INFORMATICA MUSICALE

UNIVERSITÀ DEGLI STUDI DI CATANIA
DIPARTIMENTO DI MATEMATICA E INFORMATICA
LAUREA TRIENNALE IN INFORMATICA
A.A. 2018/19
Prof. Filippo L.M. Milotta

ID PROGETTO: 04

TITOLO PROGETTO: Simple Sound Energy

AUTORE 1: Cosentino Enrico

Indice

1. Obiettivi del progetto

Per poter parlare di rilevamento di battiti è innanzitutto necessario definire matematicamente cos'è un battito, dopodiché bisogna decidere che tipo di informazione sui battiti si vuole che l'algoritmo restituisca, l'obiettivo principale di questo progetto è quello di introdurre la beat detection, pertanto mi sono limitato a farmi restituire numero e posizione dei battiti.

Su stream audio il discorso si complica leggermente perché non abbiamo a disposizione tutti i campioni della traccia in ogni momento, ma solo un loro sottoinsieme, inoltre non abbiamo garanzie che le finestre di campioni disponibili in due momenti successivi non presentino delle sovrapposizioni, per cui il calcolo di posizione e numero diventa impreciso. Per questo ho deciso di effettuare un semplice confronto visivo tra la mia implementazione dell'algoritmo e una di libreria trovata online.

2. Metodo Proposto / Riferimenti Bibliografici

Un battito è una forte variazione di intensità improvvisa all'interno di una traccia, per poter rilevare un battito dobbiamo quindi confrontare l'energia sonora della traccia in un dato istante con quella di un suo intorno. Se consideriamo un istante come 1024 campioni possiamo usare una finestra di 44032 campioni (circa un secondo di musica) come intorno o località.

L'algoritmo di rilevazione funziona così:

1. Si calcola l'energia istantanea ***e*** su 1024 campioni, questa non è altro che la sommatoria dei quadrati dei valori che assumono i campioni nei vari canali audio, nel caso di tracce stereo la formula è:

$$e = e_{stereo} = e_{right} + e_{left} = \sum_{k=i_0}^{i_0+1024} a[k]^2 + b[k]^2$$

2. Si calcola l'energia locale ***E*** su 44032 campioni sfruttando un opportuno array ***H[]*** contenente le ultime 43 energie istantanee calcolate (1024*43 = 44032), in questo caso l'energia è la media delle energie archiviate in ***H[]***

$$< E >= \frac{1}{43} \times \sum_{i=0}^{43} (E[i])^2$$

3. Si calcola la varianza ***V*** del contenuto di ***H[]***

$$V = \frac{1}{43} \times \sum_{i=0}^{43} (E[i] - < E >)^2$$

4. Si calcola la costante ***C*** di sensibilità dell'algoritmo con una formula di regressione lineare

$$C = (-0.0025714 \times V) + 1.5142857$$

5. Si shifta ***H[]*** di 1 a destra e si inserisce ***e*** in ***H[0]***

6. Si confronta ***e*** con ***C * E***, se ***e*** è maggiore abbiamo un battito, altrimenti no

Per implementare l'algoritmo ho usato il linguaggio di programmazione Processing e in particolare la libreria audio Minim, che consente di aprire file audio sia interamente in memoria che come stream. La classe MyBeatDetect che ho creato implementa l'interfaccia AudioListener tramite cui gli oggetti audio possono passare i loro campioni alla classe e si compone principalmente di due metodi: detect() e analyse(), il primo metodo va usato per gli stream audio e la chiamata va inserita all'interno del draw() (viene quindi chiamato una volta per ogni frame), mentre il secondo va usato su tracce audio interamente in memoria dato che analizza l'intera traccia con una chiamata e restituisce un array contenente la posizione dei beat nonché, indirettamente, il loro numero; l'array ha dimensione pari al numero di campioni della traccia originale diviso 1024, ogni posizione dell'array corrisponde quindi a 1024 campioni e contiene 1 se i campioni corrispondente rappresentano un battito e 0 altrimenti, il numero di battiti è quindi il numero di 1 all'interno dell'array.

Come riferimento per il progetto ho usato questa pagina web:

http://archive.gamedev.net/archive/reference/programming/features/beatdetection/default.html

3. Risultati Ottenuti

Per poter interpretare la qualità dei risultati ottenuti dal metodo analyse() è necessario utilizzare una traccia di cui conosciamo a priori numero e posizione dei battiti, per questo ho usato un treno di impulsi di durata e periodo noti, generato da linea di comando usando il programma "sox", con la sintassi "*sox -b 16 -Dr 44100 -n impulseTrain.wav synth 1s square pad 8267s repeat 53*", l'array dei risultati attesti è stato ottenuto con una riduzione del treno di impulsi a intervalli di 1024 campioni, confrontando questo array con quello ottenuto dall'output di analyse() si hanno esattamente 0 differenze, quindi l'algoritmo funziona in modo ottimale su tracce con battiti molto ben definiti.

Per quanto riguarda il metodo detect() ho potuto sperimentare più facilmente dato che è possibile applicarlo a musica reale, confrontando le due implementazioni si nota che la mia è estremamente sensibile alle percussioni, rilevando forse anche più battiti di quelli presenti, mentre non rileva gli altri strumenti, ciò è probabilmente dovuto alla natura delle percussioni: dato che queste non emettono suoni continui la loro presenza introduce necessariamente una variazione di energia, mentre i battiti degli altri strumenti sono dovuti a variazioni di frequenza, che non vengono rilevate da questo algoritmo. La funzione di libreria invece rileva anche gli altri strumenti ma è meno precisa sulle percussioni.