Handout instructions for beat tracking workshop

25/02/14

This document includes the instructions for using Sonic Visualiser and MATLAB in the context of the beat tracking workshop.

Please read the instructions closely. Most functions will only work if you type in the information exactly as is written on this sheet.

The majority of the information here will also be in the presentation slides, however it may be easier to work directly from this paper. **Any questions, just ask!**

Part II - Beat Tapping Experiment - Sonic Visualiser

Tapping to music examples

Open Sonic Visualiser

Goto *File*, *Open*, and browse to the directory ***beat_workshop*** directory

select "*tap_example1.wav*", click *Open*

Goto *Layer* and then *Add new time instants layer*

Locate the ';' (semi-colon) key on the keyboard

Put on **headphones**! Press the space bar to start audio playback

Tap the beats to the music using the ';' key

Saving the output

You can now listen back to the taps by pressing the space-bar again to restart playback

The beat taps will sound as "clicks"

Write down if you thought the taps were good (below)

_____

Let's save the taps for **objective evaluation** too

Go to *File*, *Export Annotation Layer*

Browse to the beat_workshop directory

Choose "*Files of type*" .txt files (**IMPORTANT!! - it won't work if you choose .svl)**

Write the file name **exactly** as "**my_tap_example1.txt**" and Click *Save*

Tapping another example

Before doing the objective evaluation let's do another example

In **Sonic Visualiser** go to *File*, *New Session* (say '**no**' to the save dialog box
Now, as before, load a new file, this time "*tap_example2.wav*"

And repeat everything as before, eventually saving your taps as "**my_tap_example2.txt**" in the ***beat_workshop*** directory

After listening back to your taps for "*tap_example2.wav*" again write down if you thought your taps were good

_____

Objective evaluation in MATLAB

Open **MATLAB**

Browse to the ***beat_workshop*** directory

Then on the command prompt type the following **exactly**:

```
score1 = beat_evaluation('my_tap_example1.txt')

score2 = beat_evaluation('my_tap_example2.txt')
```

Note down score1 and score2

`score1` _____%

`score2` _____%

How do they compare with your subjective assessment of how you well you tapped?

Part III - Creative applications of beat tracking - MATLAB

We'll now work through a few different examples in **MATLAB**. There's no programming involved. You'll just be calling functions in **MATLAB** by typing the commands shown on the handout sheet or on the screen. All MATLAB commands will be in `courier` font.
The aim here is to experiment with different songs and hear the results. Although some example audio files are labelled, feel free to choose any .wav file within the ***beat_workshop*** directory.

Beat Tracking - *let's test extracting the beats for a music signal*

In the **MATLAB** command prompt type:

clear; clc;

```
beats = beat_tracker('soulfunk.wav')
```

note the lack of a `';'` at the end means the output will be displayed on the screen

where `'soulfunk.wav'` corresponds to one of the .wav files in the working directory

`beat_tracker` - is the function name

`'soulfunk.wav'` is in the input to the function

`beats` is the output of the function - a list of times in seconds

Beat Randomizer - *scramble the music by randomly re-ordering the beats*

In the MATLAB command prompt type:

```
x = wavread('soulfunk.wav');

out = beat_randomizer('soulfunk.wav');
```

`'file.wav'` corresponds to a .wav file in the beat_workshop folder

To listen to the original "`x`" and the result "`out`" (on **headphones**) type:

```
soundsc(x,44100);

soundsc(out,44100);
```

If you want to save the file, (e.g. for later playback in Sonic Visualiser) type:

```
wavwrite(out,44100,'beat_randomizer1.wav');
```

We can experiment with the number of sub-divisions per beat, e.g. 2, 4, 8 by adding an extra input parameter to the function

```
out_2 = beat_randomizer('soulfunk.wav',2);

soundsc(out_2,44100);
```

What happens as you increase the number from 2 to 4? Does it still sound like music?

Beat Randomizer with Metre - *scramble the music but preserve the feeling of metre*

For a given .wav file, try:

```
x = wavread('electronic.wav');

out = beat_randomizer('electronic.wav');

out_metre = beat_randomizer_metre('electronic.wav');
```

And then compare the results (on headphones)

```
soundsc(x,44100);

soundsc(out,44100);

soundsc(out_metre,44100);
```

Once again, we can vary number of sub-divisions per beat, e.g. 2, 4

```
out_metre2 = beat_randomizer_metre('electronic.wav',2);

soundsc(out_metre2,44100);
```

Try using `'jazz_solo.wav'` as the input
What do you notice with this file compared to other inputs?

Beat Swinger - *apply a rhythm transform to add "swing" feel to the music input*

```
x = wavread('smooth.wav');

out_swing = beat_swinger('smooth.wav');

soundsc(x,44100);

soundsc(out_swing,44100);
```

To experiment more, we can modify how the beats are sub-divided, and change the stretch factor, e.g.

```
out_swing_4 = beat_swinger('electronic.wav',4);
```

Best results are obtained through experimentation, and finding the right metrical level to swing (normally the fastest)

But, we could do really weird things too

```
out_swing_1 = beat_swinger('electronic.wav',1,2/3);
```

reverse-swing feel at the beat level rather than the 1/8th note level :)

Beat Remixer - *apply beat_randomizer_metre to two songs and randomly mix between them*

```
x1 = wavread('house.wav');
x2 = wavread('electronic.wav');

out_remix = beat_remixer('house.wav','electronic.wav');
```

Let's listen to each input in turn and then the result

```
soundsc(x1,44100);
soundsc(x2,44100);
soundsc(out_remix,44100);
```

As with the other functions we can provide some additional input parameters to shape the result, e.g.
the sub-beat level for randomizing beats
the probability of choosing one song over another
the number of beats to use in the output

vary the sub-beat level
```
level = [2 4 2 1];
```

vary the probability of choosing one song over another
```
prob = [0.25 0.5 0.75 0.5];
```

make a loop to create four separate outputs using each of the levels and probabilities
```
for k=1:4,outr{k}=beat_remixer('pop.wav','electronic.wav',level(k),prob(k),4);end;
```

make a pattern to group the outputs together
```
pattern = [2 2 2 1 2 2 2 1 4 4 4 3];
```

create a new file zz which will be the final output
```
zz = [];
for k=1:length(pattern), zz=[zz outr{pattern(k)}(:)']; end;
zz = zz(:);
```

listen to zz
```
soundsc(zz,44100);
```

Please feel free to experiment with different input songs

You can also listen to each individual pattern
```
soundsc(outr{1},44100);
soundsc(outr{2},44100);
soundsc(outr{3},44100);
soundsc(outr{4},44100);
```

and make your own custom pattern, e.g.

```
pattern = [1 2 3 4 1 2 3 4];
zz = [];
for k=1:length(pattern), zz=[zz outr{pattern(k)}(:)']; end;
zz = zz(:);
```

```
soundsc(zz,44100);
```

Here is the code again for easy copying and pasting

```
for k=1:4, outr{k}=beat_remixer('pop.wav','electronic.wav',level(k),prob(k),4); end;
pattern = [2 2 2 1 2 2 2 1 4 4 4 3];
zz = [];
for k=1:length(pattern), zz=[zz outr{pattern(k)}(:)']; end;
zz = zz(:);
soundsc([zz zz],44100); % this will play the whole thing twice
```

Here is the code to save a remix that you like

```
wavwrite([zz zz],44100,'finaloutput.wav');
```

Note, the output will change every time, so you should experiment!