Introduction

In this assignment we'll make a second attempt at aligning audio files.

Submission details

- This assignment can be done individually or in pairs (though we strongly encourage you to work in pairs).
- Programs should be in Python. You can use Python 2 or Python 3.
- For submission, package up your code as a zip or tar file. Include your written answers as a pdf or word file named writeup.[pdf|docx]. There's a bunch of graphs we want you to plot, include these in your writeup, and please label them so we know which figures go with which sub-problem.
- If you have any questions about this assignment, please contact the TA (stinger@tx.technion.ac.il). Your email subject should start with CS5660.

Aligning Audio Signals – second attempt

You are going to try and improve the audio alignment you got in the previous HW assignment. This time you'll use other methods for matching a pair of peak maps. You can use the files and the peak maps you got in the previous HW assignment.

A. Using Line Fitting

You are going to build functions that compute the similarity between two audio files: audio1 and audio2.

1. Hash table:
    a. Given a peak map, you should build an array y that contains hash values of pairs of peaks in the peak map. To collect pairs of peaks you will pair each peak with at most N peaks following it. You should consider only peaks that are up to T time steps away, and differ up to F bands in frequency. You get to determine the number N, the gap in time T and the frequency gap F. T and F define a rectangular "target zone", adjacent to each peak, in which lie the peaks to be paired.
    b. Now, fill up the table y by setting y(i)=[t1,h], where h is the hash value of the i-th pair of peaks and t1 is the time at which the first peak occurred. The hash value h of a pair should be computed from f1 (the frequency of the first peak), f2 (the frequency of the second peak) and t2-t1 (the time difference between the peaks). Choose your favorite hash function and don't forget to describe what it does.
2. Matching hash values:
    Let y1 be the hash table of audio1 and y2 be the hash table of audio2. We next wish to match the two hash tables.
    a. Find all matching hash value across the two tables.
    b. Record all matches in a list of 2D pairs (t1, t2) where t1 represents the time the value h appeared in audio1 and t2 represents the time the value h appeared in audio2. The output of this stage is a list of 2D points.
    c. Produce a scatter plot of the points you got for 3 pairs of audio files. Use the same tracks you used in HW1. Add these scatter plots to your report.
    d. Do you see the patterns you were expecting to see?

3. Matching audio segments:
   To match the two audio files you need to find a sequence of hash values that appears in both audio files. Such a sequence should look like a line in the scatter plot. To detect such lines you will use two model-fitting methods:
   a. Implement line fitting by Hough transform and test it on your data.
   b. Implement line fitting by RANSAC and test it on your data.
4. Use the same 3 pairs of tracks you used in HW1 and test the above methods. How successful was each method compared to HW1? Did you get the same results from both?

B. Using Dynamic Time Warping

Next thing we'll do is try the same thing but this time using Dynamic Time Warping, instead of line fitting.

1. DTW is meant to match two time series. It expects to get as input two 1D sequences. You need to convert your arrays y1 and y2 into 1D sequences of the hash values. Write a function that converts a 2D array y of values (t, h) into a 1D array z of values h. If there is more than one hash value at a certain time, order them in z according to the hash value.
2. Implement a function that computes the DTW distance between two sequences.
3. Implement a function that computes the EDRP (Edit Distance with Real Penalty) between two sequences.
4. Compute the DTW and EDRP between the 3 pairs of tracks. Do they agree on the order of similarity?
5. Now, for each pair, take from audio1 only the sub-sequence corresponding to the mutual sentence (you should have the start and end point ready from HW1). Use a sliding window approach to compute the distance between that sub-sequence and all sub-sequences, of the same length, of audio2. Plot the resulting scores. Does the maximum correspond to the mutual sentence in audio2?
6. Now repeat step 5 but switch between audio1 and audio2. Do you get corresponding results to the ones you got in the previous item?