MATLAB HW1 (Due date Week 4)

Group work is allowed. Upload a single archive file (.zip or .rar file) to ninova for each group. The archive file should include a report (.pdf file with the names of the students, all necessary explanations and code) and all the other necessary supporting materials for the HW.

2 Music Transcription Problem

2.1 Projection

Recording of "Polyushka Poly" is played on the harmonica, the file name of the recording is `polyushka.wav`, which can be found in folder `hw1materials` . It has been downloaded from YouTube with permission from the artist.

A set of notes from a harmonica can be found in folder `hw1materials/note15`. You are required to transcribe the music. For transcription you must determine how each of the notes is played to compose the music.

You can use the matlab instructions given in ***section 3*** to convert each note into a spectral vector, and the entire music to a spectrogram matrix.

1. Compute the contribution of all notes to the entire music jointly. Mathematically, if $\mathbf{N} = [N_1, N_2, ...]$ is the note matrix where the individual columns are the notes, find the matrix $\mathbf{W}$ such that $\mathbf{NW} \approx \mathbf{M}$. The i_{th} row of $\mathbf{W}$ is the transcription of the i_{th} note.

2. Recompose the music by "playing" each note according to the transcription you just found in last question. Set all negative elements in W to zero. Compute $\hat{M} = \mathbf{N}W$, and invert the result to produce music. Return the recomposed music(although we will not score you on music).

2.3 Linear Transformation

Here we have two audios of a simple rhythm of song "silent night" played by piano (audio A) and classical guitar (audio B), and one audio of a simple rhythm of song "little star" played by piano (audio C).

All audios are given in `hw1materials/Audio`.

Based on information you have, how would you get the classical guitar version (audio D) of song "little star"? Give the final magnitude spectrogram of audio D and convert it to audio (.wav file).

Hint: Learn a linear transformation that converts the magnitude of the spectrogram for a frame in audio A to the magnitude of spectrogram for the corresponding frame in audio B. Then apply it to audio C. You may find all given audios have two channels, in this case, just work with the first channel of audio. Don't forget that the magnitude of a complex number is always a positive value. Use 1024 as sample window, instead of 2048.

Use the instruction of **section 3** to compute spectrograms and reconstruct the signal.

3 Matlab Instructions

3.1 Computing the spectrogram

Get the matlab file `stft.m` in folder `hw1materials`.
`stft.m` computes the complex spectrogram of a signal.
You can read a wav file into matlab as follows.

```
[s,fs] = audioread('filename');
s = resample(s,16000,fs);
```

Above, we resample the signal to a standard sampling rate for convenience. Next, we can compute the complex short-time Fourier transform of the signal, and the magnitude spectrogram from it, as follows. Here we use 2048 sample windows, which correspond to 64ms analysis windows. Adjacent frames are shifted by 256 samples, so we get 64 frames/second of signal.

To compute the spectrogram of a recording, e.g. the music, perform the following.

```
spectrum = stft(s',2048,256,0,hann(2048));
music = abs(spectrum);
sphase = spectrum./(abs(spectrum)+eps);
```

This will result in a 1025-dimensional (rows) spectrogram, with 64 frames(columns) per second of signal.

Note that we are also storing the phase of the original signal. We will need it later for reconstructing the signal. We explain how to do this later. The **eps** in this formula ensures that the denominator does not go to zero.

You can compute the spectra for the notes as follows. The following script reads the directory and computes spectra for all notes in it.

```
notefolder = 'notes15/';
listname = dir([notesfolder '*.wav']);
notes = [ ];
for k = 1:length(listname)
    [s,fs] = audioread([notesfolder listname(k).name]);
    s = s(:,1);
    s = resample(s,16000,fs);
    spectrum = stft(s',2048,256,0,hann(2048));
    %Find the central frame
    middle = ceil(size(spectrum,2)/2);
    note = abs(spectrum(:,middle));
    %Clean up everything more than 40db below the peak
    note(find(note < max(note(:))/100)) = 0;
    note = note/norm(note);
    %normalize the note to unit length
    notes = [notes,note];
end
```

The "`notes`" matrix will have as many columns as there are notes (15 in our data). Each column represents a note. The notes will each be represented by a 1025 dimensional column vector.

3.2 Reconstructing a signal from a spectrogram

The recordings of the complete music can be read just as you read the notes. To convert it to a spectrogram, do the following. Let **reconstructedmagnitude** be the reconstructed magnitude spectrogram from which you want to compute a signal. In our homework we get many variants of this. To recover the signal we will use the **sphase** we computed earlier from the original signal.

```
reconstructedsignal = stft(reconstructedmagnitude.*sphase,2048,256,0,hann(2048));
```