BLG354E
Homework-4

15.06.2020

Res. Asst. Yusuf Huseyin Sahin
sahinyu@itu.edu.tr

Synesthesia : *n.* a condition in which a secondary subjective sensation (often colour) is experienced at the same time as the sensory response normally evoked by the stimulus. For example, the sound of the word 'cat' might evoke the colour purple.

Oxford Concise Medical Dictionary (8 ed.)

- You should write all your code in Python language.
- Cheating is highly discouraged.
- Ninova only stores files under 20 MB. If you could not upload your results, you can share them with me via Dropbox, or send me private YouTube video links for each part's results.
- **It is a very short homework and there will be no postponement.**

1 - Audio Visualization (50 pts)

In this part of the homework, you will use DFT for audio visualization. Steps are given below for a basic audio visualizer:

- Select the visualization renewal rate (e.g. 10 FPS). Since you know the sample rate in Hertz (44100 in the given file.), you can select data from the audio array which is matching with renewal time. For example, to visualize at 10 FPS, each visualization interval comes from 4410 values in sampled audio matrix. Pad the audio matrix with zeroes if necessary.
- For each visualization interval;
  - Find FFT of that interval using built-in functions. Since it's in exponential form, it will give an array of complex numbers. Find the magnitude of each value.
  - The transformed values have a great range because of the outliers. For a better representation, logarithm operation $10 * log10(x_{ft} + c)$ is used for each value. Here c can be a small number to avoid $log10(0)$.

  - Since the obtained transformation is mirrored (see conjugate symmetry property) you may use:
    * The first half of the array (http://web.itu.edu.tr/sahinyu/HallOfTheMountainKingN2.mp4, https://web.itu.edu.tr/sahinyu/SilentKnight.mp4)
    * Shifted version of the array using fftshift function (http://web.itu.edu.tr/sahinyu/HallOfTheMountainKingFFTShift.mp4)

- After obtaining transforms for all visualization intervals, you can plot the transforms using matplotlib's "bar" function and write the results to a video file. An example video file writing operation using OpenCV is given below.

```
import numpy as np
import matplotlib
matplotlib.use("Agg")
import matplotlib.pyplot as plt
import cv2

fig, ax = plt.subplots( nrows=1, ncols=1,figsize=(10, 6))
ax.get_xaxis().set_visible(False)
ax.get_yaxis().set_visible(False)

out = cv2.VideoWriter('output.avi', cv2.VideoWriter_fourcc(*'XVID'),
    24.0, (1000,600))

for i in range(50):
    ax.plot(np.arange(1,101,1), np.random.rand(100,1))
    plt.savefig('snap.png', format='png')
    ax.clear()
    X = cv2.imread("snap.png")
    out.write(X)
out.release()
```

  You can also use moviepy library to write the frames as a video. I personally prefer this simple but effective library rather than OpenCV's video writing operations. You can create an ImageSequenceClip object and write a video as given in the following lines. (Note: While OpenCV uses BGR channel order by default, moviepy uses RGB.)

```
imageslist = []

#Add the processed images to the list.

clip = ImageSequenceClip(imageslist, fps =24)
clip.writevideofile('part1video.mp4' , codec='mpeg4')
```

- To write the audio on the video, you can benefit from moviepy library:

```
import moviepy.editor as mpe

my_clip = mpe.VideoFileClip('output.avi')
audio_background = mpe.AudioFileClip('SilentKnight.wav')
final_clip = my_clip.set_audio(audio_background)

final_clip.write_videofile('total_output.mp4')
```

Create an audio visualization of the given song. If the result mp4 file is larger than 20 MB, you can share the file with me via Dropbox.

Note: To speed-up the figure rendering, you can use 'plot' function instead of bar function. However, do not forget to limit the y-axis with constant values. If you use plot function, a frame from your solution will be like as given below.

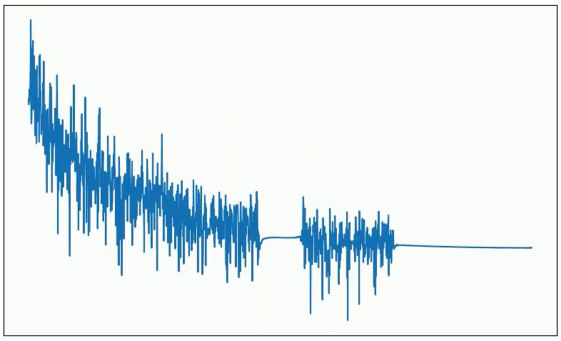

Figure 1: An example output when plot is used.

2 The Spectrogram (50 pts.)

In this part of the homework you will work on a part of Aphex Twin's famous song $\Delta M_i^{-1} = -\alpha \sum_{n=1}^{N} D_i[n][\sum_{j \in C[i]} F_{ji}[n-1] + F_{ext_i}[n^{-1}]]$. The artist hid a face in the song which can be seen by creating a spectrogram and analyzing it in log scale. To do this,

- Normalize the signal according to its max value as you've done in the previous homework.
- Define two parameters:
  - **Window Size:** Size of the **rectangular** window
  - **Step:** It is unnecessary to calculate the FFTs for every data point. Thus, starting from 0^{th} data point, you should decide on next data points according to step value.

- Fill the spectrogram matrix using absoulte FFTs of the windowed data. As in the previous part, you can also work on the first half of FFT.
- Use $10 * log10(x + c)$ to smooth the data for a better representation.
- Use pcolormesh function of matplotlib to obtain a visual from the data. The artist hid the face into log scale. Thus, you should select 'y_scale' as 'symlog'.

If everything goes well, you will obtain the face given in Figure 2.

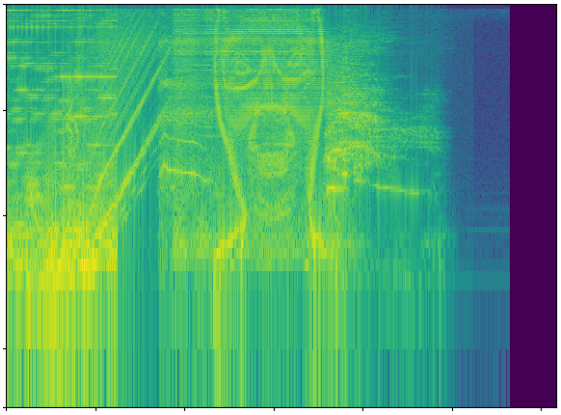

Figure 2: Aphex Twin's hidden face in the song.