CS5660: Homework 1

Authors: Po-Yih Lee, Wen Li

We work in a pair of two and both contribute equally in the homework.

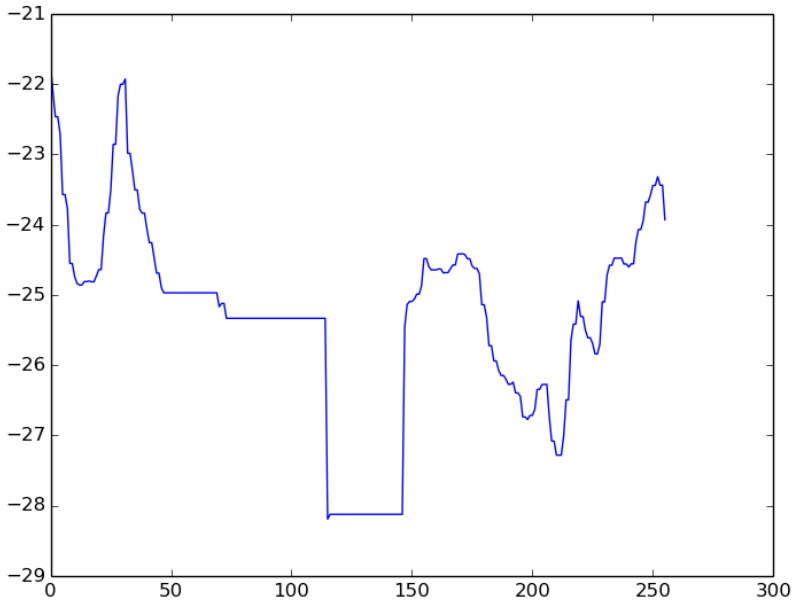
Task 1: Haar Wavelet Transform

Recall the Haar transform from class, which computes averages and differences of larger and larger windows of the signal. We want to use the Haar transform to analyze some real-world signals, so your first task is to implement it.

1. Write a function haar(x) which takes a signal x, given as a sequence of floating-point numbers, and returns the Haar transformed signal. Recall that the Haar transform only works on a signal which has a length of 2^n. For this part of the assignment, you can assume that x has a length which is a power of 2. The transformed result X should have the same length as x.
Please see the source file ./Python/hw1_task1.py
2. Write a function inverse_haar(X) which reverses the Haar transformation. You can also assume that X has a length which is a power of 2.
Please see the source file ./Python/hw1_task1.py

Task 2: Smoothing accelerometer data
A. Haar transformed data

1. Use the function you wrote in Task 1 to transform the data. Recall that Haar requires input which has a length of 2^n, so you should first trim the input to the largest 2^n-sized subsequence.
Please see the source file ./Python/hw1_task2.py
2. Graph the result using graph_accel().



B. Haar transforms and "edges"

Write a function to find the 5 biggest positive entries of the pairwise differences X1, and the 5 most negative entries. Briefly (one or two sentences) describe what is happening to the original signal at the locations corresponding to these points.

The 5 biggest positives:
positive[130] = 0.4264999999999972
positive[185] = 1.4295000000000009
positive[217] = 0.21950000000000003
positive[231] = 0.2270000000000003
positive[255] = 0.24599999999999866

There are significant decreasing at the 5 biggest positives while the actual entry of the five points on the graph is (i - 128) * 2 where i is the index of the haar transform.

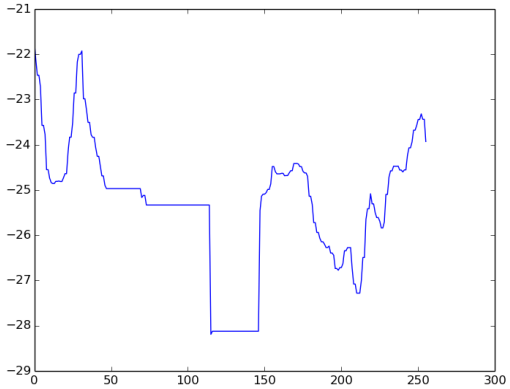
The 5 most negatives:
negative[205] = -1.3354999999999997
negative[205] = -0.19149999999999778
negative[237] = -0.16600000000000037
negative[242] = -0.3014999999999972
negative[243] = -0.19150000000000134

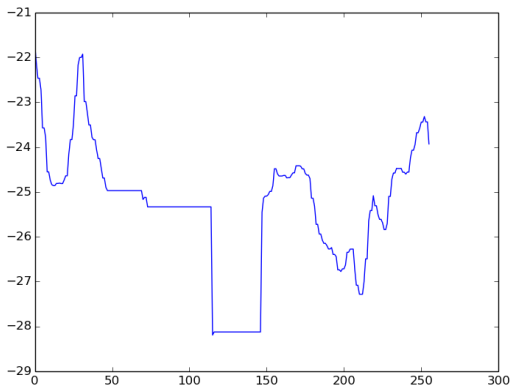
There are significant increasing at the 5 most negatives while the actual entry of the five points on the graph is (i - 128) * 2 where i is the index of the haar transform.

C. Smoothing

1. First, take the original signal x and compute its Haar-transform X. Call inverse_haar(X) and graph the result. Compare the result to the original signal (hint: they should be identical).

The left graph only apply trim function on the original signal, and the right graph is the result get from function Inverse_haar(X) when we input the X. We can see them are identical.

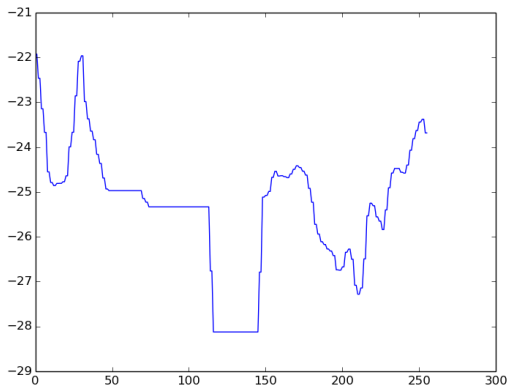
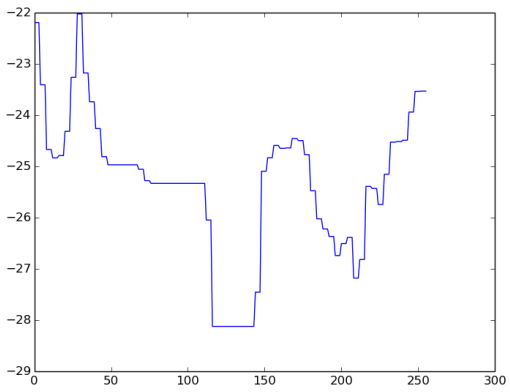
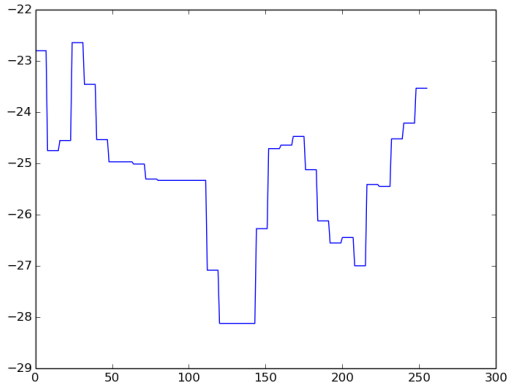
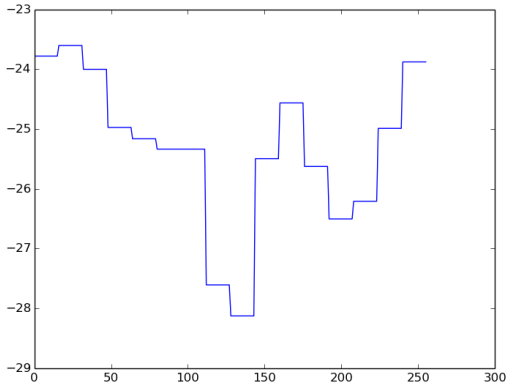




2. Start with the Haar-transformed signal X, and truncate X to the first 2^(n-1) entries. Take the inverse Haar-transform of this truncated X, and graph the result.

Please see the answer to the 3.

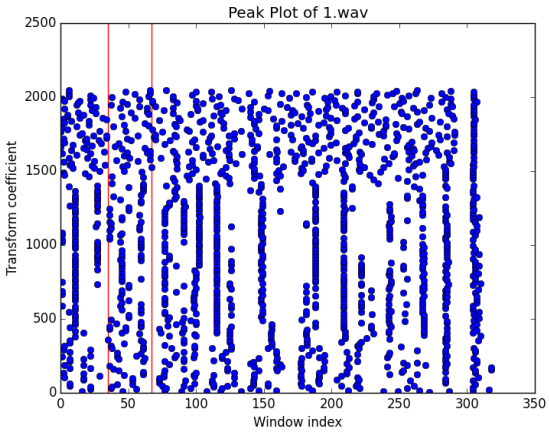
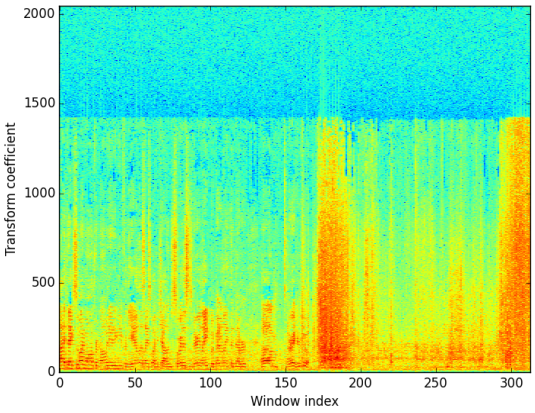
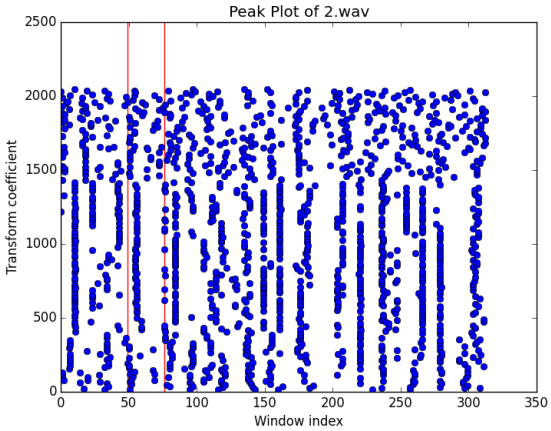
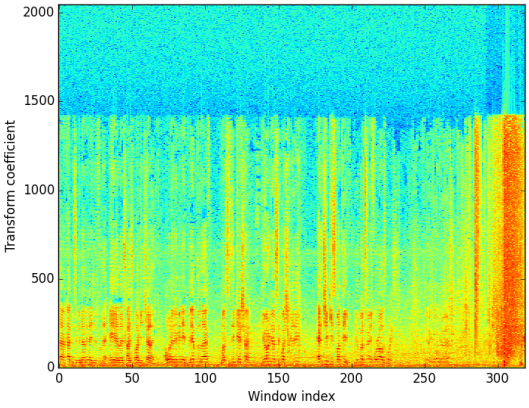
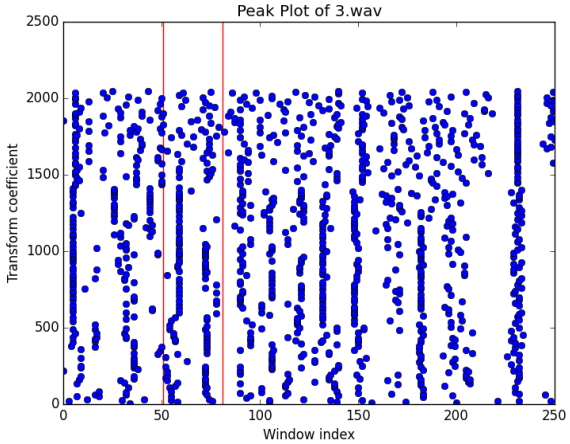
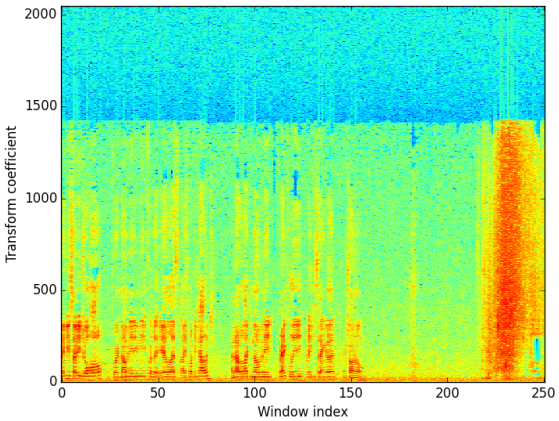
3. Do the same thing with the first 2^(n-k) entries, for k = 2, 3 and 4, graphing each result. Briefly describe (1 to 2 sentences) what happens as k increases.
We can see the signal gradually being “smooth” when the k increase from 1 to 4, and more” sharp

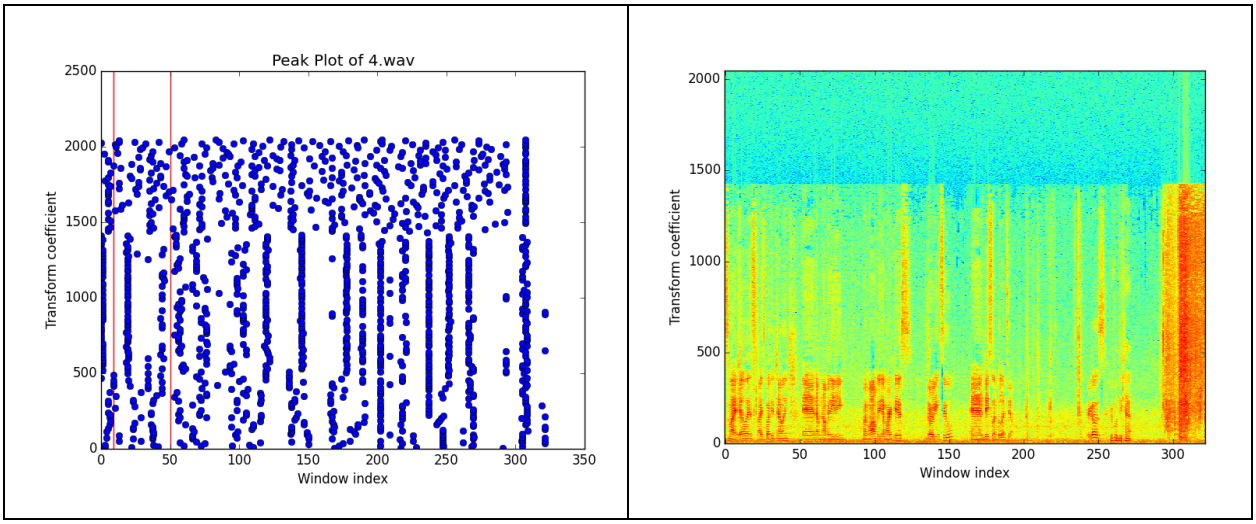
k=1	k=2
	
k=3	k=4
	

Task 3: Aligning Audio Signals
A. Using STFT
1. Write a function which computes all the peaks of a transformed signal X which are the biggest values in a 20 x 20 surrounding grid (K=20).
Please see the source file ./Python/hw1_task3.py

2. Using the function plot_peaks() in the example code provided, plot the peaks and spectrogram of 4 of the provided clips. You can choose yourself which clips, but write their names next to the plots.

3. We have provided for each of the clips the start and end points of the ALS sentence. Mark the start and end points of the sentence on each peaks-map by using the second and third
parameters of the plot_peaks() function.

1. wav	
	
2.wav	
	
3.wav	
	
4.wav	



4. Write a few sentences about the similarity between the peaks maps of the three clips you chose. Do you think they could be aligned? We think the 1.wav, 2.wav, 4.wav should be aligned since their peak map looks similiar and ALT sentences are very close in the almost same position.

5. Build a 1D array y[t] for each audio signal, where y[t]= 0 if there is no peak, and y[t]=peak_value if there is one. If there is more than one, choose one of them.
Please see the source file ./Python/hw1_task3.py

6. Select 3 pairs of clips and use cross-correlation to find the best alignment between the corresponding 1D arrays. How successful was this?
We align the signal pairs in two ways. First, we only aligned the ALS sentence from each clip based on the result of cross-correlation of the STFT transform of the them and get the result below:

1.wav vs 2.wav (ALS sentence)	2.wav should move 3 right sampling points
1.wav vs 3.wav (ALS sentence)	3.wav should move 4 right sampling points
1.wav vs 4.wav (ALS sentence)	4.wav should move 3 right sampling points

next, we align for the whole signal, and the result shows as below:

1.wav vs 2.wav (whole signal)	2.wav should move left 36 sampling points
1.wav vs 3.wav (whole signal)	3.wav should move left 48 sampling points
1.wav vs 4.wav (whole signal)	4.wav should move right 64 sampling points

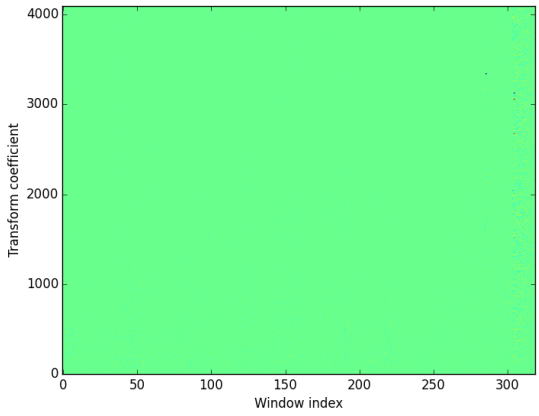
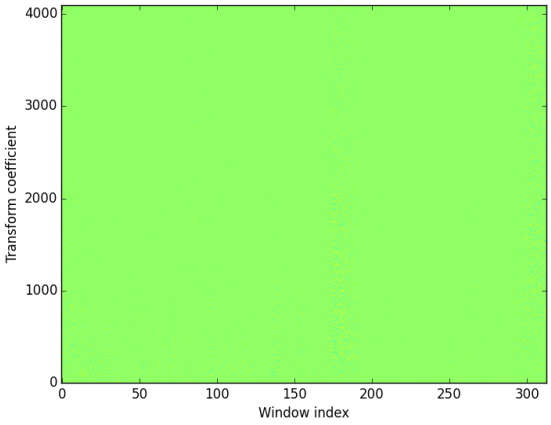
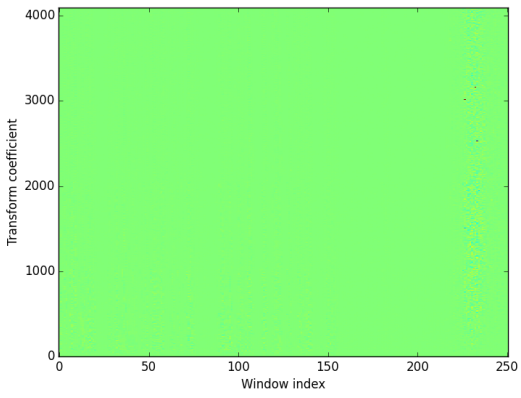
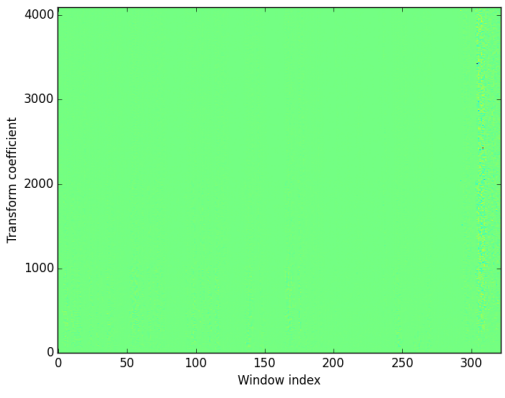
We can see th ALT sentence is very similiar so the results of correlation should be good if we only aligned them. Howerver, if our goal is to make ALT sentence be align based on the whole signal's cross correlation. Ths result is not really statisfying (2.wav should move left 10, 3.wav move left 16, and 4.wav should move left 26.)

B. Using Haar wavelets

Next thing we'll do is try the same thing but this time using Haar wavelets instead of Fourier transform.

1. Write a function short_time_haar(x). This function is analogous to the STFT, in that we break up the signal x into windows. We'll arbitrarily choose a window size of 4096 and a shift of 2048. So, your windows should be x[0:4096], x[2048:2048+4096], [4096:4096+4096],….Trim the signal x to the largest sub-signal that supports this window size. On each window, compute the Haar transform to get a 2D array, where X[t,k] holds the k'th Haar coefficient of the window at time t.

2. For the same 4 clips as above, plot the Short-time Haar transform, using plot_transform.

1. wav	2. wav
	
3. wav	4. wav
	

There are some color points on the graph. But since the density is too small, and we save the plot graph by using very low resolution png file, the difference between the points and the background is not obvious enough. We can provide high resolution result graph if requested.

3. Explain what's going on. Write a few sentences (2-4) explaining why the STFT might be better for analyzing audio signals than the Haar transform, (or maybe you think otherwise?).

First of all, STFT and short time Haar Transform share the sililiar decomposition way to deal with the signal. The STFT transfrom the specific window of original signal into series of Sin function, and the short time Haar Transform use to Haar wavelet to decomposite the original signal. We know genetically Sine and Haar are differents, if we apply Fourier Transform on Squared like function the Gibb's phemonenial will happen. So the questoin will be narrowed down to eigher Sine function or haar function is better for analyzing audio signals. We think the STFT will be better since the human being's ears recevie signal by frequencey, so using sine wave to analyize singal and find the feature in frequency domain makes more sense.

We use haar function to align the there pairs of signal, and we can see the result is also not good.

1.wav vs 2.wav (whole signal)	**2.wav should move left 2 sampling points**
1.wav vs 3.wav (whole signal)	**3.wav should move left 39 sampling points**
1.wav vs 4.wav (whole signal)	**4.wav should move right 6 sampling points**