音乐信号音符/乐谱提取

原创 VIP文章 KevinLeeee 最新推荐文章于 2019-04-29 16:37:23 发布 阅读量1w 收藏 52 点赞数 5

分类专栏： STFT时频分析应用 文章标签： STFT时频分析应用 音乐声学 音频信号处理

1 音符提取方法和基本原理

本实验使用时频分析的方法完成音乐信号的音符提取。时频分析选择短时傅里叶变换(STFT)完成。其具体方法是将信号用一定长度的窗进行分割成若一帧分别作傅里叶变换，得到它们的频率信息，而每一帧对应的位置表示了该帧所在的时间信息，由此可以实现时频分析的功能。

STFT的局限性在于，频率分辨率和时间分辨率的耦合关系。即，频率分辨率的上升会带来时间分辨率的下降，反之亦然，两者不能同时达到最高，分窗口长度。针对这一问题，本次实验需要设定合适的窗口长度，使之能够尽量同时满足频率和时间分辨率的要求。

对需要处理的音乐信号特征进行分析可以得到，该信号为时长为10s，采样频率为8kHz的双声道信号，两个声道的信号差别不大。针对音乐的节奏和可以使用窗口长度为1000的矩形窗，即将信号分为80帧，每帧信号为125ms。然后将每帧信号做fft，再求模平方得到功率谱。求功率谱中能量最大值率点，该频率即为这一时间段对应的住频率，由于每帧信号时间足够短，可以认为该频率对应的音符为这一时间段的主要音符，将其提取出即可。

在得到了各个时刻主频之后，要利用音符音高对应的基频获得主频对应的音符。由于实验者对吉他一窍不通，对钢琴有一定的涉猎，本实验以钢琴琴音符。钢琴的中央C基频约为261.63Hz，唱“do”。根据国际标准，相邻的半个音（即钢琴相邻键）的基频相差2^(1/12)倍。由此标准可以通过将获得的261.63求比例，再求以2^(1/12)为底的对数，获得其对应音符与中央C相差的键数，正值为更高，负值为更低。

钢琴一个八度的12个琴键本别可以表示为如下形式：

Do Do# Re Re# Mi Fa Fa# Sol Sol# La La# Si

其中“#”表示比该音符高半个音的黑键。

为了表示不同八度的音符，用在该音符后面紧跟的数字表示。正数表示高若干八度，负数表示低若干八度。音符持续的时间用“*”后面的数字表示，数的帧数，在时间上，即为该数字乘以125ms。

用该方法表示的钢琴音符谱较为完备。

2实验结果

本次实验获得的每帧主频以及从信号中提取的音乐音符如下：

STEP 8:

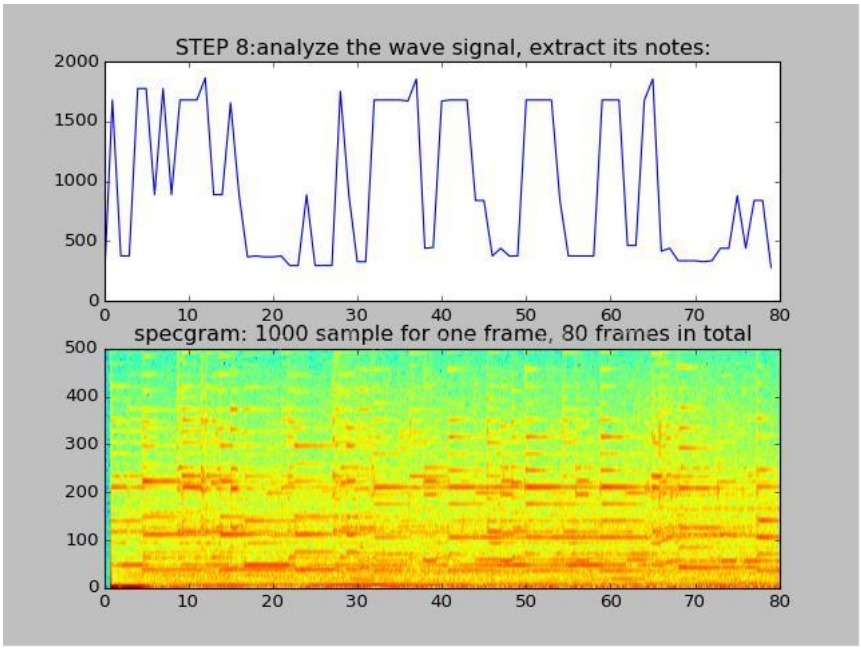
the main frequency of each frame are:

[96.0, 1680.0, 376.0, 376.0, 1776.0, 1776.0, 888.0, 1776.0, 888.0,1680.0, 1680.0, 1680.0, 1864.0, 888.0, 888.0, 1656.0, 888.0, 368.0, 376.0,368.0, 376.0, 296.0, 296.0, 888.0, 296.0, 296.0, 296.0, 1752.0, 888.0,328.0, 328.0, 1680.0, 1680.0, 1680.0, 1680.0, 1672.0, 1856.0, 440.0, 448.0,1672.0, 1680.0, 1680.0, 840.0, 840.0, 376.0, 440.0, 376.0, 376.0,1680.0, 1680.0, 1680.0, 1680.0, 840.0, 376.0, 376.0, 376.0, 376.0, 1680.0,1680.0, 1680.0 464.0, 1680.0, 1856.0, 416.0, 440.0, 336.0, 336.0,336.0, 328.0, 336.0, 440.0, 440.0, 880.0, 440.0, 840.0, 840.0, 280.0]

the notes are:

Sol-2*1 Sol#2*1 Fa#0*2 La2*5 Sol#2*3 La#2*1 La1*2 Sol#2*1 La1*1Fa#0*5 Re0*2 La1*1 Re0*3 La2*2 Mi0*2 Sol#2*5 La#2*1 La0*2 Sol#2*6 Fa#0* La0*1Fa#0*2 Sol#2*5 Fa#0*4 Sol#2*3 La#0*2 Sol#2*1 La#2*1 Sol#0*1 La0*1 Mi0*5 La0*4Sol#1*2 Do#0*1

各帧主频以及时频分析的功率谱图如下：



本次实验获得的音符与音乐的实际音符有差别。主要原因是在同一时刻存在多个音符，而主旋律的音符能量未必是最大的，另外，STFT时频分析方法
率局限性使得对音符基频的时域和频域提取都存在着误差。

3代码（python）

```
# TO DO: analyze the wave signal, extract its notes
#-----------------------------------------
windowsize=1000
halfwindowsize=500
numframe=80
fs=8000
basicfreq=[0 for x in range(0,numframe)] #the main frequency of each frame
basicfreq1=[0 for x in range(0,numframe)]
basicfreq2=[0 for x in range(0,numframe)]
nomalizedbasicfreq=[0 for x in range(0,numframe)] #nomalize the main frequency of each frame according to the standard notes basic frequency
orignotes=[0 for x in range(0,numframe)] #the ratio of each main frequency with the central C 261,63 Hz by log
notes=[0 for x in range(0,numframe)] #the notes represented by number
notes1=[0 for x in range(0,numframe)]
notes2=[0 for x in range(0,numframe)]
ratio=math.pow(2,1/12) #the basic ratio between the neighbour notes
bufferwave=[[0 for x in range(0,windowsize)] for frame in range(0,numframe)] #the signal for each frame
bufferwavefft=[[0 for x in range(0,windowsize)] for frame in range(0,numframe)] #fft each frame
powerbufferwavefft=[[0 for x in range(0,halfwindowsize)] for frame in range(0,numframe)] #get the power of each frame
#get the dictionary for the basic 12 note in one octave
dictnotes={0: 'Do', 1: 'Do#', 2: 'Re', 3: 'Re#', 4: 'Mi', 5: 'Fa', 6: 'Fa#', 7: 'Sol', 8: 'Sol#', 9: 'La', 10: 'La#', 11: 'Si'}
notesbywords=[]
#the time noted by the number of frames for each note
notestime=[]
notestime1=[]
notestime2=[]
ampli=[]
ampli1=[]
ampli2=[]
for i in range(0,numframe):
bufferwave[i]=wave_data[0][i*windowsize:i*windowsize+windowsize]
bufferwavefft[i]=np.fft.fft(bufferwave[i])
w8=0
powerw8=0
for w in range(10,halfwindowsize): #to eliminate the low frequence noice search from 80 Hz
powerbufferwavefft[i][w]=math.pow(abs(bufferwavefft[i][w]),2)
if powerbufferwavefft[i][w]>powerw8:
powerw8=powerbufferwavefft[i][w]
w8=w*fs/windowsize
```

```
ww8=w
basicfreq[i] = w8
orignotes[i] = round(math.log(w8 / 261.63, ratio))
note = round(math.log(w8 / 261.63, ratio))  # get the ratio with 261.63 Hz by log
nomalizedbasicfreq[i] = ratio ** note * 261.63
ampli.append(powerw8**(1/2))
for x in range(ww8-1,ww8+2):
powerbufferwavefft[i][x]=0
if note >= 0:
if note < 12:
k = 0  # get the octave range of this certain note
notes[i] = note
else:
k = 0
while note >= 12:
k = k + 1
note = note - 12
notes[i] = note
else:
k = 0
while note < 0:
k = k - 1
note = note + 12
notes[i] = note
if i == 0:
notesbywords.append(dictnotes[notes[i]] + '%d' % k)
k = 0
length = 1  # get the number of frame this certain note remains
else:
if notes[i - 1] != notes[i]:
notesbywords.append('*%d ' % length + dictnotes[notes[i]] + '%d' % k)
notestime.append(length)
k = 0
length = 1
else:
length = length + 1
powerw8=0
for w in range(10, halfwindowsize):  # to eliminate the low frequence noice search from 80 Hz
#powerbufferwavefft[i][w] = math.pow(abs(bufferwavefft[i][w]), 2)
if powerbufferwavefft[i][w] > powerw8:
powerw8 = powerbufferwavefft[i][w]
w8 = w * fs / windowsize
ww8 = w
note1 = round(math.log(w8 / 261.63, ratio))  # get the ratio with 261.63 Hz by log
basicfreq1[i] = ratio ** note1 * 261.63
ampli1.append(powerw8 ** (1 / 2))
for x in range(ww8 - 1, ww8 + 2):
powerbufferwavefft[i][x] = 0
if note1 >= 0:
if note1 < 12:
k = 0  # get the octave range of this certain note
notes1[i] = note1
else:
k = 0
while note1 >= 12:
k = k + 1
note1 = note1 - 12
notes1[i] = note1
else:
k = 0
while note1 < 0:
k = k - 1
note1 = note1 + 12
notes1[i] = note1
if i == 0:
```

```
notesbywords.append(dictnotes[notes1[i]] + '%d' % k)
k = 0
length1 = 1  # get the number of frame this certain note remains
else:
if notes1[i - 1] != notes1[i]:
notesbywords.append('*%d ' % length + dictnotes[notes1[i]] + '%d' % k)
notestime1.append(length1)
k = 0
length1 = 1
else:
length1 = length1 + 1
powerw8=0
for w in range(10, halfwindowsize):  # to eliminate the low frequence noice search from 80 Hz
#powerbufferwavefft[i][w] = math.pow(abs(bufferwavefft[i][w]), 2)
if powerbufferwavefft[i][w] > powerw8:
powerw8 = powerbufferwavefft[i][w]
w8 = w * fs / windowsize
note2=round(math.log(w8/261.63,ratio)) #get the ratio with 261.63 Hz by log
basicfreq2[i] = ratio**note2*261.63
ampli2.append(powerw8 ** (1 / 2))
if note2>=0:
if note2<12:
k = 0 #get the octave range of this certain note
notes2[i]=note2
else:
k=0
while note2>=12:
k=k+1
note2=note2-12
notes2[i]=note2
else:
k=0
while note2<0:
k=k-1
note2=note2+12
notes2[i]=note2
if i==0:
notesbywords.append(dictnotes[notes2[i]]+'%d'%k)
k = 0
length2 = 1 #get the number of frame this certain note remains
else:
if notes2[i-1]!=notes2[i]:
notesbywords.append('*%d '%length+dictnotes[notes2[i]]+'%d'%k)
notestime2.append(length2)
k = 0
length2 = 1
else:
length2=length2+1
notesbywords.append('*%d' % length)
notestime.append(length)
notestime1.append(length1)
notestime2.append(length2)
music=(").join(notesbywords)
print("STEP 8:")
print("the main frequency of each frame are:")
print(basicfreq) #print the basic frequency of the notes
print("the notes are:")
print(music) #print the notes using the rule described in the report
print(notestime)
print(notestime1)
print(notestime2)
print(ampli)
print(ampli1)
print(ampli2)
#i am afraid that you can't read the notes if you don't read chapter 2.8 of my report
```

```
plt.figure()
plt.subplot(211)
plt.plot(basicfreq)
plt.title(r'STEP 8:analyze the wave signal, extract its notes:')
plt.subplot(212)
plt.specgram(wave_data[0],Fs=1000)
plt.title(r'specgram: 1000 sample for one frame, 80 frames in total')
```