Assignment 9: Analysis / Synthesis of Speech

CS 4347: Sound and Music Computing

due Sunday 12 April 2015, 11:59pm SGT

NOTE: For inquiries on this assignment, please contact ZHU Shenggao (a0107950@nus.edu.sg).

0. Download this file; note that it's sampling rate is Fs = 22050 Hz.
   `http://www.comp.nus.edu.sg/~duanzy/clear_d1.wav`

1. Analyze the above audio file.
   - Load the wav file and split the data into frames. Each frame has a window size $N = 128$ samples and has *no* overlap. Only include complete frames (e.g., if the final frame has less than N samples, omit that frame). Hint: total # of frames = 463
   - For each frame, apply a Hamming window first, and then calculate the magnitude spectrum using FFT. Remember to discard the "negative frequencies" of the spectrum (i.e., array indices above $N/2$).
   - For each frame's magnitude spectrum, find the maximum value (noted as M) of the spectrum, as well as the index of M. The corresponding frequency in Hz (noted as f) of that index is given by: f = index / N * Fs. Store the frequency f and amplitude M in an array.
     - Hint: first create an array to store these values for all frames:
       `freq_amp = numpy.zeros((n_frames, 2)).`
   - Output that array to a csv file `freq_amp.csv`. To make the output more legible, use:

     ```
     numpy.savetxt('freq_amp.csv', freq_amp, fmt='%.6g', delimiter=',')
     ```

     The file should begin (small variations due to floating-point inaccuracy may occur):

     ```
     10335.9,0.00139136
     10335.9,0.00117254
     11025,0.0013192
     ...
     ```

2. Reconstruct that audio using sine waves.
   - For each frame, based on its frequency f and amplitude M calculated above, generate a sine wave with N samples. The sine wave should also have frequency f and amplitude M. Hint: sample code for reference

     ```
     sine_wav = M * numpy.sin(2 * numpy.pi * f / Fs * numpy.arange(N))
     ```

   - Reconstruct the audio. Concatenate the generated sine waves of all frames into an array (say, `re_wav`), and save it as an audio file `reconstructed.wav`. Hint: normalize the array and convert it to 16-bit integer first. Sample code:

     ```
     re_wav_norm = re_wav / re_wav.max() * 32767
     re_wav_norm = re_wav_norm.astype(numpy.int16)
     scipy.io.wavfile.write('reconstructed.wav', Fs, re_wav_norm)
     ```

- When listening to the reconstructed audio, you may want to lower your computer's volume at first (just in case some surprising sound is generated)!

3. Plot the spectrograms of the original audio and the reconstructed audio.
   - Hint: for each audio, save the magnitude spectrum of all frames into an array (463 rows by 65 columns). This array is called the spectrogram. Remember window size $N = 128$, *no* overlap, Hamming windowing.
   - Plot the two spectrograms (say `spectrogram` and `re_spectrogram`) as two subplots in the same figure. Normalize each spectrogram (so that the maximum becomes 1) before plotting. The x-axis should be the frames, and y-axis the frequency bins. Also remember to add axis labels and titles to make a good figure. Save the figure as `spectrogram.png`. Hint: use `pylab.imshow()`. Sample code:

```
pylab.subplot(2,1,1)
pylab.imshow(spectrogram.T/spectrogram.max(),
             origin='lower', aspect='auto')
pylab.subplot(2,1,2)
pylab.imshow(re_spectrogram.T/re_spectrogram.max(),
             origin='lower', aspect='auto')
```

   - Figure 1 shows an example of the spectrograms:

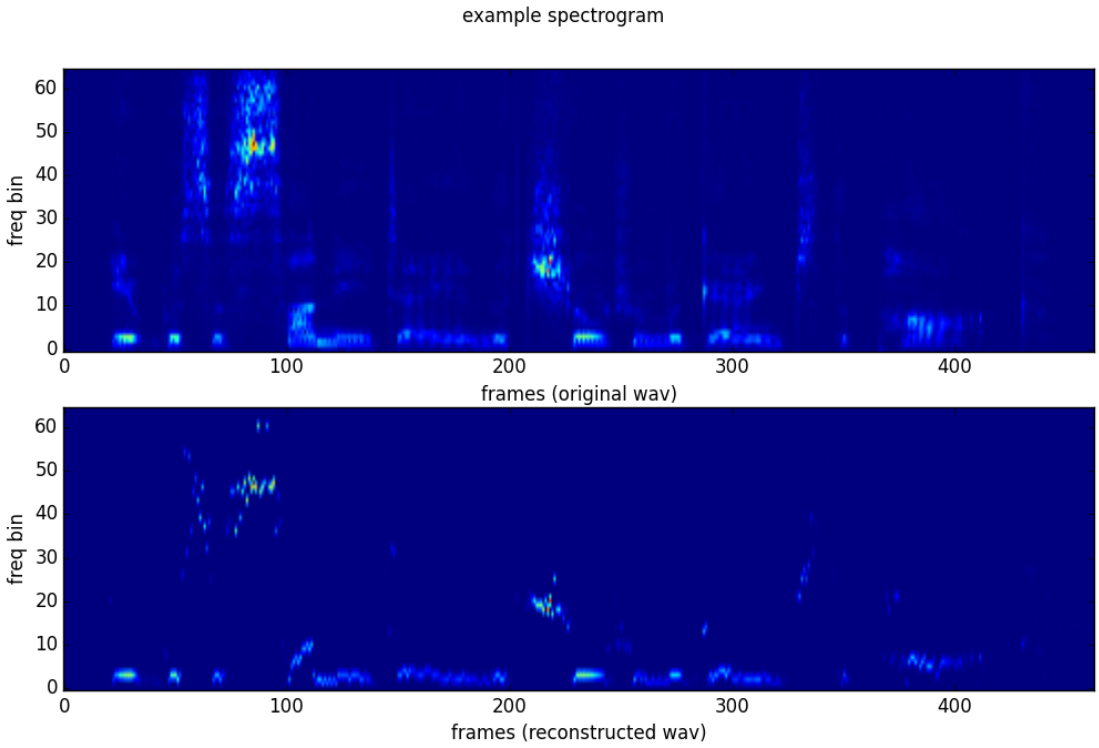

Figure 1: Example spectrogram

4. Write a file called `discussion.txt` and answer the following questions:
   - Listen to your reconstructed audio. Describe the difference between it and the original audio. Briefly explain the reasons for this difference.
   - Compare the two spectrograms in your plot. What difference can you observe regarding the frequency distribution for each frequency bin?

5. Upload your result csv file `freq_amp.csv` to

   `http://cs4347.smcnus.org`

   This will automatically grade the values you calculated. If any mistake is found, please check your program and resubmit. You are welcome to submit as many versions as you wish before the submission deadline.

   Submit a zip file containing (1) your program's source code (as a .py), (2) the reconstructed audio file `reconstructed.wav`, (3) the figure `spectrogram.png`, (4) your discussion file `discussion.txt`, and an optional README.txt file to the same website. You may use anything in the python standard library, numpy (including pylab / matplotlib), and scipy libraries. No other libraries are permitted.

   Grading scheme:

   - **2/6 marks**: automatic grading of `freq_amp.csv`
   - **1/6 marks**: reconstructed audio file `reconstructed.wav`
   - **1/6 marks**: the figure `spectrogram.png`
   - **1/6 marks**: discussion file `discussion.txt`
   - **1/6 marks**: source code quality