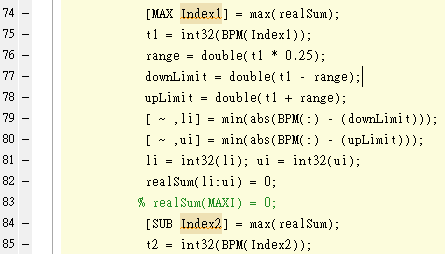
Music Information Retrieval HW4 Report

102062209 邱政凱

這次的作業要我們實作的是Tempo estimation，要從Ballroom Dataset之中計算不同種類舞曲的tempo和beats.

Q1.

第一題的部份是要我們用Fourier tempogram的方式把Dataset中音樂的tempo算出來，並且分別去計算P Score和ALOTC Score。實作的部分我使用的是matlab的tempogram toolbox。步驟上首先就是用spectral flux的方式計算出音樂能量的novelty curve，並且對此novelty curve再做一次STFT以得到所謂的Fourier Tempogram。接著，題目要求我們在此tempogram上面找出兩個predominant tempo。我使用的方法是：用abs(tempogram) 取出tempogram的絕對強度，接著用sum(tempogram,2)的方式把所有時間點上對應的BPM加總，接著用max取出這個加總的向量中強度最大的tempo。第二個tempo我的計算方式是把第一個算出來的tempo，其上下25%的tempogram都歸零之後在此次用同上的方法取出強度最大的tempo。這麼做的原因是因為如果不歸零就取對應次大能量的BPM的話，基本上只會取到跟最大能量正負一個index的BPM而已。

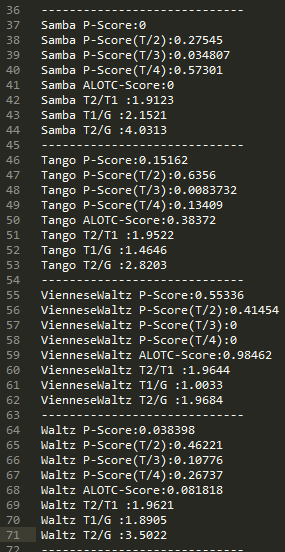
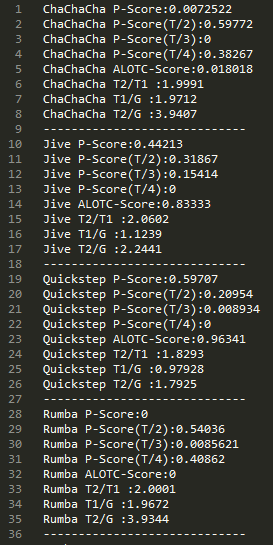


Code的部分大致上就如上，不過這邊有個重點是，最後從sum vector取出來的只是tempo的index，還必須從原本在計算Fourier Tempogram時取出來的row index mapping回到BPM的Vector中取出對應的BPM才行。

首先用這兩個tempo計算estimation的P Score，兩個tempo的權重分別對應各自在tempogram上的強度，估算的Tempo如果介於正確答案的8%以內則依照權重計分，而ALOTC則只需兩個tempo有任一個落在8%以內即得一分。

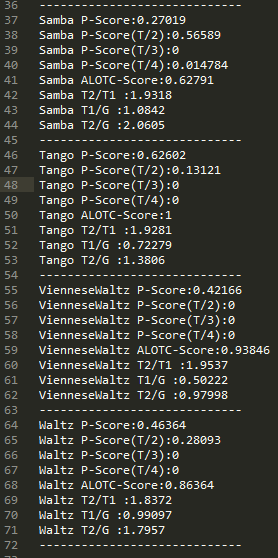
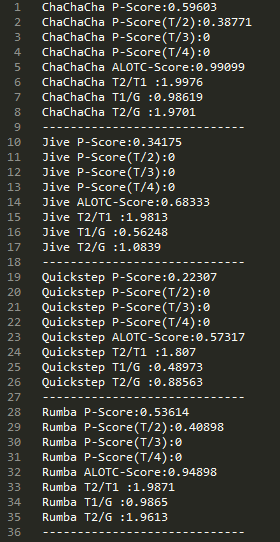
依照每個genre，在跑每首音樂的時候就把分數累加，算完一整個genre之後便除以該genre內的音樂數以求得平均分數。(Rumba的部分我把三種Rumba都合到同一個genre來計算)

這一題其實一開始讓我有點小挫折，因為算出來的P Score非常非常非常的不理想，Cha Cha Cha、Rumba、Samba、Waltz，總共有一半(四種)曲風的P Score是***0***分，也就是沒有任何一首的tempo估測正確的。以下是輸出的結果：



**Figure(1) First run of Fourier Tempogram algorithm**

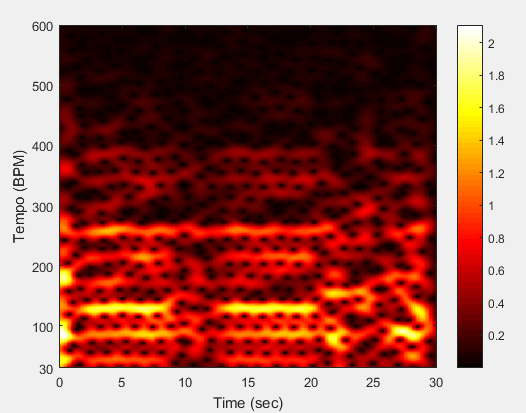
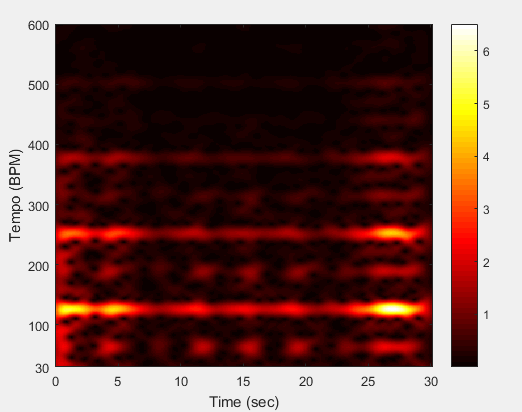
可以看到八個genres裡面有四個genre幾乎是完全沒估測對。納悶了很久，赫然發現matlab的tempogram toolbox中，在運算時是以22050 Fs為基準在運算的，而Ballroom Dataset原本的Sampling Frequency是44100。我認為也許是因為這樣的關係，造成了toolbox中使用的feature rate(novelty curve的sampling rate)因此多了一倍(當然也有可能不是因為這個原因，不過還是基於好奇心做了別的實驗)。於是我把計算Fourier Tempogram時使用的Feature rate除以2，發現計算出來的結果比一開始還要好上許多。以下是把Feature rate除以2之後再照著上面的步驟跑一次之後的結果：



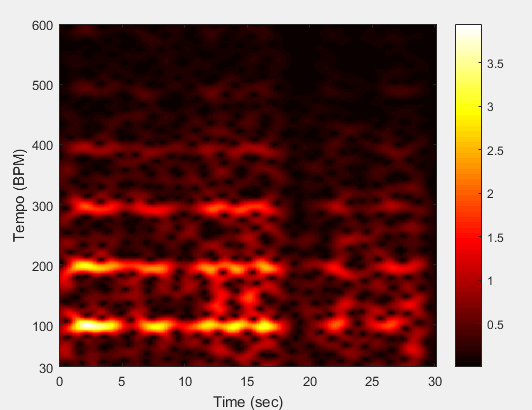
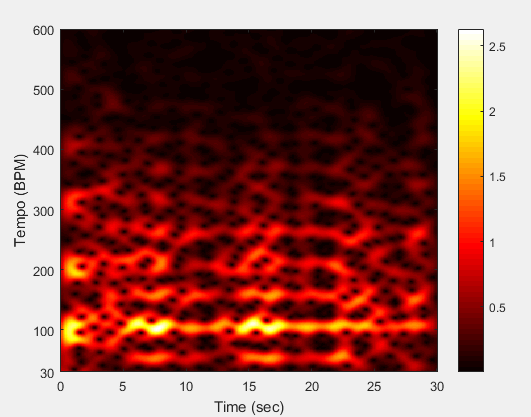
**Figure(2) Outcome of Fourier tempo estimation using half feature rate**

可以很明確的發現，整體的estimation表現比前面還有好上許多，除了Jive / Quickstep / Viennese Waltz三個genre的表現跟前面相比有些許下降外，其他genre的表現都來到正常的範圍了。儘管可能這麼做有點改變了Fourier tempogram estimation的物理意義，但為求較好的estimation結果，接下來幾題我用Fourier tempogram計算估測分數時都會使用一半的feature rate。(比較時我會同時比較修改feature rate前後的) 。 不過不論如何，因為我的算法的關係，我抓取的T1跟T2勢必會有一段距離，所以用P Score來算時基本上一定只會有一個tempo是正確的，因此絕對會比ALOTC還要來的低許多。

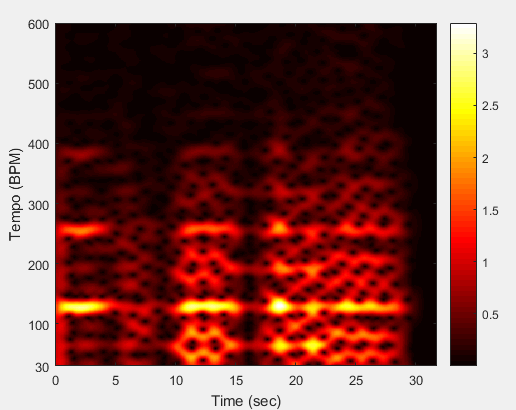
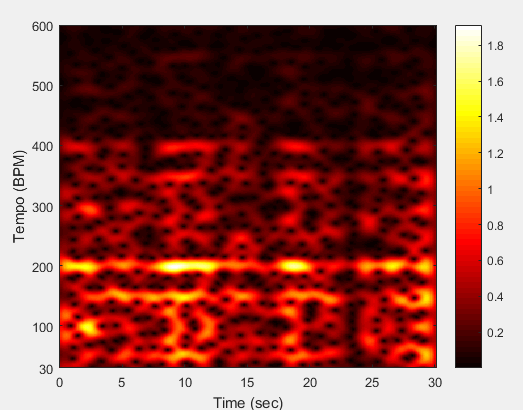
在這邊附上不同種類音樂CLIP的範例tempogram做個簡單的比較：



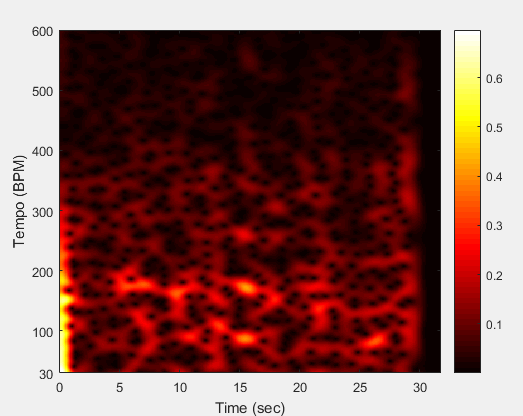
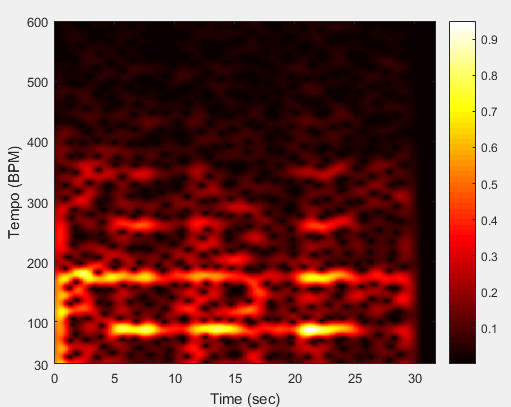
ChaChaCha Jive



Quickstep Rumba



Samba Tango



Viennese Waltz Waltz

可以發現，ChaChaCha和Rumba的tempogram是裡面相對乾淨的(也就是說整個clip的tempo是穩定的)，造成了較高的P Score。而Jive、Quickstep、Samba的tempogram則顯得較為雜亂，可見舞曲的tempo可能因為較複雜的配樂而顯得不穩定，而導致較低的P Score。Viennese Waltz和Tango的tempogram雖然稱不上乾淨，但是大致還是整齊的，P Score介於中間。Waltz則因為整個tempogram強度都偏弱，所以反而正確的tempo附近的強度容易凸顯出來，因此P Score也不至於太低。

Q2 、

這部分的結果我已經有同時印在Figure(1)跟Figure(2)之中了。因為T2跟T1兩個predominant tempo有指定順序(T2較快T1較慢)，我們可以看到，其實不論哪種音樂genre，T2/T1幾乎都是等於2，也就是說最強的兩個tempo之間幾乎就是單純的兩倍數的的關係。這樣的結果其實完全在意料之中，就跟Pitch detection會有octave equivalence一樣，用STFT計算Novelty curve的tempogram也很自然會有這種harmonics / sub harmonics的問題。不過像是tempogram較雜亂的Quickstep或Waltz，比例就會偏離2稍微多一點。

T1/G跟T2/G就比較有意思了，從這兩個數值我們可以看到每個genre的tempogram的不同特性。ChaChaCha、Rumba、Samba、Waltz是T1跟G比較接近，而Jive、Quickstep、Viennese Waltz則是T2跟G比較接近 (在沒有改變feature rate的狀況下是：前者T1、T2分別是G的2、4倍，而後者的T1則吻合正確的G，T2是G的兩倍)。Tango則是比較沒有明顯的趨勢。

經觀察可以發現，ChaChaCha、Rumba、Samba、Waltz四種樂曲的Tempo都偏低，而Jive、Quickstep、Viennese Waltz則是Tempo都偏高。

稍微去聽一下各個樂種的曲子，我覺得解釋方式也許是：容易被判快的樂種，他的背景配樂經常會有速度較快打擊樂、弦樂等伴奏，速度比實際舞曲的Tempo還快，因此導致容易被判快。而容易被判慢 (其實在沒有改變feature rate的狀況下，這些樂種的T1是跟正確Tempo吻合的) 的樂種，因為配樂幾乎就是在舞曲的拍點上進行，因此不會被判快。Tango則比較特別，有時配樂會比較快，有時又會在舞曲的拍點上，因此比較難以歸類到這兩類中。

Q3、

這部分的結果我也已經有同時印在Figure(1)跟Figure(2)之中了。

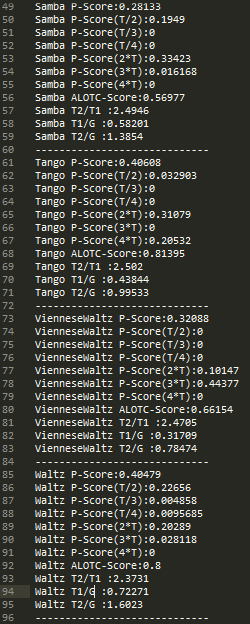
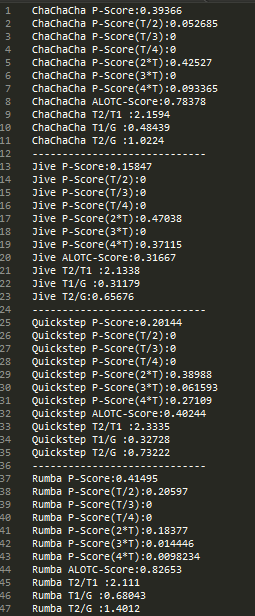
這部分，用未修改Feature rate時的結果來解釋會較為清楚其物理意義。就如同上一題所說，ChaChaCha、Rumba、Samba、Waltz (同時也是在未修改feature rate時P Score都是0的樂種) 容易把tempo判快，T1、T2會變成G的2、4倍，因此T/2和T/4 在這些樂種裡面算出來的P Score都會特別好，而T/2還會大於T/4。Tango在這邊也是T/2的效果最好。

而剩下的Jive、Quickstep、Viennese Waltz就如同前面所說，T1大致對應G，T2大致對應2\*G，因此T/2在這些樂種上算出來的P Score勉強還不會掉到太低，不過T/4就全部都是0了)。T/3的部分，除了Jive和Waltz以外其他P Score幾乎都是0，至於Jive和Waltz相較於其他樂種不一樣的地方也許是他們的tempogram比較沒有那麼整齊的長條狀能量分布而是比較鬆散的分布，因此就算不是sub harmonic，抓出來的tempo也許還是能碰巧落在G的正負8%以內吧。

Q4、

這題是要用ACF(Autocorrelation Function)重作第一、二題。

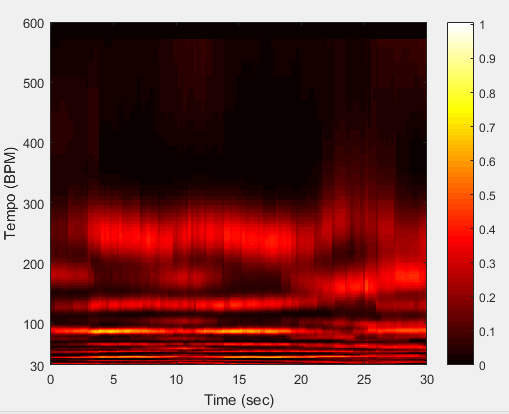
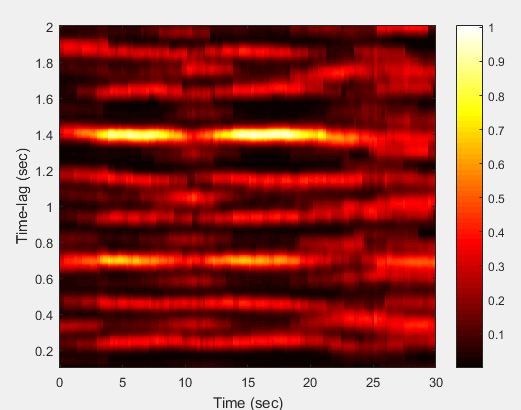
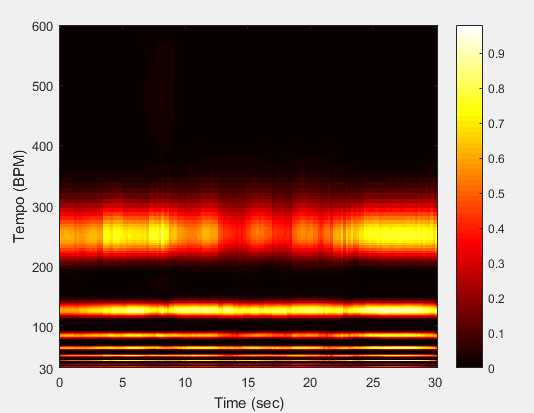
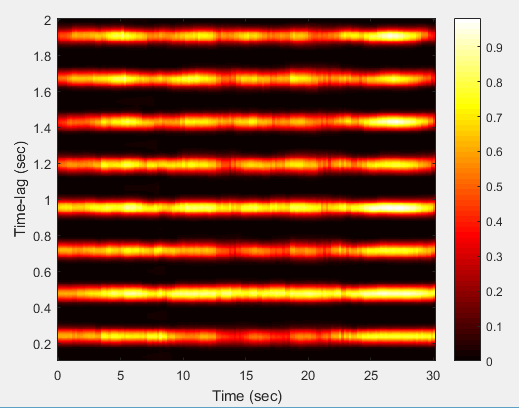
實作的部分，基本上跟第一題大致一樣，不過一開始算出來的ACF tempogram是time-lag的，所以必須把y軸rescale轉換成BPM。另外值得一提的是，用ACF實作的話不需修改Feature rate效果才會比較ok，因此ACF的部分我就沒有去動feature rate了。以下是實驗跑出來的結果：



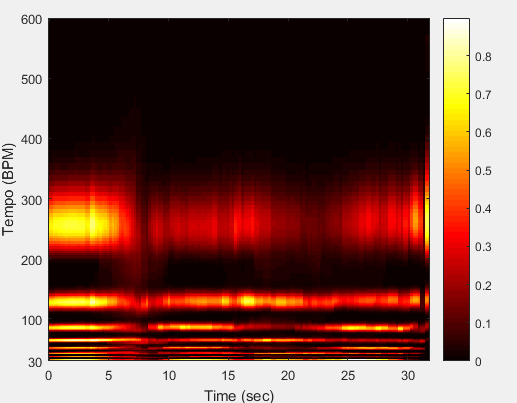
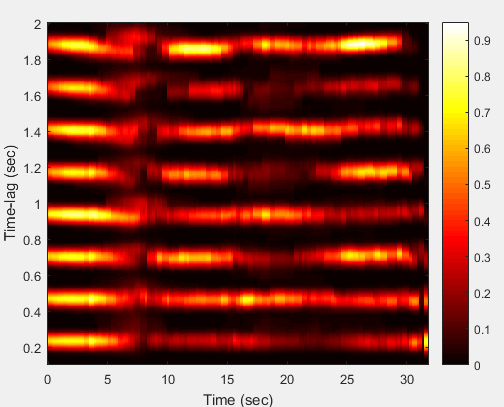
**Figure(3) Outcome of ACF tempogram**

然後以下這邊則選取幾個樂種展示跑出來的ACF Tempogram，左邊是time-lag scale，右邊則是time-BPM。從Time-lag acf tempogram中不難看出來，不同於Fourier Tempogram的是，他的能量強度在lag軸上是循環的，不像Fourier tempogram或是轉換成time-BPM scale之後的tempogram。而在Time-BPM scale中不像是Fourier tempogram會算出很多harmonics，ACF tempogram容易算出來的是sub harmonics。這其實不難解釋，看ChaChaCha左側的time-lag tempogram就知道，0.25 sec的Lag也可以解釋其實是0.5 sec、0.75sec的Lag，Lag越久BPM自然越低。

ChaChaCha



Jive



Tango

在跑出來的estimation結果方面，可以發現其實整體結果都比Figure(2)來的稍差。有趣的是，在Fourier Tempogram(未修改Feature rate)中P Score較低的樂種，也就是容易被算快的較慢的樂種，在ACF Tempogram中反而表現較好，而在Fourier Tempogram(未修改Feature rate)中P Score較高的樂種，也就是G較高的樂種，在ACF Tempogram中反而表現較差。(ACF Tempogram的表現比較近似於我修改過feature rate之後的Fourier tempogram)。

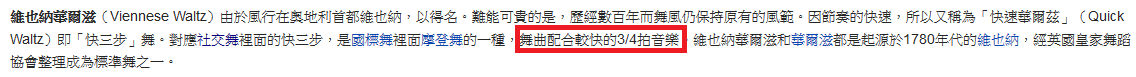
這可以很簡單的解釋為Fourier tempogram傾向於找出harmonics，所以對較慢的樂曲不利，而ACF傾向於找出Sub harmonics，因此對較快的樂曲不利。

T1/T2的部分，跟Fourier tempogram相似，大部分的樂種都是2左右，不過有幾個樂種會逼近2.5。T1/G跟T2/G的部分，可以發現除了ChaChaCha跟Tango以外幾乎沒有樂種的T1或T2的平均是穩定落在G的附近的，也就是說正確的tempo有時是T1算出來的，有時是T2算出來的，這點跟Fourier tempogram時不太一樣，大部分的樂種的正確的 G的估測幾乎都是由T1或T2一方獨佔。

Q5、

這部分的結果我也直接印在Figure(3)了。可以很明顯地看到，不同於Foureir Tempogram(未修改Feature rate的狀況)時，很多樂種較好的P Score都是在T/2或T/4時得出來的，ACF Tempogram傾向於在T\*2(Viennese Waltz是T\*3)時有較好的P Score表現。

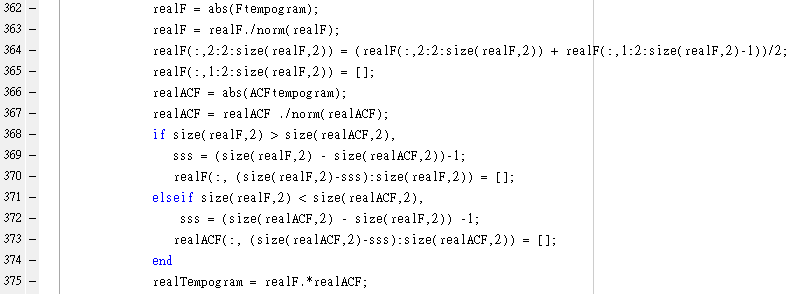
這部分的解釋其實就跟上面那題一樣，ACF Tempogram傾向於找出Sub harmonics，因此傾向於把tempo算的太慢，因此把估測的tempo乘上倍數可以修正這樣的誤差。(相對於Fourier tempogram就是容易算出harmonics因此會把tempo算的太快需要簡單除2/4來修正誤差)。而Viennese Waltz是用3\*T應該是因為他是3/4拍的曲風吧！



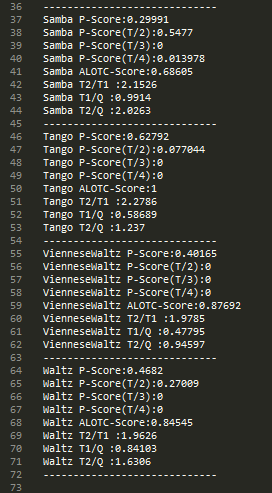
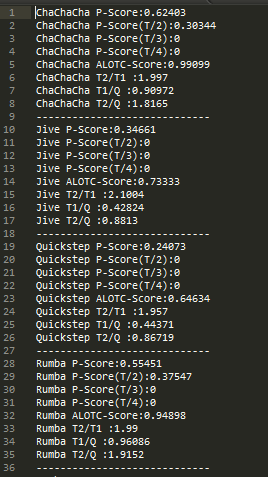
(取自維基百科)

Q6、

這題則是要我們想一個可以改進P Score的演算法。我試驗了兩種方式，第一種是在每次算tempogram時，都同時算出Fourier tempogram(feature rate除以二計算出tempogram之後修正column數)和Autocorrelation tempogram，然後把兩者做點積。Code大致上如下：

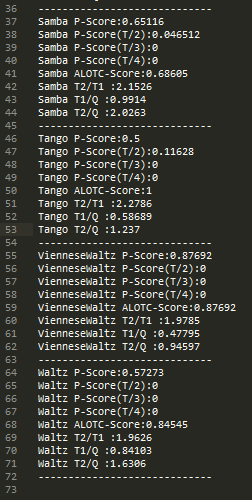
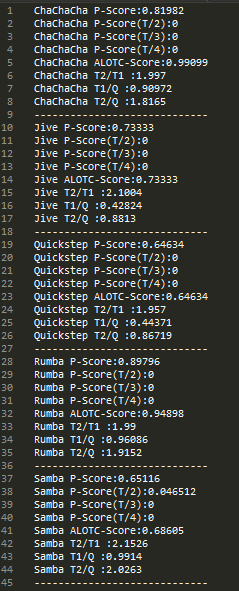


這樣子結合Fourier跟ACF，我原本的預想是因為一個傾向算出harmonics，一個傾向算出sub harmonics，那麼兩者的交集應該就會是正確的tempo G了，所以採用點積的方式，不過結果似乎不盡理想，大致如下：



可以看到整體的P Score是有”微幅”的上升，不過基本上是不能說有明顯的改善的。原因的部分我在想會不會是跟Normalization有關，不過似乎把兩個tempogram都normalize並不會有改善。

另一個方法則有點算是利用了對於樂種的先備知識。首先我們觀察Fiugre(2)，也就是把Feature rate除以2之後計算Fourier tempogram的outcome。我們這部分利用的就是它大部分的樂種幾乎都是T1和T2某個estimation佔了全部的正確估測值，因此，觀察不同樂種T1/G和T2/G哪個是相當接近於1.0的，我們在estimate P Score時使用的權重S1和S2，就直接把對應該樂種的權重設成1，另一個設成0(例如ChaChaCha因為T1比較接近，那ChaChaCha的S1就是1，S2就是0；Jive則是反過來)。結果大致如下：

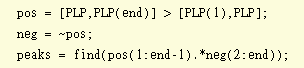


可以看到，整體的P Score表現都有了驚人的提升。不過這種方法如果沒有先對樂種經過預處理得知其特性的話是不能使用的，所以雖然效果很好不過實際可以應用的範圍應該有限。

Bonus、

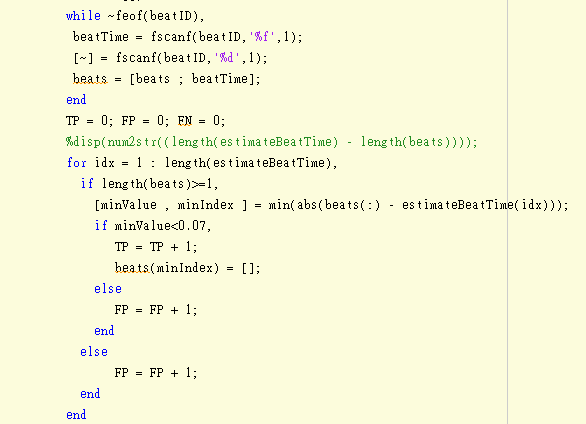
Bonus的部分做的是Beat tracking，並且計算不同樂種tracking的F Score。跟tempo estimation一樣，這邊我使用的也是tempogram toolbox，並用PLP算法來進行。不同於tempo estimation算的是novelty curve的frequency，PLP算的是novelty curve的Phase。

PLP的計算方式是先算出tempogram，接著把tempogram的值轉換成phase，並且計算某時間點對應tempogram上最大強度的tempo在該時間點上應該會出現的phase，並且跟前面直接從tempogram上算出來的phase做比較，若吻合則PLP curve的值就會較高。算出PLP curve之後，找出所有curve上的local maximum，

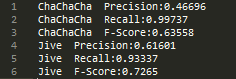


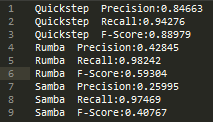
把這些點根據feature rate和sampling frequency轉換成對應的時間，並且跟annotation上的時間點做比對，若落於正負0.07秒以內則算做正確的beat。

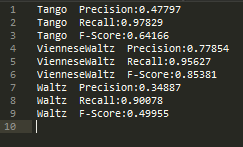
我的作法是先把annotation的值都抓進list裡，接著把PLP算出來的beat點也放進一個list裡，接著遍歷PLP算出來的beat list，一旦annotation裡有對應的(+-0.07 sec)beat，就把它從annotation中消除，並且TP += 1，如果沒有，則FP +=1，遍歷完之後annotation中若還有剩下來的beat，則算做FN。Code如下。



而做出來的結果大致如下：







其實可以看出來，F Score較高的樂種也是前面提到所謂的tempo較快，而配樂基本上都在拍點上的曲子，而配樂不會剛好在拍點上的其他曲風則會導致較低的F Score。