione la Tabella 1.1 in cui sono descritte alcune tonalità e gli accordi che le caratterizzano: in ogni tonalità si possono identificare sette gradi, ciascuno dei quali è rappresentato da un accordo. Se

Tonalità	I	II	III	IV	V	VI	VII
C	**Cmaj7**	Dm7	**Em7**	Fmaj7	G7	**Am7**	Bø
G	Gmaj7	**Am7**	Bm7	**Cmaj7**	D7	**Em7**	Gbø

Tabella 1.1: Accordi appartenenti alle tonalità

si eseguisse una successione di accordi estratti dalla prima riga della tabella, ad esempio Dm7 - Em7 - Am7, tale successione può essere definita come appartenente alla tonalità di C.
Nella Tabella 1.1 sono stati evidenziati in grassetto gli accordi in comune tra le tonalità di C e G. Vi è un aspetto problematico da tenere in considerazione: in riferimento all'esempio precedente, laddove si esaminassero esclusivamente gli accordi di Em7 - Am7, senza la presenza dirimente di Dm7, si andrebbe incontro ad uno stato di incertezza circa l'identificazione della tonalità, poiché questi due accordi da soli appartengono anche alla tonalità di G.
Ai fini della corretta stima della tonalità attuale la scelta della finestra temporale sulla quale effettuare l'analisi è di fondamentale importanza, ragion per cui, a seconda della necessità del musicista e della complessità del brano analizzato, la finestra temporale è stata resa modificabile a seconda dei parametri necessari, di modo che possa essere estesa o ridotta al bisogno. Sempre in riferimento all'esempio precedente la finestra temporale analizzata, in principio, aveva dimensione tre (Dm7 - Em7 - Am7) considerando quindi le tre frazioni temporali corrispondenti ad ogni accordo. Successivamente veniva limitata a due frazioni temporali, peggiorando la precisione del risultato e portando il sistema ad uno stato di incertezza tra la tonalità C e G.

1.1.3 Perfezionamento del musicista

L'obiettivo di questo lavoro consiste nello sviluppo di un sistema di supporto all'improvvisazione musicale per il musicista che vi si cimenta. L'atto dell'improvvisazione si caratterizza per una composizione istantanea e creativa, priva di una scrittura antecedente che viene eseguita su di una base musicale. Essa dipende fortemente dall'emozione e dalla sensibilità del musicista, il quale deve però sottostare a determinati vincoli dettati dalle caratteristiche della musica sulla quale si trova ad improvvisare. Una di queste caratteristiche è la tonalità che deve essere ben nota al musicista che si cimenta nell'improvvisazione: infatti ad ogni tonalità è possibile associare una scala musicale che racchiude al suo interno le note che l'improvvisatore può eseguire coerentemente con il brano che sta interpretando.
Un musicista esperto ha dunque l'opportunità di utilizzare il sistema al fi-

ne di perfezionare la sua preparazione musicale, anche ampliando le proprie conoscenze nell'ambito dei diversi generi musicali (ad esempio il Blues).

1.1.4 Utilizzo per scopi didattici

Il software sviluppato nell'ambito di questa di tesi è altresì pensato per il musicista neofita che, acquisite le conoscenze musicali di base tecniche e teoriche, desideri esercitarsi nell'applicazione di ciò che ha appreso durante i suoi studi preliminari. Grazie alla possiblità di scegliere il tipo di scala da seguire nell'utilizzo del sistema, egli ha l'opportunità di impostare il proprio studio da autodidatta partendo dalle scale più semplici con meno note, per poi cimentarsi in esecuzioni più complesse, il tutto con l'aiuto e la supervisione del software che tramite l'interfaccia e l'uso appropriato dei colori segnala all'utente quando questi esegue le note in maniera corretta o commette delle imperfezioni.
Nell'ambito di un corso musicale in presenza di un docente, il software può essere usato per integrare le lezioni e per permettere allo studente di esercitarsi nell'applicazione delle conoscenze teoriche acquisite. Questo è un aspetto molto importante nello studio della musica, perché permette allo studente di ottenere un riscontro quasi immediato della teoria, oggetto di studio, nella pratica, oggetto di esercitazione.

1.2 Progetti consultati

Di seguito verranno descritti i progetti più significativi che sono stati consultati ai fini di comprendere lo stato dell'arte in materia di riconoscimento armonico e melodico.

1.2.1 NNLS-Chroma e Chordino

Un progetto analogo nasce nel 2010 come lavoro di tesi di Matthias Mauch [Matthias Mauch, 2010], il quale si prefigge come obiettivo l'implementazione di un sistema di analisi della musica volto a riconoscere gli accordi presenti in un brano musicale ma non al momento della sua esecuzione. Il suo riconoscimento infatti si svolge su di una registrazione audio preesistente rispetto al momento dell'analisi. Esso si divide in due moduli fondamentali: NNLS-Chroma e Chordino. Il primo analizza il file audio ricevuto in ingresso e per ogni sample analizzato costruisce il chromagram, struttura dati formata da dodici elementi rappresentanti le dodici note della scala nelle ottave alte, e il bass chromagram, struttura dati formata da dodici elementi rappresentanti le

dodici note della scala nelle ottave basse. Chordino invece elabora i dati ottenuti dall'analisi di NNLS-Chroma e tenta di stimare, mediante un approccio probabilistico e l'uso di un dizionario di accordi, gli accordi che compongono il brano. In Figura 1.4 vi è la rappresentazione grafica dell'elaborazione dei dati prodotta da NNLS-Chroma che utilizza i colori per evidenziare la presenza o meno di determinate note e la successiva stima degli accordi prodotta da Chordino.
Ci si è ispirati al lavoro di Mauch per sviluppare le strutture dati prendendo spunto dal chromagram e il dizionario accordi, il quale è stato modificato per assecondare i bisogni di questo lavoro.
Il limite principale di NNLS-Chroma e Chordino consiste nell'impossibilità di utilizzarlo in real-time, ossia durante l'esecuzione del musicista, poiché è implementato per analizzare esclusivamente un prodotto musicale già finito.

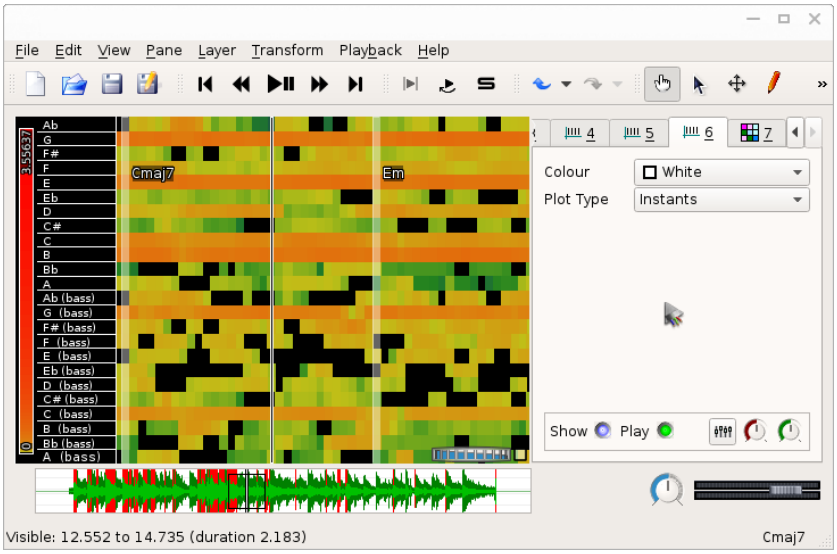

Figura 1.4: Chordino e NNLS-Chroma

1.2.2 ChucK: Strongly-timed, Concurrent, and On-the-fly Music Programming Language

ChucK è un linguaggio di programmazione orientato all'analisi e alla produzione musicale che offre al programmatore numerosi framework per facilitare lo sviluppo di applicazioni. ChucK è frutto della tesi di Ge Wang presentata per la laurea ottenuta presso l'Università di Princeton nel 2008 [Wang Ge, 2008]. Oggi è assistente professore all'Università di Stanford e

lavora presso il dipartimento di Musica.
Il sistema di riconoscimento armonico e melodico qui trattato è stato interamente sviluppato mediante il linguaggio ChucK.

1.2.3 Sonic Visualiser

Sonic Visualiser è un software opensource distribuito sotto la licenza GNU General Public License v2 atto alla visualizzazione ed all'analisi del contenuto dei file audio musicali. Le sue capacità sono state sfruttate al fine di comprendere il comportamento delle forme d'onda dei vari strumenti musicali studiati e del loro relativo spettro delle frequenze. In particolare è servito per analizzare le forme d'onda di pianoforte, chitarra acustica, vibrafono e organo a canne ed il loro relativo spettro. L'acquisizione di questi dati e la loro comprensione hanno apportato un contributo determinante nella progettazione e nello sviluppo del sistema di riconoscimento.

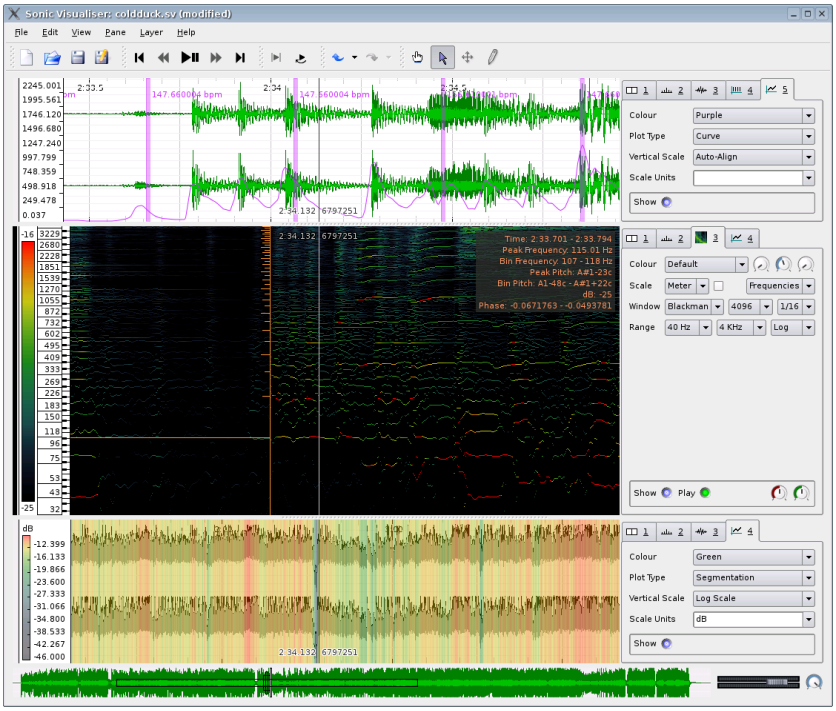

Figura 1.5: Interfaccia di Sonic Visualiser

1.3 Struttura del sistema

Il sistema di riconoscimento è stato sviluppato mediante il sistema operativo libero e opensource **ArchLinux**, al quale fa capo una community molto numerosa e attiva nel campo dell'Informatica musicale che mette a disposizione dei programmatori numerosi tool di sviluppo atti all'analisi e alla produzione musicale.
Gli aspetti inerenti al riconoscimento armonico e melodico sono stati interamente sviluppati mediante il linguaggio di programmazione musicale **ChucK**. In particolare nel Capitolo 2 verrà descritto il metodo utilizzato per l'analisi sonora e il riconoscimento armonico e melodico, le scelte progettuali prese per risolvere determinate problematiche, per poi concludere con la descrizione dell'implementazione.
I dati prodotti dall'analisi effettuata dal software programmato in ChucK verranno presentati tramite un'interfaccia grafica realizzata utilizzando il linguaggio **Python** supportato dalle librerie grafiche **Qt**. L'interfaccia utente ed il suo utilizzo saranno descritti in maniera dettagliata nel Capitolo 3.
Nell'effettuare il test per la verifica del funzionamento di tutto il sistema ci si è serviti di numerosi software. Tra questi **AriaMaestosa** è un programma che permette la scrittura di brani musicali in linguaggio MIDI con la possibilità di trasmetterli a **FluidSynth**, altro software in grado di generare i suoni musicali. Questi due programmi sono stati utilizzati per creare la base musicale del test. **JACK Audio Connection Kit** è un software che gestisce la scheda audio e semplifica la sua connessione con i software applicativi. In particolar modo si è scelto questo tool per collegare la base costruita mediante AriaMaestosa e FluidSynth e gli ingressi della scheda audio con il software programmato in ChucK. La struttura del test e i risultati ottenuti saranno oggetto del Capitolo 4.

Capitolo 2

Riconoscimento armonico e melodico

Questo capitolo descrive il metodo utilizzato per effettuare l'analisi sonora e il riconoscimento armonico e melodico. In Figura 2.1 è possibile vedere la struttura generale del sistema.

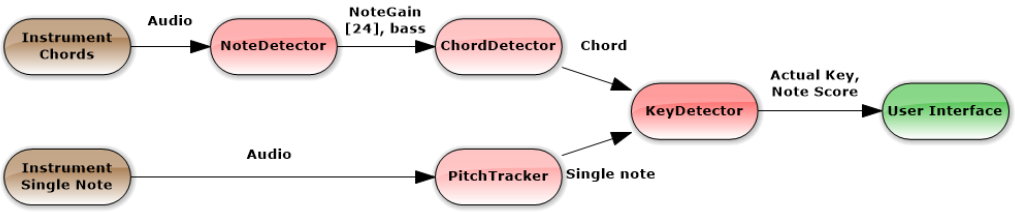

Figura 2.1: Struttura generale del Riconoscimento Armonico e melodico

Come si può notare, il sistema accetta in ingresso due strumenti musicali: uno che esegue l'accompagnamento come successione di accordi e l'altro che esegue invece delle parti melodiche formate da successioni di note.
Il flusso audio dell'accompagnamento viene ricevuto in ingresso dal **NoteDetector**, il quale, mediante l'analisi spettrale, identifica il contributo di ogni nota dell'accordo e la sua nota più bassa in termini di frequenza. Questi dati, un volta ottenuti, vengono mandati in ingresso al **ChordDetector** che individua l'accordo attuale. Infine, l'accordo rilevato viene a sua volta ricevuto dal **KeyDetector** che stima la tonalità in cui lo strumento che accompagna sta suonando.
Il flusso audio del solista, invece, viene dato in input al **PitchTracker** che identifica la nota suonata dallo strumento. Questa nota viene poi data al **KeyDetector** per poter essere confrontata con la tonalità attuale. In ca-

so la nota risulti all'interno della tonalità attuale (con opportuni parametri aggiuntivi descritti in seguito), essa verrà valutata come corretta, altrimenti come non corretta.

2.1 Analisi Spettrale

Per poter riconoscere gli accordi e fare delle stime sulla tonalità attuale, si è ritenuto necessario effettuare l'analisi spettrale del flusso audio proveniente dagli strumenti musicali, sfruttando il potente strumento della Trasformata Veloce di Fourier, il quale, con i dovuti accorgimenti, fornisce un'analisi molto dettagliata del segnale in questione.
La trasformata di Fourier permette di scomporre qualsiasi segnale analogico in una somma di sinusoidi con frequenze e ampiezze diverse. Questa caratteristica permette di analizzare un segnale audio e determinare quali componenti frequenziali sono presenti al suo interno.

2.1.1 Approccio utilizzato

La scheda audio utilizzata per il campionamento del segnale audio possiede una frequenza di campionamento di $48000Hz$, il che permette di analizzare una banda del segnale pari a circa la metà, cioè $24000Hz$. Effettuando una FFT con finestra a 4096 punti e dividendo il sample rate per questo numero si ottiene:

$$fftStep = \frac{48000Hz}{4096} = 11.71875Hz$$

Questo importante dato esprime la distanza tra due punti della FFT, che chiamerò $fftStep$. Osservando la Figura 2.2 è necessario evidenziare che più le note hanno una frequenza bassa, più sono vicine tra loro.
Considerando le note F_2 ed E_2 che hanno una differenza di $87.3070Hz - 82.4070Hz = 4.9Hz$ e confrontandole con $fftStep = 11.71875Hz$, la risoluzione data da questa scelta di $fftStep$ non è sufficiente a garantire il riconoscimento di queste due note, poiché entrambe capitano tra due punti adiacenti della FFT. L'obiettivo è riconoscere almeno le note dell'ottava 1, cioè F_1 ed E_1 che hanno una differenza di $43.6540Hz - 41.2030Hz = 2.451Hz$.
Raddoppiando il numero dei punti della finestra FFT sino a 8192 si raggiungerebbe un $fftStep = 5.859375Hz$, che non sarebbe ancora sufficiente.
Tenendo conto che le note suonate negli accordi solitamente non superano i 4000Hz, la soluzione è stata effettuare il downsampling del segnale da $48000Hz$ a $12000Hz$, portando la banda analizzata del segnale a $6000Hz$, mettere un filtro passa basso per evitare l'effetto aliasing e calcolare l'FFT

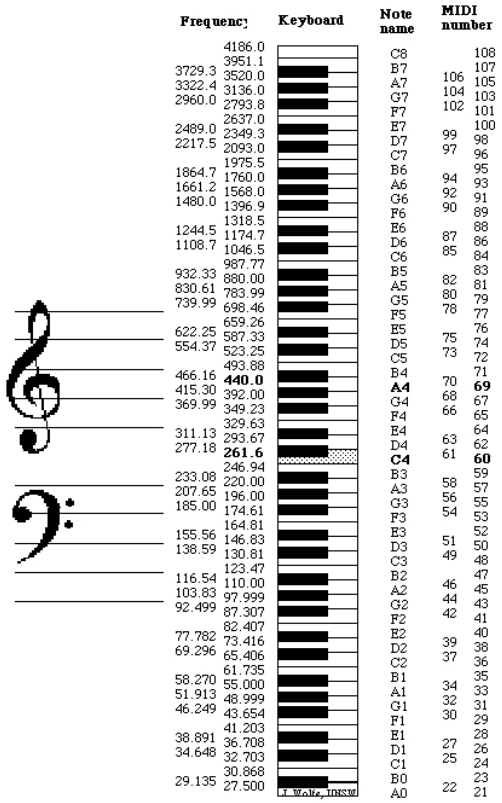

Figura 2.2: Note musicali sul piano e relative frequenze

con 8192 punti:

$$fftStep = \frac{12000Hz}{8192} = 1.46484Hz$$

L'aliasing è il fenomeno per il quale due forme d'onda diverse possono diventare indistinguibili una volta campionate, questo succede quando il segnale in ingresso è sottocampionato. Per evitare ambiguità il filtro passa basso anti-aliasing è stato impostato ad una frequenza di taglio pari a $6000Hz$.
Con questo sistema si è raggiunto un compromesso: da un lato la banda analizzata del segnale è diminuita da 24000Hz a 6000Hz, dall'altro i punti della FFT sono piú vicini tra loro e il sistema è in grado di riconoscere anche le note nell'ottava bassa, importantissime per il riconoscimento degli accordi.
A questo punto i dati della FFT vengono mappati in un vettore di 128 elementi, rappresentanti tutte le note previste dal protocollo MIDI, in cui è possibile vedere il contributo frequenziale di ogni nota. Dopodiché, come verrà affrontato nello specifico più avanti, le note con lo stesso nome ma in ottave diverse vengono raggruppate in un'unica struttura dati di 24 elementi

(cfr. Figura 2.3), di cui 12 che rappresentano i toni delle ottave basse (dalla 0 alla 3) e 12 che rappresentano i toni delle ottave alte (dalla 4 alla 7).

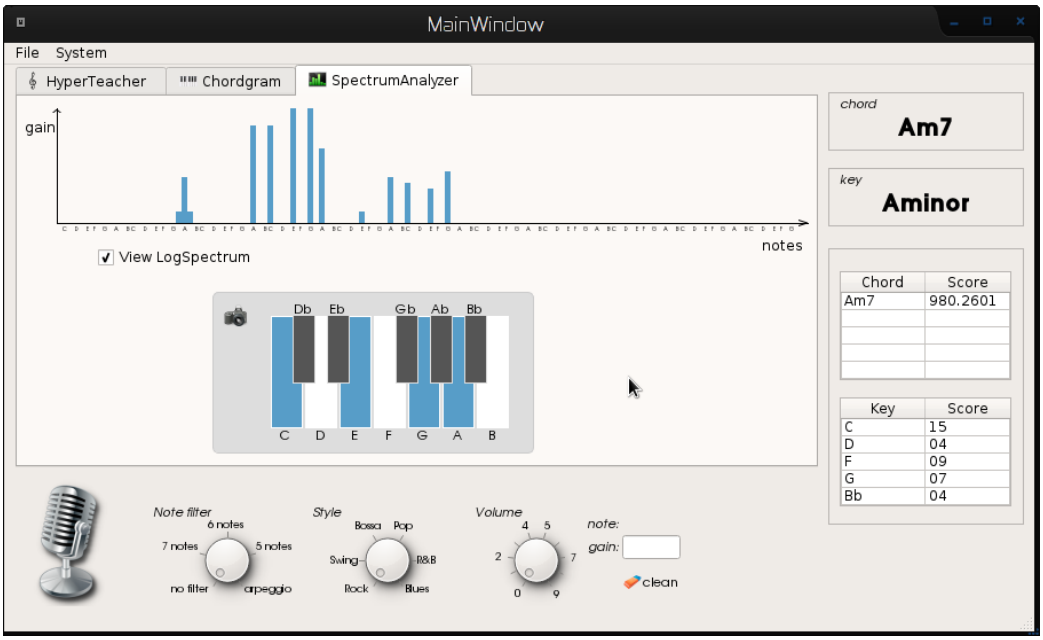

Figura 2.3: Il grafico in alto rappresenta le note contenute in *notes*[128] che vengono poi mappate in *NoteGain*[24], rappresentato in basso

2.1.2 NoteDetector

L'oggetto **NoteDetector** attraverso la trasformata veloce di Fourier (FFT: Fast Fourier Transform) fornisce una serie di funzionalità e dati utili allo scopo del riconoscimento armonico, in particolare produce un vettore *Notegain[24]* nel quale è contenuto il relativo contributo frequenziale di ogni semitono.

::analyseLogFreq(spectrum, fft_size)

Questo metodo prende in ingresso i dati ottenuti dal calcolo della FFT (*spectrum* è lo spettro, *fft_size* è la grandezza della finestra FFT), mappa il loro contenuto nell'attributo *notes[128]* affinché il contributo in ampiezza di ogni nota venga identificato in una cella di questo array.
A questo punto nei primi 12 elementi dell'array *NoteGain[24]* vengono mappati tutti i contributi delle singole note nelle ottave basse (dalla 0 alla 3) e nei restanti 12 elementi vengono mappati i contributi delle note nelle ottave

più alte (dalla 4 alla 7).
Es: Facendo riferimento all'espressione

$$i < 12 : NoteGain[i] = \sum_{k=0}^{3} notes[i + k * 12]$$

$$i >= 12 : NoteGain[i] = \sum_{k=4}^{7} notes[i + k * 12]$$

NoteGain[0] sarà calcolato come $C_0 + C_1 + C_2 + C_3$, il cui risultato è da interpretare come il contributo totale della nota C nelle ottave basse. Infine il metodo si preoccupa di trovare il massimo valore tra i primi 12 elementi di NoteGain, il quale sarà considerato come il basso dell'accordo attuale e verrà utilizzato per il riconoscimento dello stesso.

::integrate()

Il metodo *integrate* somma *NoteGain[]* con un numero *step* di istanti precedenti
$NoteGainIntegrate_t = \sum_{k=t-step}^{t} NoteGain$
Questo per cercare di limitare le oscillazioni che le varie armoniche generate dalle note presentano durante l'inviluppo del suono.

2.2 Riconoscitore degli Accordi

Una volta eseguita l'analisi del segnale, grazie alla quale è stata preparata la struttura dati $NoteGain[24]$ ed è stata trovata la nota più bassa, si può procedere al riconoscimento dell'accordo. In un dato istante ci sarà un certo contributo frequenziale delle note musicali, dal quale si può estrarre l'accordo attuale. Per poterlo estrarre è stato progettato un semplice algoritmo di ranking che, incrociando i dati estratti dall'analisi spettrale con un dizionario di accordi, individua dei possibili candidati.

2.2.1 Dizionario accordi

Nel dizionario accordi sono presenti 7 classi di accordi:

maggiori (0):
vi appartengono gli accordi che presentano la terza maggiore e la settima maggiore. Es: Dmaj7 (Re maggiore con settima maggiore)

minori (1):
vi appartengono gli accordi che presentano la terza minore e la quinta giusta. Es: Em (Mi minore), Gbm7 (Sol bemolle minore settima)

settima (2):
vi appartengono tutti gli accordi di settima di dominante. Es: G7 (Sol settima), E13 (Mi tredicesima, è considerato di settima perché possiede 3, 5, 7, 9, 11 e 13)

semidiminuiti (3):
vi appartengono gli accordi semidiminuiti, essi possiedono la 3 minore, la 5 bemolle e la 7

diminuiti (4):
vi appartengono tutti gli accordi diminuiti, hanno 3 minore, 5 bemolle e 7 bemolle

sospesi (5):
vi appartengono gli accordi sospesi. Es: Csus9 (Do sospeso con la nona)

maggiori o settima (6):
vi appartengono gli accordi che potrebbero essere considerati, a seconda del contesto, maggiori o settima. Di solito sono delle triadi, cioè composti da tre note. Es: C (Do maggiore), G (Sol maggiore)

Successivamente, prendendo spunto dal dizionario sviluppato da Matthias Mauch in Chordino [Matthias Mauch, 2010], sono state create delle maschere di bit, una per ogni tipo di accordo. Prendendo come esempio:

$$0\_maj7 = 1,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,0,0,1,0,0,0,1$$

è la riga corrispondente al tipo di accordo maggiore con settima maggiore. Nel dettaglio:

0
: il numero prima dell'underscore identifica la classe dell'accordo, in questo caso è un accordo *maggiore* (classe 0)

$maj7$
: la stringa successiva prima dell'uguale indica il tipo di accordo, in questo caso *Maggiore con settima maggiore*

$1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0$
: i 12 numeri riportati indicano la presenza (1) o meno (0) di una nota nel dato accordo e si riferiscono alle note medio-basse, nell'esempio è presente solo la tonica

$1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1$
: le ultime 12 sono relative alle note medio-alte. Qui è presente la tonica, la terza maggiore, la quinta e la settima maggiore.

La maschera va sempre vista in relazione alla tonica dell'accordo, cioè la nota che gli da il nome. Ad es: Dmaj7 possiede la stessa maschera sopracitata, i bit della quale vanno visti come intervalli relativi alla sua tonica, cioè il D (Re). Perciò partendo dal D e avendo la maschera $1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1$, possiamo dire che questo accordo possiede il D (il primo bit), Gb (il quinto bit), A (l'ottavo bit) e il Db (dodicesimo bit).

2.2.2 Algoritmo di Ranking

Per ognuno dei 12 semitoni della scala viene caricato il dizionario degli accordi nella struttura dati $chordNotes[12][nTypes][24]$, dove

12
: sono le note musicali comprese quelle alterate,

nTypes
: è il numero di tipi di accordo presenti in dizionario,

24
: è il numero di bit di ogni maschera,

in modo che ogni nota possa avere tutti i tipi di accordo previsti. Gli input all'algoritmo sono: $NoteGain[24]$ in cui sono presenti i contributi frequenziali di tutte le note e *bass* ossia l'indice (compreso tra 0 e 11) della nota di basso rilevata dal NoteDetector.
A questo punto, dato $Notegain[24]$ in cui è contenuto il contributo di ogni nota e $chordNotes[12][nTypes][24]$, la struttura dati degli accordi appena creata, si calcola il punteggio dell'accordo:

$$score_{tone,type} = \frac{\sum_{i=12}^{23} NoteGain[i] \cdot chordNotes[tone][type][i]}{arctg(\sum_{j=12}^{23} NoteGain[j])}$$

L'accordo con il punteggio più alto vince il Ranking e viene scelto come accordo attuale. Nel caso di ex-equo viene aggiunto il punteggio della nota del basso in questo modo

$$score_{tone,type} = score_{tone,type} + NoteGain[bass] \cdot chordNotes[tone][type][bass]$$

Da notare che in quest'ultimo passaggio solo l'accordo che ha il basso uguale a *bass* riceve il punteggio aggiuntivo.

2.2.3 ChordDetector

Per implementare il riconoscitore di accordi è stata creata la classe **ChordDetector**, la quale si preoccupa di caricare il dizionario degli accordi, ricevere in ingresso il vettore **NoteGain[24]** e restituire in uscita l'accordo trovato e la sua classe.

::readChords(string path)

Tale metodo si preoccupa di leggere il file la cui path è contenuta in *path* e salvare il contenuto nella struttura **chordNotes[12][nTypes][24]**.

::analyseChord(float NoteGain[])

Una volta caricato il dizionario si può iniziare l'analisi per trovare l'accordo attuale. Questo metodo prende in ingresso il vettore **NoteGain[24]** considerando gli ultimi 12 elementi e calcola il punteggio di tutti gli accordi come descritto precedentemente nell'algoritmo per calcolare il Ranking. In caso ci siano più accordi con lo stesso punteggio più alto crea una lista di candidati, la quale puo' essere ottenuta con il metodo *getChordCandidates()*.

2.3 Riconoscitore della tonalità

Grazie ai dati forniti dal Riconoscitore Accordi, il Riconoscitore della tonalità cerca di determinare la tonalità attuale mediante un algoritmo di riconoscimento basato sull'accordo attuale e sugli accordi rilevati in precedenza entro una certa finestra temporale.

2.3.1 La tonalità

Come detto in precedenza una successione di accordi genera una **tonalità** alla quale è sempre associato il concetto di **scala musicale**, che rappresenta

le note appartenenti alla tonalità attuale. La scala può essere vista come un insieme di note, in certi casi sottoposte a dei vincoli sulla loro successione, che gli strumenti melodici possono eseguire senza risultare "stonati", effetto che solitamente si verifica quando alcune delle note suonate risultano al di fuori della scala presa in considerazione.

2.3.2 Algoritmo di riconoscimento

L'algoritmo di riconoscimento tenta di stimare la tonalità analizzando una finestra temporale di n accordi, nel seguente modo:

1. Una volta determinata la classe dell'accordo, viene dato un punto alle tonalità alle quali potrebbe appartenere.
Es: facendo riferimento alla Tabella 2.1 Cmaj7 appartiene alla classe degli accordi maggiori, i quali, nelle tonalità maggiori sono presenti al I grado e al IV grado della scala, perciò in questo caso l'algoritmo assegna un punto alla tonalità di C (Cmaj7 è il I grado) e di G (Cmaj7 è il IV grado).

Tonalità	I	II	III	IV	V	VI	VII
C	**Cmaj7**	Dm7	Em7	Fmaj7	G7	Am7	Bø
Db	Dbmaj7	Ebm7	Fm7	Gbmaj7	Ab7	Bbm7	Cø
D	Dmaj7	Em7	Gbm7	Gmaj7	A7	Bm7	Dbø
Eb	Ebmaj7	Fm7	Gm7	Abmaj7	Bb7	Cm7	Dø
E	Emaj7	Gbm7	Abm7	Amaj7	B7	Dbm7	Ebø
F	Fmaj7	Gm7	Am7	Bbmaj7	C7	Dm7	Eø
Gb	Gbmaj7	Abm7	Bbm7	Bmaj7	Db7	Ebm7	Fø
G	Gmaj7	Am7	Bm7	**Cmaj7**	D7	Em7	Gbø
Ab	Abmaj7	Bbm7	Cm7	Dbmaj7	Eb7	Fm7	Gø
A	Amaj7	Bm7	Dbm7	Dmaj7	E7	Gbm7	Abø
Bb	Bbmaj7	Cm7	Dm7	Ebmaj7	F7	Gm7	Aø
B	Bmaj7	Dbm7	Ebm7	Emaj7	Gb7	Abm7	Bbø

Tabella 2.1: Accordi appartenenti alle tonalità

2. Il punteggio è sempre relativo alla finestra di n accordi. Ad ogni nuovo accordo suonato, quindi, i punteggi vengono aggiornati aggiungendo i punti dell'accordo attuale e sottraendo quelli dell'$(n-1)$esimo accordo passato, ossia il più vecchio.

3. In caso di più tonalità con lo stesso punteggio vincente, viene creata una lista di tonalità candidate

2.3.3 Valutazione della nota del solista

Un ulteriore funzionalità di questo oggetto consiste nel ricevere in ingresso una nota singola, confrontarla con l'attuale tonalità e i parametri dell'oggetto e infine valutare la nota come corretta o non corretta.
Nella valutazione è possibile impostare due opzioni:

Filtro note
possiede 5 modalità:

no filter
la nota non viene filtrata, e viene sempre valutata come corretta

7 notes
verranno ammesse tutte le 7 note della scala maggiore, le altre vengono valutate come non corrette

6 notes
verranno ammesse 5 note della scala maggiore (I, II, III, V, VII) più una nota selezionata secondo il seguente criterio:
gli accordi relativi ad una tonalità maggiore si possono dividere in due gruppi:

Gruppo A
I, III, VI

Gruppo B
II, IV, V, VII

Se l'accordo attualmente suonato fa parte del Gruppo A, al set di note viene aggiunta la VI.
Se invece l'accordo attualmente suonato fa parte del Gruppo B, al set di note viene aggiunta la IV. Nel Gruppo A non viene ammessa la IV poiché negli accordi appartenenti al gruppo è sempre presente la III che in termini di frequenza dista dalla IV di un semitono. L'eventuale contemporaneità di queste due note estremamente contigue genererebbe una dissonanza.

5 notes
verranno ammesse 5 note della scala maggiore (I, II, III, V, VI), la cosiddetta "scala pentatonica"

arpeggio
verranno ammesse solo le note che compongono l'accordo attuale.

Stile
per ora solo lo stile Blues è implementato e aggiunge alle note am-

messe dalla scala pentatonica la nota Blues, cioè, riferendosi alla scala maggiore, il II# grado (secondo grado aumentato).

2.3.4 KeyDetector

In questo paragrafo verrà descritto l'oggetto **KeyDetector**, il quale si occupa del riconoscimento della tonalità, per farlo riceve in input gli accordi rilevati da ChordDetector.

::keyUpdate(int *tone*, **int** *class_*, **int** *actualChordNotes*[])

Questo metodo riceve in ingresso l'intero *tone*, il quale identifica la tonica dell'accordo (C, D, E, etc.), l'intero *class_* che identifica la classe dell'accordo (maggiore, minore, settima, etc.) e un array *actualChordNotes*[] nel quale sono contenute le note dell'accordo.
Come illustrato in precedenza nella Tabella 2.1, a seconda del valore di *class_* e di *tone*, vengono assegnati dei punti alle tonalità che potrebbero comprenderli.
Es: nel caso l'accordo riconosciuto da ChordDetector sia Cm7, *tone* assumerà il valore 0, cioè la nota C, e *class_* il valore 1, cioè la classe degli accordi minori (identificata dall'1). A questo punto le tonalità che comprendono l'accordo Cm7 incrementano di 1 il loro punteggio, in questo caso sono le tonalità di Eb, Ab e Bb.
La struttura dati dove vengono salvati i punteggi è *key*[12]. Oltre ad aggiungere un punto a queste tonalità si sottrae il punto ottenuto all'$(n-1)$esimo istante precedente. Gli istanti precedenti vengono tutti salvati in $keyHistory[n][12]$, il quale viene usato come un buffer circolare aggiornando opportunamente gli indici $last\_item$ e $first\_item$ che indicano rispettivamente l'ultimo elemento e il primo elemento del buffer.
Il contenuto di *actualChordNotes*[] viene memorizzato all'interno dell'oggetto e utilizzato dal metodo **evaluateNote()** solo nel caso in cui il Filtro Note sia impostato su Arpeggio.

::evaluateNote(int *note*)

Questo è il metodo che si preoccupa di controllare, come descritto in precedenza, se la nota *note* è una nota in scala, cioè appartenente alla tonalità attuale, secondo i parametri di Stile e Filtro Note. Restituisce 1 se la nota è in scala, altrimenti -1.

::getActualKey()

Questo metodo restituisce la tonalità attuale, calcolata precedentemente come il massimo tra gli elementi di $key[12]$.

2.4 Riconoscimento Melodico del solista

Per il riconoscimento melodico il linguaggio mette a disposizione l'oggetto **PitchTracker**. Esso implementa l'algoritmo di Helmholtz che riesce a identificare la frequenza fondamentale della singola nota suonata. Tramite poi la funzione di libreria $Std.ftom(freq)$, che data una frequenza resituisce la nota MIDI corrispondente (compresa tra 0 e 127), si può calcolare la nota suonata come resto della divisione per 12, dal quale si ottiene un numero compreso tra 0 e 11 (0 - C, 1 - Db, 2 - D, etc.). Dividendo per 12 invece si ottiene l'ottava corrispondente.

Capitolo 3

Interfaccia Utente

In questo capitolo verrà descritta l'interfaccia utente implementata per interagire con il sistema. Essa è stata realizzata utilizzando le librerie **Qt5** all'interno del linguaggio **Python3**. La parte grafica è stata sviluppata attraverso il tool grafico **Qt Designer**, il quale mette a disposizione tantissimi oggetti che poi sono stati inseriti all'interno dell'interfaccia. Per poterla integrare all'interno del linguaggio Python3, si è utilizzato il progetto opensource **pyQt**, il quale offre un tool, *pyuic5*, che converte il file creato attraverso Qt Designer direttamente in linguaggio Python. Il risultato è un file in linguaggio Python che è possibile includere nel proprio file sorgente, in modo da poter utilizzare l'interfaccia all'interno del programma. Si ottiene così la possibilità di interagire con la finestra attraverso il progetto **pyQt**, che mette a disposizione del programmatore numerose funzionalità che permettono di creare contenuti dinamici all'interno dell'interfaccia grafica.

Prendendo come riferimento Figura 3.1, verrà ora analizzata la parte statica dell'interfaccia la quale è sempre presente in tutte le schermate poiché le sue funzionalità sono sempre necessarie durante l'utilizzo. In essa sono presenti i controlli per il sistema di analisi audio, che permettono di modificare i parametri come Note Filter, Style e Volume descritti nel precedente capitolo. Da sinistra troviamo:

il microfono se cliccato, comunica al sistema audio di iniziare o terminare l'analisi. È stato necessario inserire un pulsante di start e stop in modo da semplificare l'utilizzo da parte dell'utente. Una volta premuto start il sistema inizia a riconoscere note e accordi e lo spartito inizia a scorrere orizzontalmente; premendo di nuovo il microfono il riconoscimento si ferma e lo spartito smette di scorrere, permettendo all'utente di poter consultare ciò che è stato rilevato durante la sua esecuzione musicale.

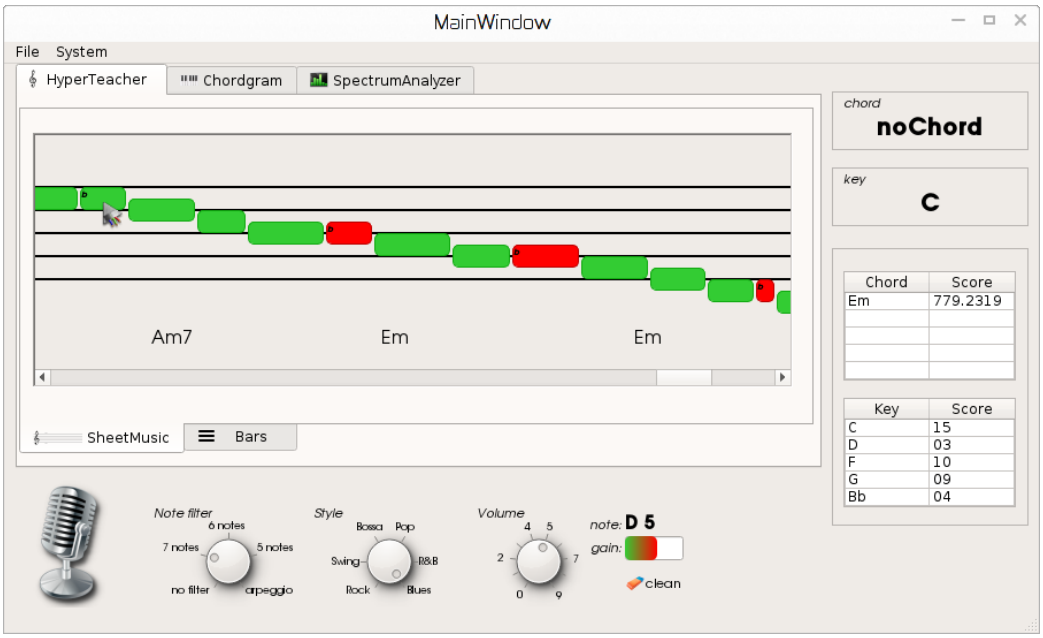

Figura 3.1: Schermata principale: sullo spartito sono presenti le note suonate dal solista e in basso gli accordi rilevati

Note Filter comunica il livello di filtraggio delle note. Di default è impostato su *no filter*, quindi tutte le note che verranno suonate verranno visualizzate in verde. Nel caso sia impostato su di un qualsiasi altro filtro, le note corrette verranno visualizzate in colore verde. Invece nel caso in cui la nota eseguita non appaia coerente con la tonalità rilevata e le impostazioni selezionate, questa sarà visualizzata in rosso. Il filtro permette di selezionare la scala sulla quale il musicista vuole esercitarsi. Ruotando il selettore in senso orario il sistema utilizzerà un filtro gradualmente sempre più rigido nell'individuazione delle note valutate come corrette: da un massimo di dodici note nell'impostazione più flessibile *no filter* si passa a sette note nel *7 notes*, sei nel *6 notes*, cinque nel *5 notes* e infine all'impostazione più rigida *arpeggio*, la quale permetterà solo l'esecuzione delle note presenti nell'accordo attuale, in genere tre o quattro note a seconda del tipo di accordo suonato dallo strumento di accompagnamento.

Style imposta le regole stilistiche a cui deve rispondere il filtro delle note secondo il genere musicale prescelto dall'utente. Al momento solo Blues è implementato. Esso aggiunge alla scala a 6 note, impostabile mediante il Note Filter, la nota blues, la quale se suonata verrà visualizzata di

colore verde. La scala blues è una particolare scala nata in America che caratterizza il genere musicale Blues: se paragonata alle scale maggiori o minori canoniche, essa possiede una nota dissonante che la rende molto particolare. Prendendo come riferimento la tonica come I grado della scala minore, la nota blues si trova al IV grado aumentato. Ad es: scala di Am, la nota blues è il Eb.

Volume imposta il livello di sensibilità sonora nel riconoscimento delle note suonate dal solista. Più è alto il volume e più il sistema sarà sensibile alle note suonate. Questa feature è stata introdotta in modo da evitare che il sistema fosse troppo sensibile allo strumento in input, ed escludere il rischio di riconoscere note anche quando in realtà non è stata suonata alcuna nota (ad esempio il rumore delle dita del chitarrista quando cambia nota durante l'esecuzione avrebbe potuto determinare un errore nel riconoscimento).

note visualizza il nome della nota attualmente suonata dal solista.

gain indica l'intensità della nota, in altre parole il volume al quale la nota è stata suonata.

clean permette di terminare l'attuale sessione di riconoscimento per iniziarne una nuova, cancellando tutte le note e gli accordi fin qui riconosciuti. Questa funzione è stata inserita al fine di permettere all'utente, qualora volesse intraprendere una nuova sessione di esecuzione musicale, di cancellare ciò che è stato riconosciuto sino a quel momento.

Nella parte destra dell'interfaccia troviamo tre riquadri:

chord comunica l'accordo attualmente suonato dall'accompagnamento. Questo dato coincide con l'accordo rilevato dall'oggeto ChordDetector tramite l'algoritmo di ranking descritto nel capitolo 2.

key indica la tonalità attuale rilevata dal KeyDetector in base agli accordi riconosciuti da ChordDetector.

accordi e tonalità con relativi punteggi vicini a quelli indicati da *chord* e *key*, sono mostrati nel terzo riquadro sottostante ai precedenti a scopo di debugging.

3.1 HyperTeacher

Nella prima schermata, mostrata in Figura 3.1, si trova lo spartito, dove sono presenti in verde le note suonate dal solista e nella parte inferiore gli accordi

suonati dall'accompagnamento. Tale visualizzazione non rispetta le regole della notazione musicale infatti la durata delle note è rappresentata dalla lunghezza dei blocchi colorati. Nel caso la nota riconsciuta sia una delle note alterate, una piccola *b* viene posta all'interno del blocco corrispondente, ad indicare che essa è bemolle. Portando il puntatore sopra ogni blocco, inoltre, verrà visualizzato il nome della nota selezionata.
Con questa visualizzazione, il musicista può iniziare a usare il sistema settando i parametri Note filter, Style e cliccando sul microfono. A questo punto inizia l'analisi delle note del solista e degli accordi dell'accompagnamento, che appariranno sullo spartito una volta riconosciuti. Cliccando nuovamente sul microfono, il sistema si mette in pausa e l'utente ha l'opportunità di rivedere la propria performance utilizzando la barra di scorrimento nella parte inferiore dello spartito.
Sulla base dei parametri impostati mediante i selettori, verranno visualizzate in verde le note corrette ed in rosso le note non corrette.

3.2 ChordGram

In questa sezione verrà descritto il ChordGram (Figura 3.2), cioè il diagramma prodotto dall'analisi spettrale descritta nel capitolo 2 e rappresentato come tasti del pianoforte. Ogni tasto è in realtà una barra il cui riempimento descrive il contributo frequenziale di una data nota nelle varie ottave.
Questa è la parte più importante nella quale si è concentrata la maggior parte dello studio preliminare riguardo l'analisi sonora. Come verrà descritto in maniera più approfondita in seguito, sono stati affrontati diversi aspetti critici, come la rilevazione di un nuovo accordo e la scelta del momento più opportuno per effettuare l'analisi spettrale, nei quali l'interfaccia grafica ha avuto un ruolo fondamentale per l'individuazione e la risoluzione di queste problematiche.

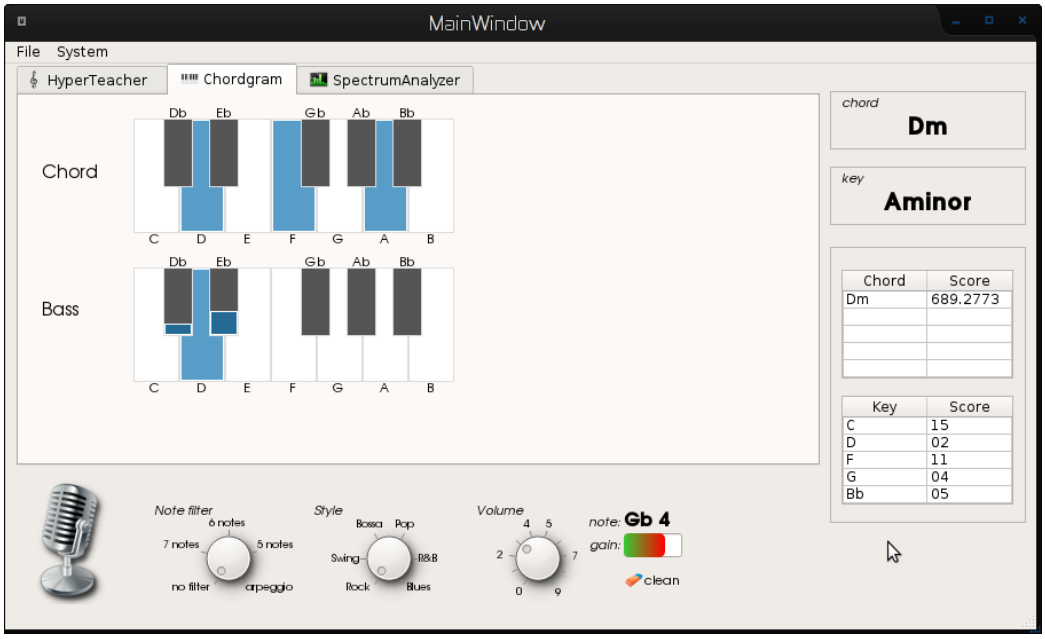

Figura 3.2: Schermata ChordGram

3.2.1 Chord

Le 12 barre superiori rappresentano graficamente il contenuto degli ultimi 12 elementi del vettore *NoteGain*[24] (vedi cap. 2, NoteDetector): più è alto il valore di un elemento, più la barra corrispondente sarà riempita. Nella Figura 3.2 sono state riconosciute le note **D**, **F** e **A** e l'accordo **Dm**, il quale è effettivamente formato da queste tre note.

Il ruolo di questi dodici elementi è fondamentale nel funzionamento del riconoscimento armonico degli accordi. L'algoritmo di ranking sviluppato e descritto nel capitolo 2 si basa principalmente su questo risultato ottenuto dal ChordDetector: il valore rappresentato da ognuna di queste barre contribuirà al calcolo del punteggio di tutti gli accordi. Chi otterrà il punteggio più alto tra questi verrà eletto come l'accordo migliore che più si avvicina all'accordo effettivamente suonato.

3.2.2 Bass

Le dodici barre inferiori invece rappresentano graficamente il contenuto dei primi dodici elementi del vettore $NoteGain[24]$, relativi alle note basse dell'accordo riconosciuto. In questo caso la nota riconosciuta è **D** dato che è l'elemento con il valore più alto tra i dodici.
Il riconoscimento della nota bassa acquista un particolare valore nel momento in cui, durante il calcolo dei punteggi mediante l'algoritmo di ranking, due o più accordi ottengono lo stesso punteggio.
Il punteggio dato dalla nota bassa, dipendendente dal suo contributo in ampiezza, viene sommato agli accordi ex-equo coerentemente con le loro maschere.

3.3 SpectrumAnalyzer

In Figura 3.3 è visualizzato lo spettro in scala logaritmica che rappresenta il contenuto del vettore $notes[128]$, mettendolo in relazione con gli ultimi 12 elementi di $NoteGain[24]$. Durante lo studio in merito al riconoscimento degli accordi, si è presentato il problema di strumenti a corde percosse o pizzicate come il pianoforte e la chitarra, i quali durante l'attacco (parte iniziale dell'evoluzione del suono) presentano del **rumore**, dato dall'impatto del martelletto nel caso del pianoforte e del plettro o delle dita nel caso della chitarra, che rende difficoltoso il riconoscimento.

Ciò che vuole descrivere questa schermata è il momento in cui viene analizzato lo spettro alla ricerca dell'accordo: quando un accordo viene suonato c'è una prima fase molto breve di salita del volume (detto attacco, come si può vedere in Figura 3.5) nella quale è considerevole la presenza del rumore. Una volta raggiunto il picco massimo, il volume diminuisce e il rumore decade rapidamente lasciando spazio alle armoniche che compongono il suono che definisce l'accordo.

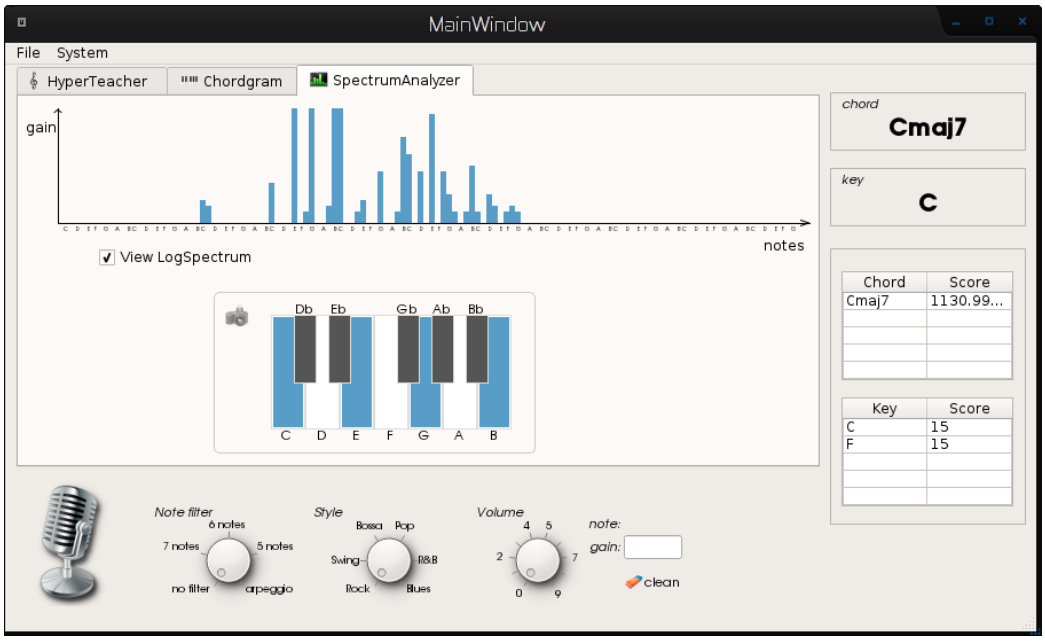

Figura 3.3: Schermata Spectrum Analyser: accordo con rumore prima del riconoscimento

In Figura 3.4 è raffigurato il momento in cui è stato effettuato il riconoscimento (icona della macchina fotografica in alto a sinistra nel riquadro in basso e sfondo grigio scuro), che permette di notare, se confrontato con lo spettro rappresentato in Figura 3.3, come le barre dello spettro siano più delineate rispetto alla figura precedente. Il rumore è notevolmente diminuito lasciando spazio alle armoniche appartenenti all'accordo.

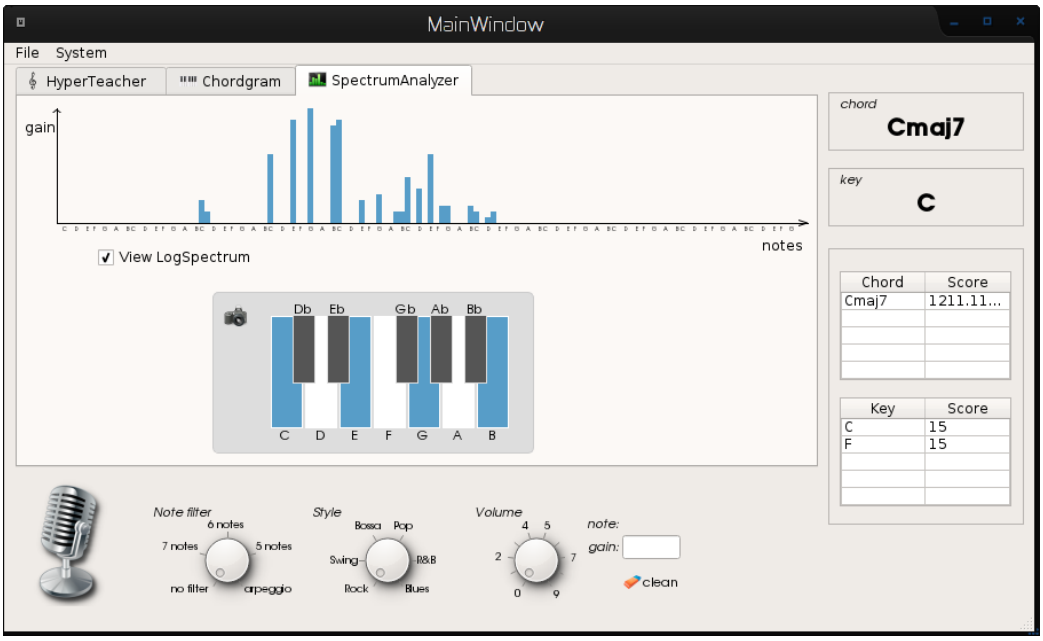

Figura 3.4: Schermata Spectrum Analyser: accordo senza rumore durante il riconoscimento

3.3.1 Considerazioni

Durante i test si è sperimentato che entro un intervallo di tempo di circa 0.1 secondi (cfr. Figura 3.5) nella maggior parte degli strumenti il rumore non è più presente, ed è proprio in questo istante che viene idealmente scattata una fotografia dello spettro ed effettuato il riconoscimento dell'accordo. Per quanto riguarda la rappresentazione di questa fase nell'interfaccia, l'istante durante il quale viene effettuata l'analisi è evidenziato dalla comparsa di un'icona che raffigura una macchina fotografica e dal cambio di colore dello sfondo del ChordGram.

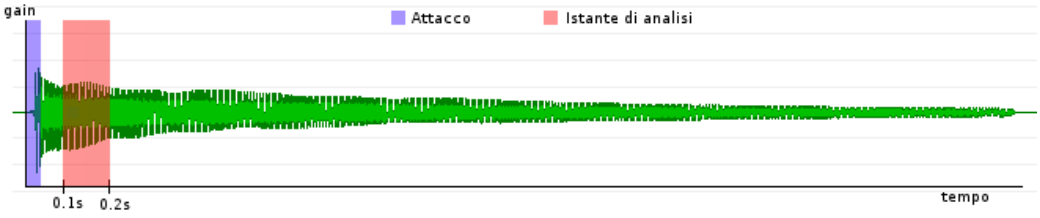

Figura 3.5: Forma d'onda di un accordo di chitarra: in blu è evidenziata la fase iniziale del suono (attacco) e in rosso l'istante in cui viene effettuata l'analisi dell'accordo

In Figura 3.5 è possibile vedere una forma d'onda durante la sua evoluzione. Le sue fasi principali sono 4:

Attacco è la parte iniziale dove il volume del suono sale raggiungendo un picco massimo. L'analisi di questa fase è fondamentale per l'inizio del riconoscimento di un nuovo accordo, infatti, prima di eseguire l'analisi spettrale e il conseguente riconoscimento di accordo e tonalità, il sistema rimane in attesa che il suono raggiunga il suo picco massimo.

Decadimento è la fase in cui il suono, raggiunto il picco massimo, decade in maniera significativa sino a raggiungere un volume stazionario.

Sustain è la fase dove viene mantenuto il volume raggiunto dopo la fase di decadimento. Questa fase in strumenti come pianoforte o chitarra non è presente poiché il suono dopo il picco massimo decade sempre, mentre invece è presente in strumenti come organo, violino e fiati.

Rilascio è l'ultima fase dove, una volta terminata la stimolazione dello strumento da parte del musicista, il suono decade nuovamente sino al completo abbassamento di volume.

Per semplicità nella Figura 3.5 è stata evidenziata, in blu, solo la prima fase, quella dell'attacco, in modo tale da poter rappresentare il momento in cui viene eseguito il riconoscimento, in rosso, la quale risulta nella fase di decadimento. Come si può notare, la fase dell'attacco è molto breve e il picco massimo viene raggiunto quasi istantaneamente, dando inizio alla fase di decadimento. Dopo un intervallo di tempo di 0.1 secondi, in piena fase di decadimento, viene eseguita l'analisi e il riconoscimento dell'accordo.

Capitolo 4

Valutazione sperimentale del sistema

In questo capitolo verrà affrontata la valutazione del riconoscimento degli accordi, del riconoscimento melodico e del riconoscimento della tonalità. Per effettuare la valutazione sono stati effettuati diversi test a seconda del modulo preso in considerazione.
Per il riconoscimento accordi è stato fatto suonare uno stesso accordo (in particolare Am7) da diversi strumenti ed è stato confrontato lo spettro con il chordgram ottenuto e con il risultato calcolato dal ChordDetector. Per il riconoscimento della tonalità è stato selezionato lo strumento più semplice da riconoscere (il vibrafono, poiché possiede uno spettro molto scarno di armoniche) che tramite l'ambiente di test ha eseguito diversi accordi dai quali, una volta riconosciuti esattamente dal ChordDetector, il KeyDetector ha potuto estrapolare la tonalità.

Infine è stato testato anche il riconoscitore melodico, la cui implementa-

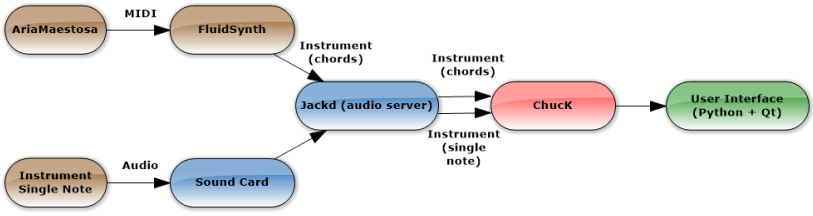

Figura 4.1: Struttura del test

zione si è basata su di un oggetto fornito dal linguaggio di programmazione musicale ChucK, il PitchTracker. Utilizzando l'ingresso audio della scheda audio è stata collegata al sistema una chitarra elettrica che ha eseguito delle

sequenze di note.
In Figura 4.1 è rappresentata la struttura mediante la quale è stato effettuato il test.
Il software cardine senza il quale sarebbe stato molto difficile realizzare il sistema è il server audio **Jackd**, facente parte del progetto opensource **Jack**. Esso fornisce tutto il sistema di backend audio sfruttando il paradigma client-server. Le applicazioni che vogliono usare il server Jackd si connettono ad esso come client, con la possibilità di avere porte di input e porte di output. In questo caso specifico, Jack ha permesso la connessione dei canali di ingresso della scheda audio all'applicazione di riconoscimento melodico sviluppata in linguaggio ChucK. Inoltre è stato possibile avviare il software fluidSynth come client Jack, in modo da poter collegare l'output audio di fluidSynth in ingresso al riconoscitore accordi, sviluppato sempre all'interno del programma scritto in ChucK.

4.1 Valutazione riconoscimento accordi

Per valutare il riconoscimento accordi sono stati fatti dei test con diversi strumenti i quali hanno eseguito gli stessi accordi.
Gli strumenti scelti per il test sono stati:

Vibrafono

Pianoforte

Chitarra acustica

Organo a canne

4.1.1 Vibrafono

Il vibrafono è un particolare strumento a percussione caratterizzato da uno spettro molto povero di armoniche, come si può vedere in Figura 4.2. L'accordo suonato e riconosciuto è stato **Am7**, il quale è formato dalle note **A C E G**. Nel ChordGram riportato in basso nella figura è possibile osservare come le barre corrispondenti a queste note vengano riempite con il contributo di C E G e A.
Prendendo in considerazione lo spettro in alto, si può constatare che le uniche barre presenti nello spettro riportano i nomi delle note rappresentate nel ChordGram. Questa è una particolarità dello spettro del vibrafono, che alle frequenze fondamentali delle note suonate aggiunge soltanto armoniche

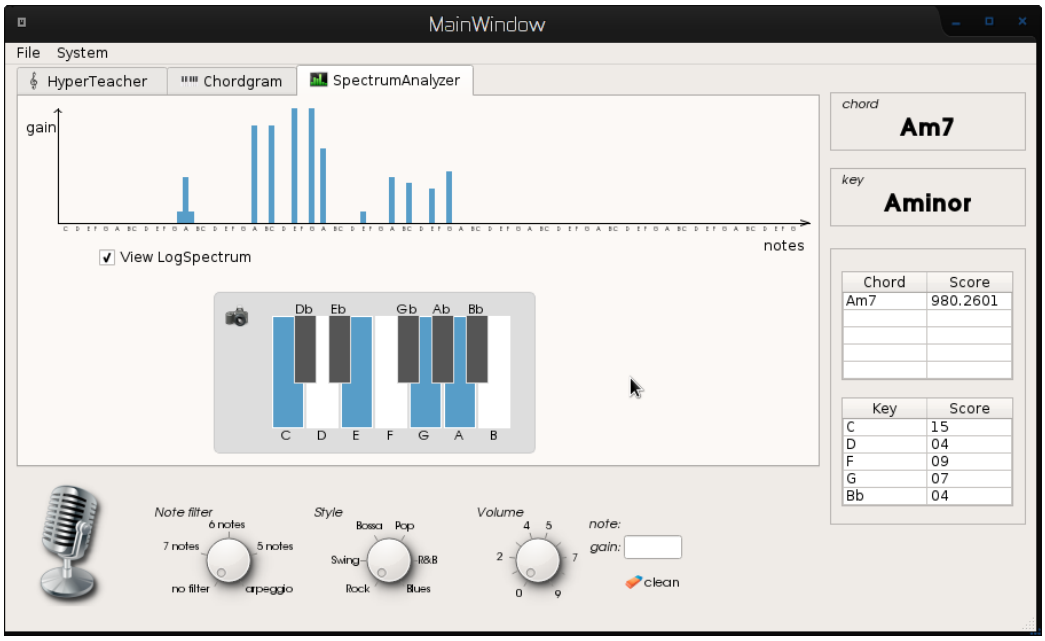

Figura 4.2: Spettro rappresentante l'accordo di Am7 suonato dal vibrafono

distanti un'ottava, rendendo facilitato il rilevamento degli accordi.

4.1.2 Pianoforte

Se si confrontasse con lo stesso accordo questa volta suonato dal pianoforte, mostrato in Figura 4.3, si potrebbe notare come lo spettro diventi più ricco di armoniche. Infatti il timbro del suono del pianoforte è molto diverso da quello del vibrafono e quando lo si ascolta si ha l'impressione di essere molto più "pieno" da un punto di vista auditivo. Inoltre il pianoforte produce il suono tramite dei martelletti che percuotono le corde: alla pressione di un tasto corrisponde il movimento di un martelletto, il quale muovendosi percuote la corda associata. Il suono di tutto il meccanismo necessario alla percussione della corda verrà classificato come rumore ed è presente in maniera considerevole durante l'attacco del suono.

Questa ricchezza dello spettro ha reso più difficoltoso il riconoscimento dell'accordo suonato, ma è proprio grazie allo studio di questo strumento che è stato possibile identificare il problema del rumore durante l'attacco e poter poi scegliere l'istante ideale per poter effettuare l'analisi e il riconoscimento senza ottere risultati inesatti.

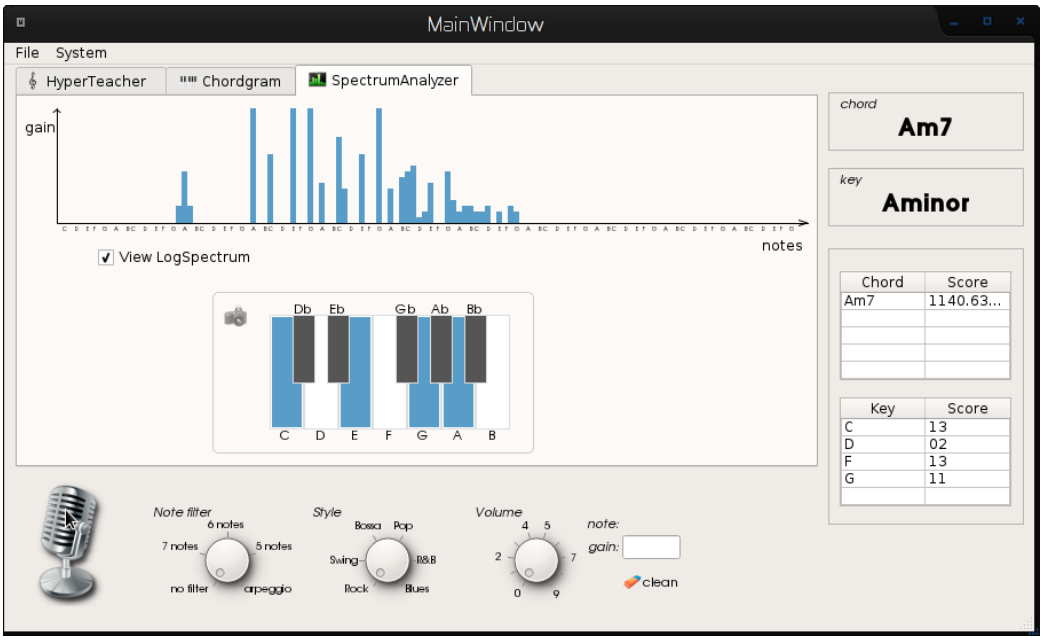

Figura 4.3: Spettro rappresentante l'accordo di Am7 suonato dal pianoforte

4.1.3 Chitarra acustica

In Figura 4.4 è raffigurato lo spettro di una chitarra che esegue il medesimo accordo di pianoforte e vibrafono. Lo spettro della chitarra, come si può notare dalla figura, ha un tipo di spettro ancora diverso. Esso presenta un numero di armoniche superiore rispetto allo spettro del pianoforte, ma che possiedono un contributo frequenziale relativamente inferiore e il sistema di riconoscimento riesce senza difficoltà a identificare l'accordo corretto.
Anche la chitarra acustica, essendo uno strumento a corde pizzicate, presenta del rumore durante la fase dell'attacco dovuta all'impatto di dita o plettro con le corde che il sistema, con gli accorgimenti presi per eliminare il rumore durante l'analisi del pianoforte, riesce ad eliminare in maniera efficiente.

4.1.4 Organo

Uno spettro ancora più complesso è quello raffigurato in Figura 4.5 che rappresenta l'accordo suonato da un organo a canne. Qui si può notare come il sistema abbia riconosciuto un accordo diverso, benchè molto simile, da quello suonato in realtà. Ciò è dovuto al fatto che l'organo introduce nel suo spettro un numero considerevole di armoniche diverse dalle frequenze fondamentali delle note suonate, con un contributo in ampiezza decisamente superiore rispetto agli altri strumenti. Per questo tipo di strumenti occorre un tipo di

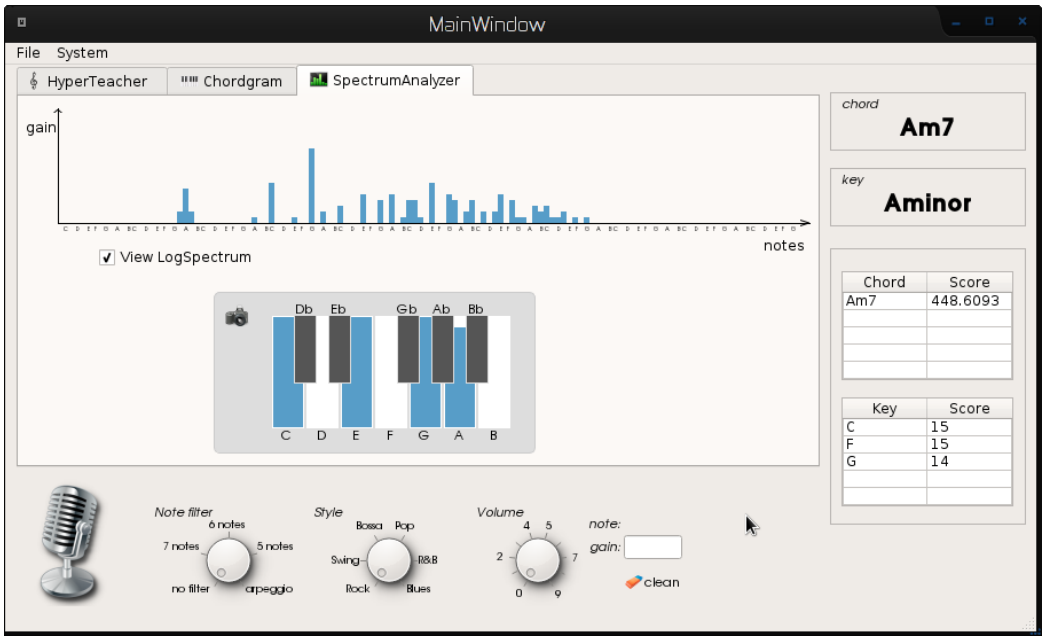

Figura 4.4: Spettro rappresentante l'accordo di Am7 suonato dalla chitarra

approccio diverso non affrontato nell'ambito di questa tesi.

Tutte queste differenze di armoniche sono ciò che caratterizza il cosiddetto timbro dei vari strumenti, cioè quelle particolarità del suono che permettono di identificarli.

4.1.5 Problematiche riscontrate

Rumore bianco

Uno dei problemi più importanti che sono stati riscontrati è stato il rumore bianco durante l'attacco degli strumenti. Il suono di uno strumento affronta diverse fasi: la fase iniziale, durante la quale il volume del suono aumenta, è detta **attacco**. In questa fase gli strumenti come pianoforte e chitarra, presentano del rumore bianco dato dalla percussione delle corde. Questo rumore rende difficoltoso il rilevamento dell'accordo suonato, dato che lo spettro presenta armoniche che non fanno parte dell'accordo. Per ovviare a questo problema, come spiegato anche nel capitolo 2, prima di effettuare l'analisi si attende un intervallo di tempo di circa 0.1 secondi dopo il picco massimo durante il quale il rumore decade, lasciando spazio alle armoniche che compongono l'accordo.

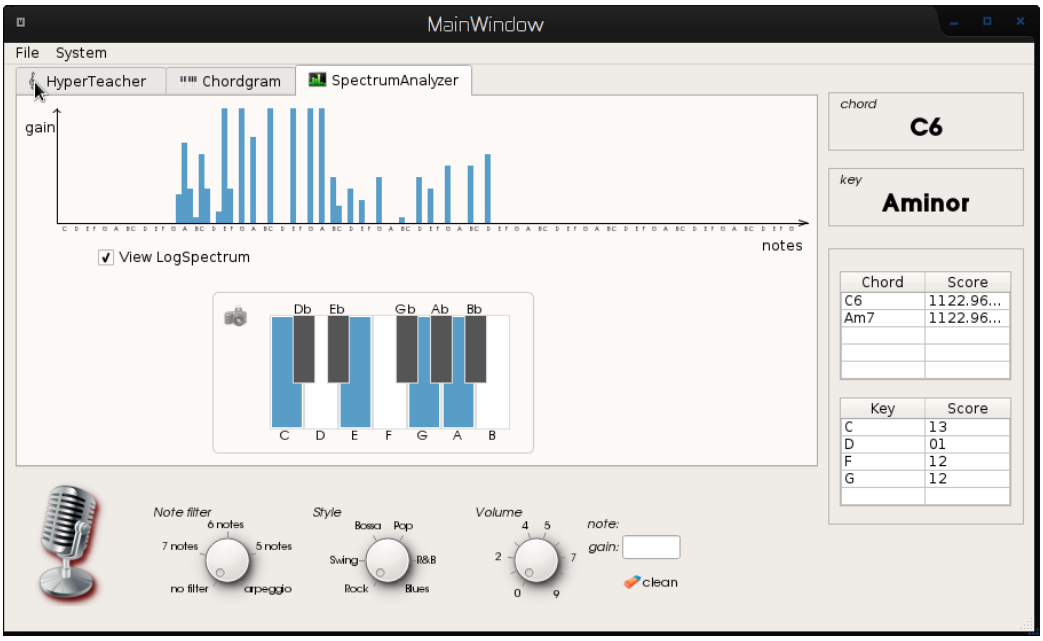

Figura 4.5: Spettro rappresentante l'accordo di Am7 suonato dall'organo a canne

4.2 Valutazione Riconoscimento tonalità

Il test per riconoscere la tonalità si è concentrato più sulla scelta di accordi invece che su strumenti diversi.
Per effettuare il test è stato utilizzato il campionatore opensource **fluidsynth** per generare i suoni e il sequencer MIDI opensource **AriaMaestosa** come input a fluidsynth. In AriaMaestosa è possibile scrivere una partitura musicale, la quale verrà convertita in segnali MIDI che verranno mandati al campionatore, in questo caso fluidsynth.

In Figura 4.6 sono raffigurati gli accordi usati per il test, divisi per battute. Nella battuta 1, è presente l'accordo di Cmaj7, accordo appartenenente alla classe degli accordi maggiori, secondo l'algoritmo verrebbe assegnato un punto alla tonalità di C e un punto a G. Battuta 2 troviamo un accordo sospeso, Csus9, in questo caso nessuna tonalità riceve punti. In battuta 3 c'è il Dm, in 4 il Dm7 e il Dm79, tutti appartenenti alla classe di accordi minori: ricevono un punto la tonalità di C, F e Bb. A questo punto la tonalità con più punti al momento è C. Andando avanti troviamo G75+, Am, Am7, Em. Alla fine dell'esecuzione la tonalità con maggiore punteggio è quella di C, la quale verrà eletta come tonalità corrente.

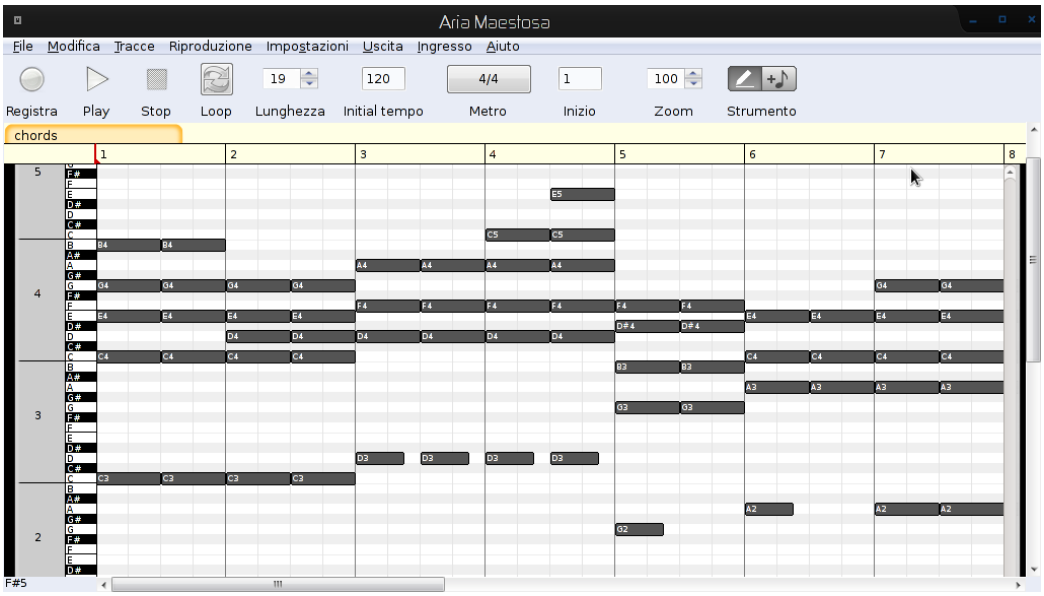

Figura 4.6: Interfaccia di AriaMaestosa

4.3 Valutazione riconoscimento melodico

Per il riconoscimento melodico è stata collegata in ingresso al sistema una chitarra elettrica attraverso l'interfaccia audio di input della scheda audio.
Il riconoscimento melodico è molto attendibile se utilizzato con strumenti come pianoforte, vibrafono e chitarra, riuscendo a riconoscere perfettamente la nota suonata e la sua ottava. In strumenti come l'organo il sistema ha delle difficoltà nel riconoscere l'ottava giusta: questo perché l'organo presenta molte armoniche nelle ottave alte che hanno un contributo frequenziale significativo.
È importante sottolineare inoltre il fatto che un musicista solista, per ragioni che riguardano l'espressività dell'esecuzione, delle volte si ritrova a suonare più note simultaneamente, ciò non è assolutamente previsto nell'algoritmo di riconoscimento e potrebbe essere un interessante sviluppo futuro.

4.4 Risultati ottenuti

Il test riguardante il riconoscitore di accordi usando il pianoforte, il vibrafono e la chitarra ha ottenuto dei risultati soddisfacenti, il 90% degli accordi

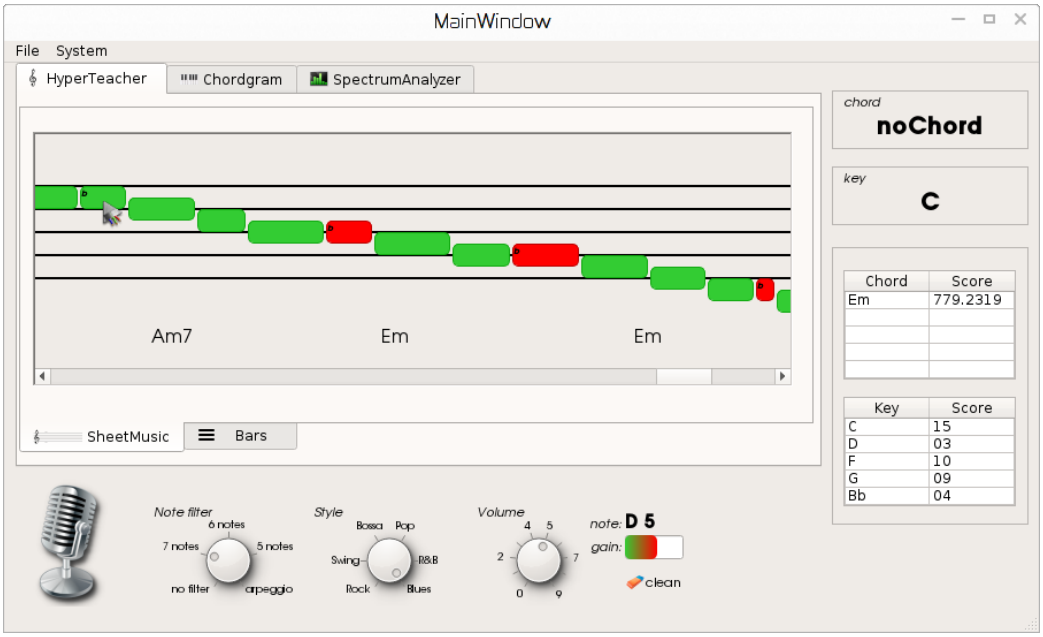

Figura 4.7: Esempio di riconoscimento melodico, in verde le note giuste, in rosso le note fuori scala eseguite dal musicista

suonati sono stati riconosciuti correttamente, mentre nell'organo a canne i risultati sono drasticamente più bassi e gli accordi riconosciuti non sono attendibili.
Per quanto riguarda il riconoscitore della tonalità i risultati sono attendibili a patto che la finestra temporale di analisi degli accordi passati sia impostata adeguatamente. Una finestra troppo piccola causerebbe delle valutazioni erronee poiché verrebbe analizzato un lasso temporale insufficiente con il rischio di instabilità nella tonalità riconosciuta. D'altra parte una finestra temporale troppo estesa potrebbe risultare insensibile ad eventuali cambiamenti di tonalità durante il brano analizzato.
Il riconoscitore melodico è molto efficiente nel riconoscere le note singole. Eventuali esecuzioni comprendenti più note contemporaneamente rendono il riconoscimento non attendibile.

Capitolo 5

Conclusioni

Alla luce di quanto finora dimostrato si può riassumere pertanto che, rispetto agli obiettivi che ci si era prefissati circa lo sviluppo di uno strumento di supporto all'improvvisazione musicale si è raggiunto un risultato soddisfacente sebbene ulteriormente perfezionabile. Ci si è soffermati in particolar modo sull'aspetto riguardante il riconoscimento degli accordi e la loro qualificazione all'interno della tonalità in modo da creare un fondamento solido alle applicazioni che ne fanno uso. Sarebbe degno di ampliamento il sistema vero e proprio di supporto all'improvvisazione tramite: lo sviluppo di ulteriori stili musicali selezionabili dall'utente (ad esempio Jazz, Rock etc.), la possibilità di attivare la correzione delle note sbagliate oltre alla loro mera segnalazione, la visualizzazione delle note suggerite appartenenti alla scala musicale relativa alla tonalità rilevata durante l'esecuzione. Si auspica dunque la realizzazione di uno strumento più completo in grado di soddisfare tutte le esigenze compositive del musicista con particolare attenzione alla fase dell'improvvisazione.

5.1 Possibili sviluppi futuri

A seguito dei vari test di valutazione svolti è emerso che uno dei principali problemi del sistema consiste nella difficoltà di riconoscere gli accordi eseguiti da determinati strumenti musicali, pertanto per far fronte a tale imprecisione il sistema potrebbe essere perfezionato mediante l'uso di reti neurali.
Grazie al sistema realizzato in questa tesi e mediante l'implementazione di funzionalità specifiche, gli Iper-strumenti come le interfacce gestuali, che ancora sono lontani dall'essere affidabili e precisi, potranno godere di un supporto notevole in quanto a precisione e usabilità.
Come funzionalità aggiuntive una possibile estensione potrebbe essere la pos-

sibilità di sfruttare le informazioni raccolte dal riconoscitore di tonalità al fine di armonizzare la parte eseguita dal musicista, arricchendo la melodia suonata con ulteriori note coerenti prodotte dal sistema.
Inoltre potrebbe essere utile il salvataggio della partitura ottenuta durante una sessione di utilizzo in formato MIDI, in modo da poter essere importata anche in altri software di produzione musicale.

Bibliografia

[Leonello Tarabella, 2010] Tarabella, L., *Scolpire il suono*

[Matthias Mauch, 2010] Mauch, M., *Chordino*,

[Wang Ge, 2008] Wang, G., *The ChucK Audio Programming Language*,