Jeehyun Kim
Yihan Chan

Signal Processing
November 5, 2015

Homework 2

Source citation for Part B:
https://github.com/pierre-rouanet
https://en.wikipedia.org/wiki/Dynamic_time_warping
https://github.com/mission-peace

Aligning Audio Signals - second attempt

A. Using Line Fitting
Part A is done in IPython Notebook.

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.

See the implementation in HW2.IPynb.

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 dont forget to describe what it does.

I used linear hash function

$$f1 + (f1 - f2) + (t2 - t1)$$

for this question. This is function represents a relationship between signal at t1 and signal at t2.

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.

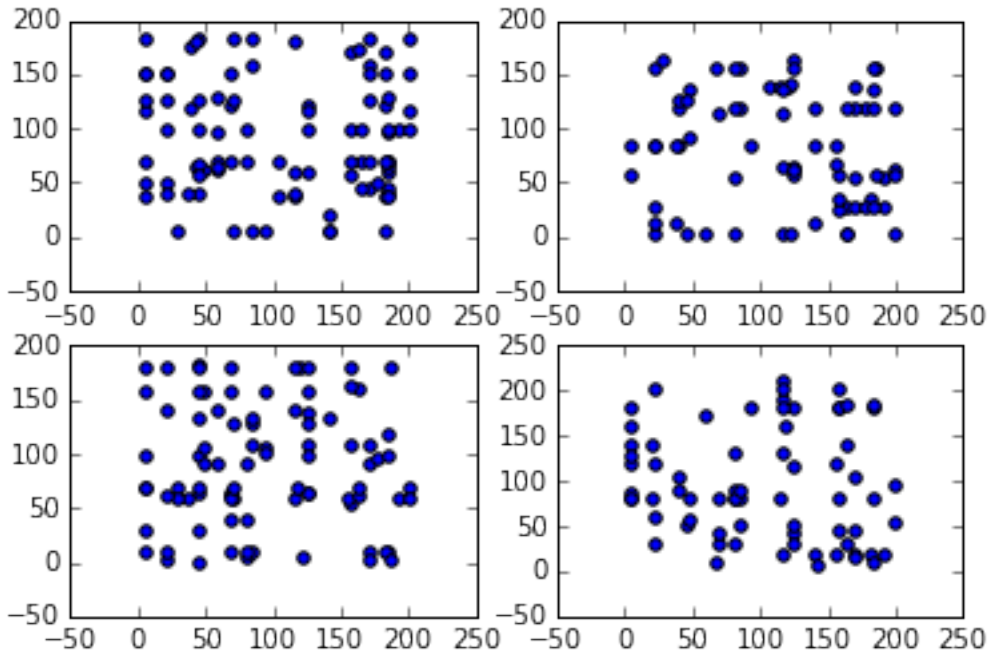
a. Find all matching hash value across the two tables.

See the implementation in HW2.IPynb.

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.

See the implementation in HW2.IPynb.

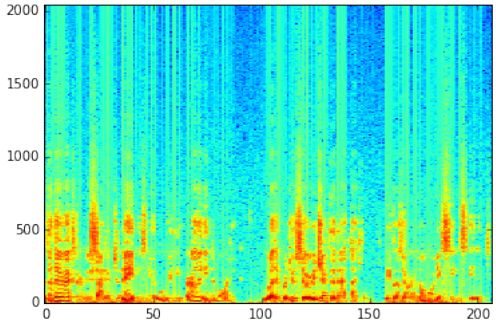
c. Produce a scatter plot of the points you got for 3 pairs of audio files. Use the same tracks you used in HW1. Add these scatter plots to your report.



from left to right, up to down is the hashmap got from pairing [audio1-audio2],[audio1-audio3],[audio1-audio4],[audio1-audio recorded by my teammate in last homework]

d. Do you see the patterns you were expecting to see?

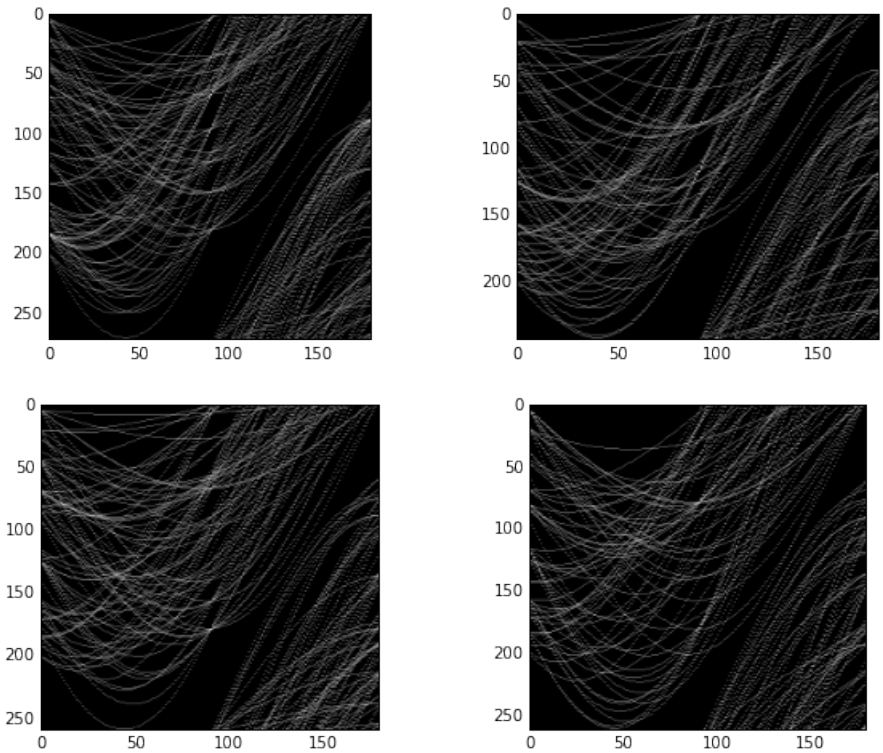
As images shown above, I can't easily fine a line that expects to see. I think the problems could be multiple. First, the harsh function we use may not be well differentiable, that's to say, there are many collisions which result in a grid-like hashmap. Second, the method we choose to pick peaks might be problematic.



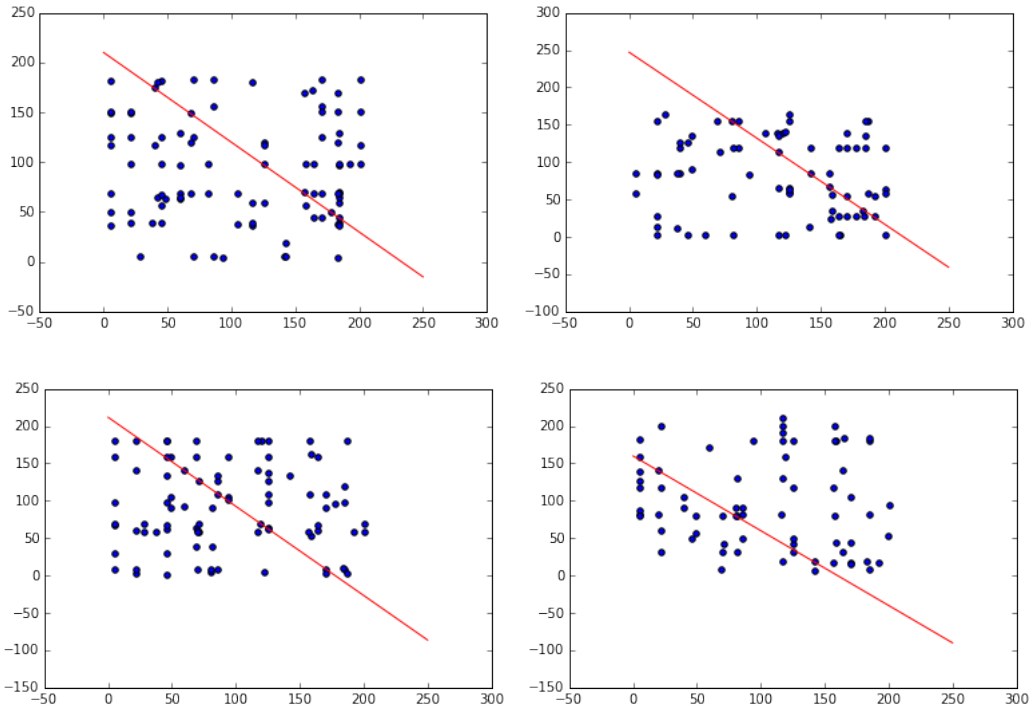
Through the spectrogram we can see that the peaks appear within certain frequency and time domain. In last homework we pick peaks from a 20*20 region which I cast doubt on because the scale of frequency[0:2000] are actually much larger than scale of time [0:250]. Therefore, I set the region to 20*200.

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.

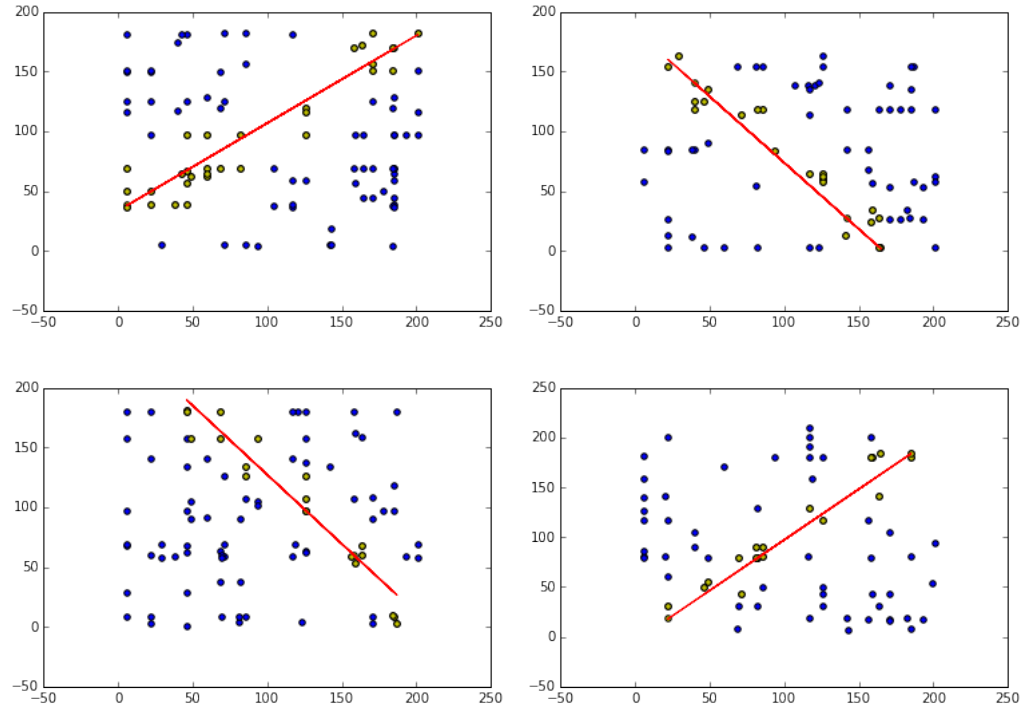


I implement my own hough algorithm. See the details in HW2.ipynb. Above is the plot of Hough space. From the images I didn't find a noticeble dot that represent a line. Therefore, I pick up Hough peaks and draw the peak line on hashmap.



b. Implement line fitting by RANSAC and test it on your data.

I implemented my own RANSAC function. See the detail in HW2.ipynb. Here is the RANSAC result:



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?

The test result has been shown above. I think the performance of hash method is not good enough. A desirable line should be a continuous set of dots shaped in line within a certain time domain. But actually there is no visible line pattern shown in hashmap. I think the problem should be mainly attributed to the hash function selection and peaks pickup.

B. Using Dynamic Time Warping

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.

I used the function conv_to_1D(y, sample). Here,

```
ysort = sorted(y.items(), key=operator.itemgetter(0))
```

is to sort y by time. Next, if there is more than one hash value mapped to the key, I sorted by using y1.sort().

2. Implement a function that computes the DTW distance between two sequences.

I made two attempts before using the Python dtw library. For the actual computation, I used the code from https://github.com/pierre-rouanet/dtw/blob/master/dtw.py, which is the source code for the library. Instead of importing the library, I looked at source code directly in order to see how the library was implemented.
My previous attempts are functions dtw2, and dtw3, also in the same script. The part I struggled is finding the path, which is dealt with _trackeback(D) in the original author's program.

3. Implement a function that computes the EDRP (Edit Distance with Real Penalty) between two sequences.

The functions are called edrp1(z1, z2, m , n) and edrp2(z1, z2) in the program. edrp1 computes recursively, and edrp2 computes using dynamic programming. For the actual computation, I used edrp2.

4. Compute the DTW and EDRP between the 3 pairs of tracks. Do they agree on the order of similarity?

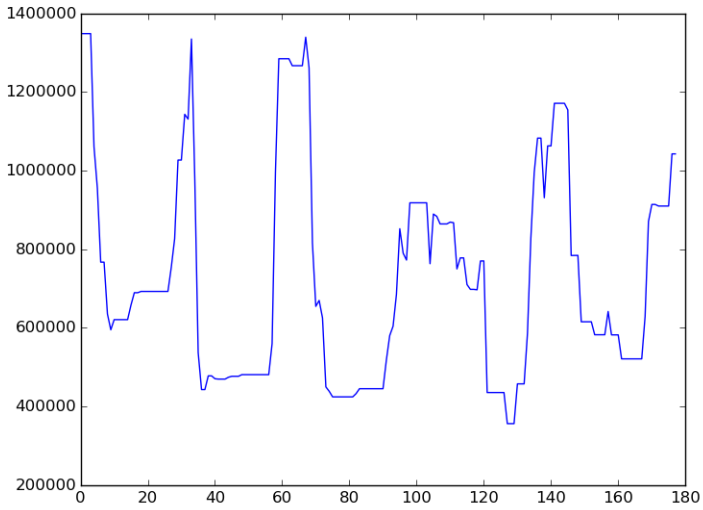
I used tracks 1, 4 and 6.
The columns in the printout below are n, m, dist1(dtw) and dist3(edrp2) from left to right.

```
DTW & EDRP by entire sequences
1 4 441637.5 2220
1 6 448282.500367 1855
4 1 441637.5 2220
4 6 449812.872393 1855
6 1 448282.500367 1855
6 4 449812.872393 1855
```

I believe the values for DTW and EDRP are supposed to be similar, and agree on the order of similarity according to the lecture note. I suspect the EDRP having the wrong order here. I actually wrote two methods for EDRP, and unfortunately the recursive version did not work on the whole sequence because it reached maximum recursion depth. I think I could have checked whether EDRP was correct if both my versions worked by comparing the two answers.

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?

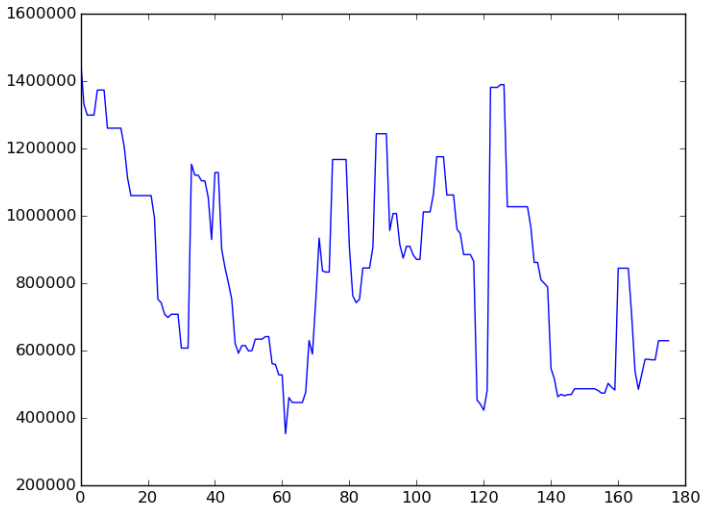
```
DTW by window approach
1 4 127 356269.137281
```



This indicates the phrase "Early in the morning" starts at 127. Unfortunately, it does not seem the maximum corresponds to the mutual sentence.

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?

```
DTW by window approach
4 1 61 352237.407173
```



This indicates the phrase "Early in the morning" starts at 61. Unfortunately, it does not seem the maximum corresponds to the mutual sentence.

I think for wave 1, the maximum should be observed around 65, and for wave 4, 81.