Simple Sound Energy Beat Detection

Enrico Cosentino

Progetti
Informatica Musicale 2018/19

Indice

- Inserite un elenco puntato degli argomenti trattati in questo progetto, come se li doveste presentare in 10 slide, non una di più, non una di meno.
- Considerate di dover presentare il vostro lavoro in 15-20 minuti al massimo, con 10 slide
  - Vi verrà chiesto qualcosa di simile anche per la tesi di laurea, ma lì avrete 5 minuti. Consideratelo un esercizio
- Basate i contenuti di queste slide sulla documentazione dello Step 2

Indice

- Cos’è la beat detection?
- Rilevamento su file audio
- Rilevamento su stream audio
- Algoritmo di rilevamento parte 1
- Algoritmo di rilevamento parte 2
- Algoritmo di rilevamento parte 3
- Dettagli sull’implementazione
- Risultati file audio
- Risultati stream audio 1
- Risultati stream audio 2

Cos'è la beat detection?

- È il rilevamento dei battiti di una traccia audio
- Un battito è un'improvvisa variazione di energia
- Ha applicazioni nello sviluppo di giochi digitali o di lettori multimediali

Rilevamento su file audio

- Si consuma molta memoria per tenere il file aperto
- Possediamo tutta l'informazione relativa alla traccia
- Possiamo ottenere in output vari parametri
- In questo caso l'output sarà numero e posizione dei battiti

Rilevamento su stream audio

- Possediamo solo alcuni campioni alla volta
- Alcuni campioni potrebbero venirci passati più volte
- Non possiamo effettuare analisi approfondite
- Confrontiamo allora la mia implementazione con una di libreria in maniera visiva

Algoritmo di rilevamento parte 1

- Si calcola l'energia istantanea ***e***

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

- Si calcola l'energia locale ***E*** sfruttando un array ***H[]*** contenente lo storico delle energie locali

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

Algoritmo di rilevamento parte 2

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

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

- Si calcola la costante di sensibilità ***c***

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

Algoritmo di rilevamento parte 3

- Si shifta ***H[]*** di una posizione a destra e si inserisce ***e*** in ***H[0]***
- Si confronta ***e*** con ***c*E***
- Se ***e*** è maggiore abbiamo un battito, altrimenti no

Dettagli dell'implementazione

- Linguaggio di programmazione Processing
- Libreria audio Minim
- Classe MyBeatDetect implementa l'interfaccia AudioListener
- Metodi analyse() per file audio e detect() per stream audio
- Minim possiede una sua classe BeatDetect

Risultati file audio

- Ci serve una traccia con numero e posizione dei battiti noti
- Un treno di impulsi generato parametricamente fa al caso nostro
- Il confronto tra l'output dell'algoritmo e l'array delle posizioni dei battiti del treno di impulsi segnala 0 differenze
- Su tracce con battiti molto ben definiti l'algoritmo funziona in maniera ottimale

Risultati stream audio 1

- Possiamo applicare l'algoritmo su qualunque traccia musicale e verificare visivamente il risultato
- La mia implementazione è estremamente sensibile alle percussioni e ignora il resto
- Quella di libreria è sensibile anche ad altri strumenti ma meno precisa sulle percussioni
- Perché?

Risultati stream audio 2

- Probabilmente perché le percussioni non emettono suoni lunghi, per cui ogni nota introduce nuova energia
- Gli altri strumenti invece cambiando nota non modificano l’energia della traccia, ma la frequenza, che questo algoritmo non rileva

Conclusioni

L’algoritmo presentato è il modo più elementare di effettuare beat detection, e i risultati dimostrano che funziona bene solo su tracce audio altrettanto elementari, rimane comunque un punto di partenza importante per comprendere e affrontare il problema del rilevamento dei battiti.

GRAZIE PER L'ATTENZIONE