Modified Patchwork Algorithm:

A Novel Audio Watermarking Scheme

Based on: the same-name paper, written by In-Kwon Yeo, Hyoung Joong Kim, July 2003.

Contents:

1. Introduction

a. What is the Novel Audio Watermarking Designed for

The audio Watermarking Algorithm is a watermarking process applied on Audio signal. It is based on a Watermark designed originally for Image processing, and Image Watermarking. The Image watermarking is applied on the pixel, in the spatial domain. The temporal domain that corresponds for the Audio to the spatial domain for image is not a good domain to apply the watermark to. Actually the temporal domain is very sensible to all the forms of echo, noise addition, down sampling, encoding. That's why the Watermark audio algorithm will be applied in the frequency domain.

The goal, of the Audio Watermarking, is to embed information inside an audio file. The embedded content should be statically undetectable, inaudible, robust again manipulation and secure. Thus to be efficient for monitoring a file, fingerprinting, and to indicate if a content has been manipulated.

b. Work done on it.

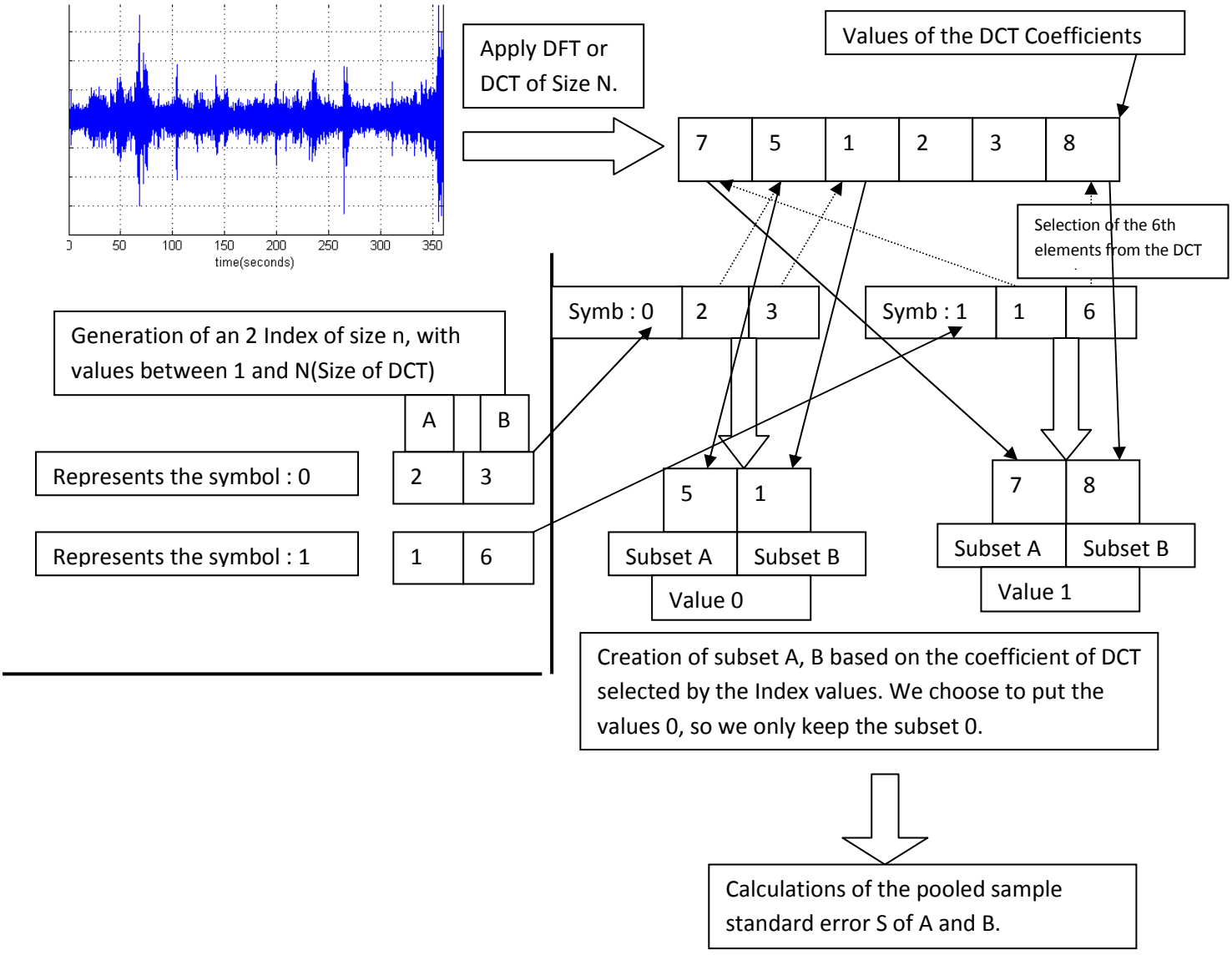
I had several goal in this project that I may have manage to achieve:

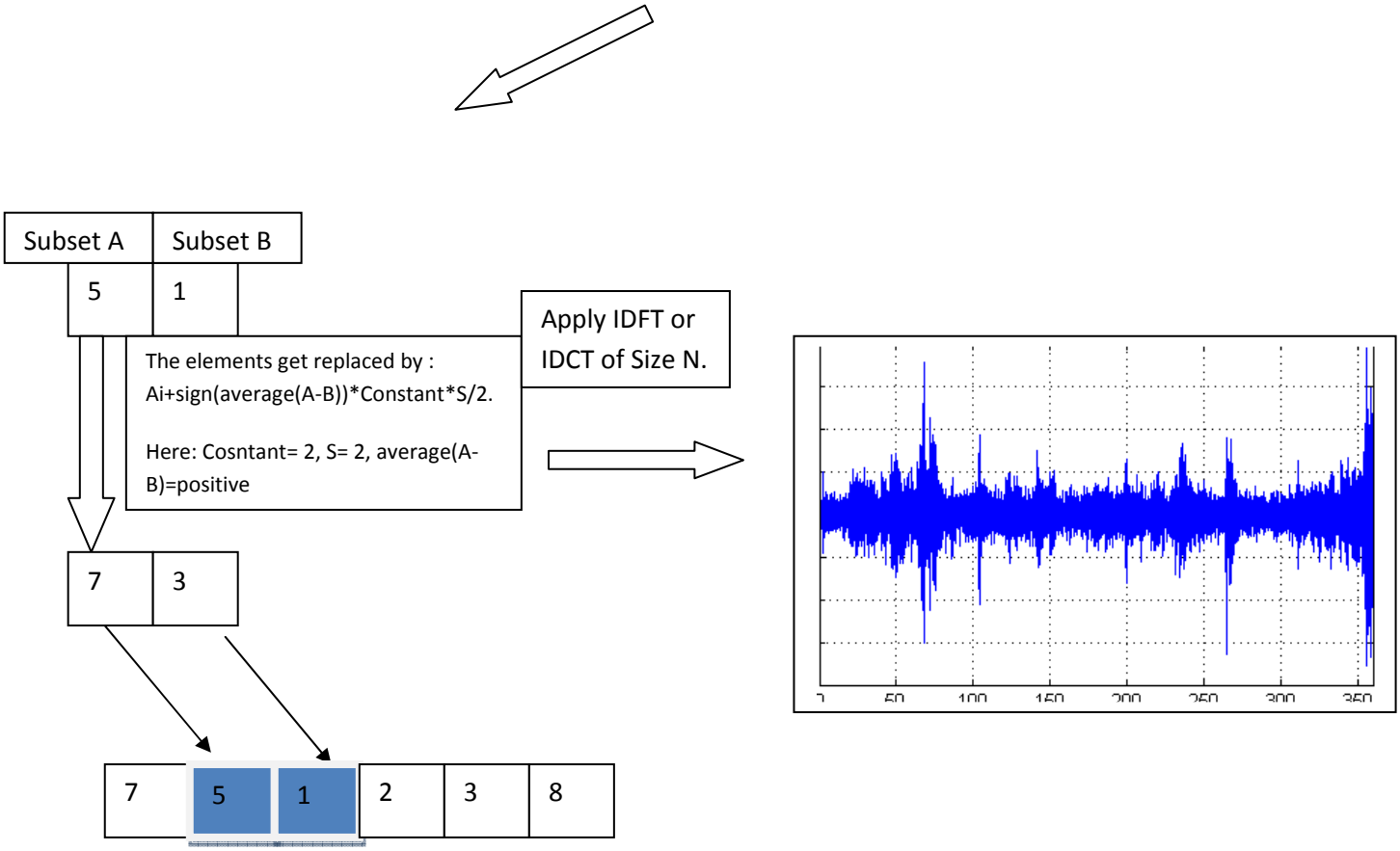
- Understanding what the Watermarking is, his uses and applications.
-Understanding deeply the Modified Patchwork Algorithm is, as well as for the Image as for the audio files.
-Understanding the paper.
-Implement the embedding, and deembedding algorithm
-Verification of the watermarking properties
-Measure how well the algorithm performs
-Give some possibilities on how to make the algorithm better on speech processing.

3. Implementation of the Algorithm.

The algorithm can been summarize like this, but the example below is far easier to understand:

a. General Example of an Encoding.





b. Encoding

```
->Apply a DFT, or DCT of size N on an audio frame an
store the coefficients F.

->From a secret key, generate 2 Index with 2 n values
pseudo-randomly chosen between 1 and N. Each index
will represented 0 or 1.

->Define the subset A and B, with A=[ F coefficients with
subset equals to the first n elements of the Index of the
desired values] . Same for B with the last n elements.

->Calculate the mean and the pooled sample standard error
S of elements of A and B.

->Replace them by Ai*=Ai + sign(average(A-B))* C * S/2
      C is a constant
      Make the large value set larger, and small value set smaller
      The distance between two sample is always bigger than
square(C ) *S

->Apply the inverse DCT. The signal is watermarked.
```

Understanding the algorithm, and translating it in pseudo code is not easy. But once it has been done, the implementation is straightforward. I have been using Matlab.

One of the biggest difficulties I had, was with generating a pseudo random vector based on a key to generate the Index for each symbol. A lot of popular function exists, like md5 for example. But they are all hash function that accept an integer and return a key. A hash function works in one way, but not in the reverse way. It means that the function can output the key based on the integer inputted (seed) but cannot output the integer based on the key. It is a basic security principle. Moreover the key should be randomly linked to the integer. No correlation should be found between the key and the integer. For example, if I increment the input integer of one, my key should not be

less similar to original key (with no incremented input integer), that if I would have incremented the input integer of 100. Some function can be found as a library in Matlab. But none of these output pseudo random Vector based on a key. So I had to build such a function.

To generate a Vector with pseudo random values comprised between 1 and n, I have been using my own function (genPseudoVector) based on Matlab function md5. The algorithm was developed empirically, checking at each step that the function was respecting the assumptions of a hash function (no reverse processing possible, and no correlation between the seed and the key). The algorithm is not easy to describe in such a short report. But the main idea behind it is to use the function md5 several times based on a seed which is based on the previous key but randomly.

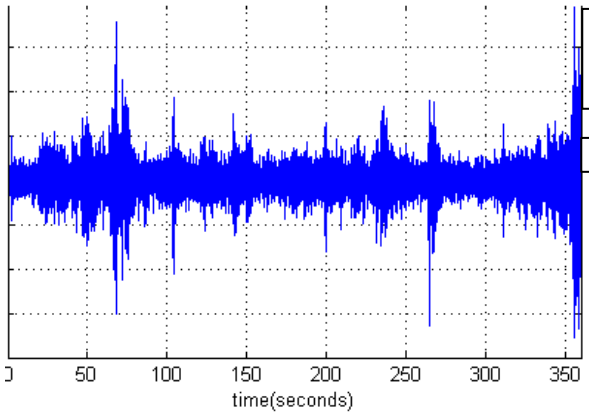
The then implementation of the embedding function was quite easier once the algorithm understand although that not easy to debug.

The signification of the equations to determine the new value is the following: The sign function enables to know which subset (A or B) we will reduce, and which one we will increase. Actually, it seems logical to increase the subset with the higher average, and decrease the other in order to increase the distance between the two subset, and thus affording us to have a more robust code. S is used to make the difference between the two new subsets proportional to the original difference to make the watermarking not detectable.

c. Decoding

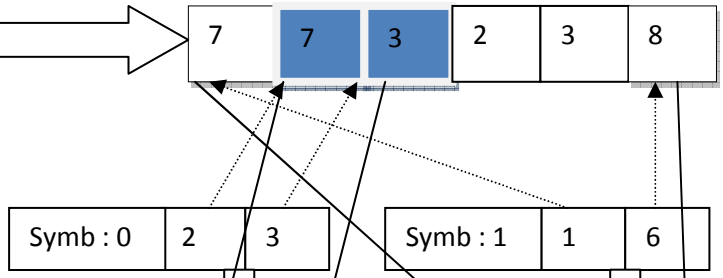
The decoding function is quite simpler and easier that the embedding function. But the flow chart is a good help to understand it.

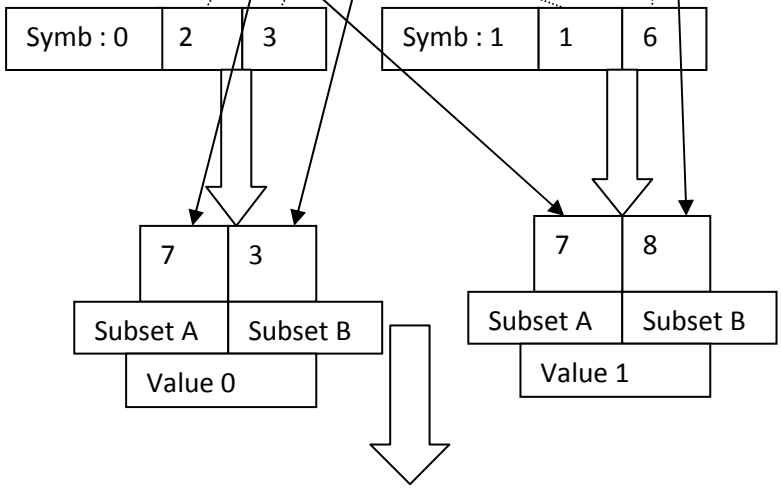
```
-Retrieve the modified DFT,DCT coefficient for the Index
representing 0, and 1. A0 and B0 and A1 and B1.
-Compute the sample means and pooled sample error
(S0,S1) for A0,B0…
- See what for what set we get the biggest values T:
- T0= (ave(A0)-ave(B0))² /S²0 or T1= (ave(A1)-ave(B1))² /S²1.
- If T is bigger than a predecided threshold then we assume
that the signal is watermarked.
```



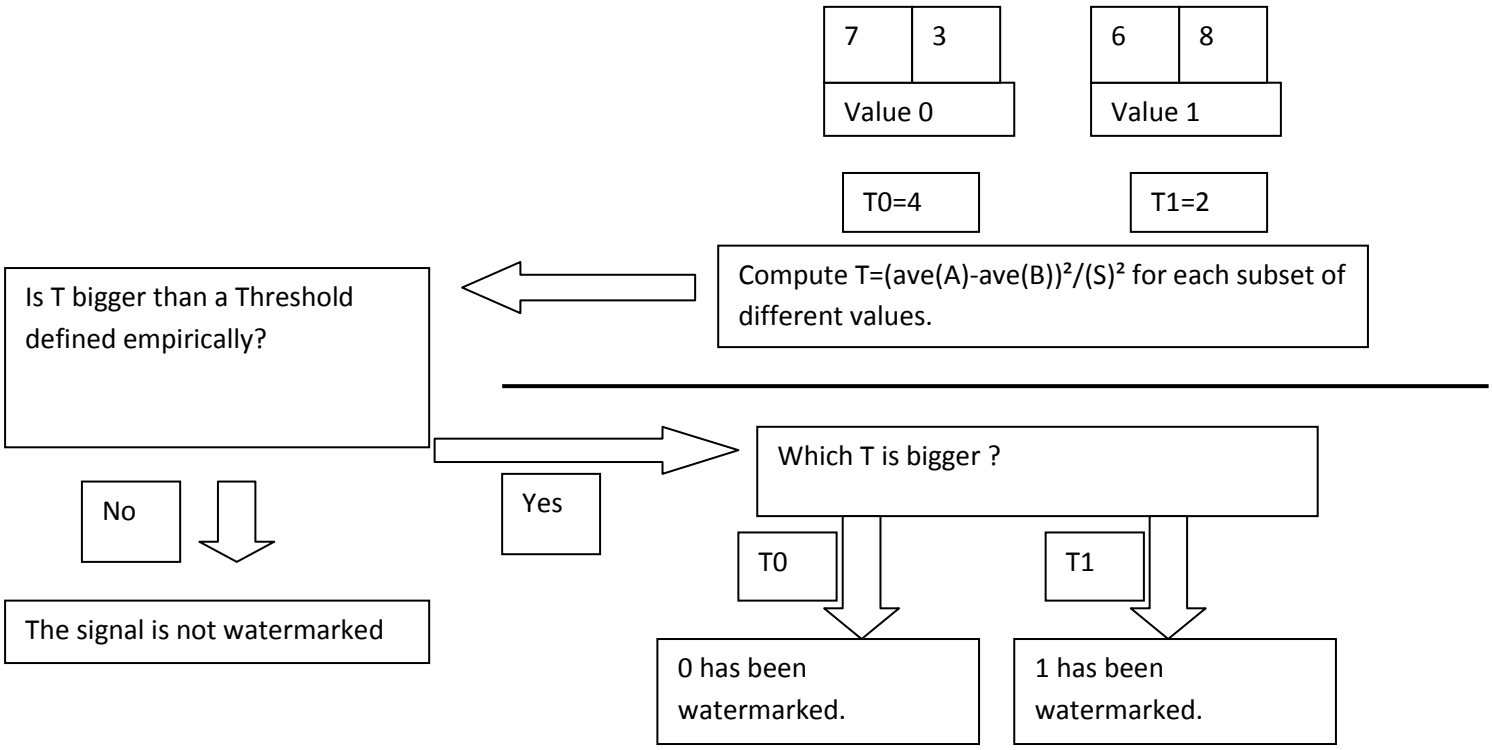
Apply DFT or DCT of Size N.

The algorithm don't know what symbol has been assigned, but it knows the key. It would then extract the information for the two Index set, and see which is the right one.





Do the inverse of the embedding algorithm. Look which data have been modified.



Note: this example is not supposed to work as the numbers of samples is too small and statistically insignificant. To be significant we should have a large amount of values for the samples of each subset A and B.

3. Verification of the Basic Properties.

a. Inaudible

The Inaudibility of the watermark depends of the value of the constant C used to write the watermark. The bigger C, the more likely to be heard the song will be. It is a simply proportionality relation. When the song is heard, it is heard at different specific frequencies depending on the frequencies of the Index. When C is big, you can hear a noise which sounds like a addition of very pure harmonic.

b. Statistically undetectable

To verify this properties I have tried to use all the basic tools we have used or discovered during the lecture. Including Fourier transform, wavelet, cepstral processing, LPC, correlation, auto correlation (in time, and frequencies) on the signal itself or on some frame of the signal. But has the frequencies modified are randomly chosen it is impossible to detect anything. Therefore the signal seems with the few tools I know undetectable without knowing the original signal.

d. Secure

The security of the signal depends directly upon its undetectability. Actually if the signal Is detected it means that the embedded information could be recovered and then reinserted in the audio files after modification. So unless we know the key, the signal is not detectable and therefore secure.

3. Performance of the Audio Watermarking.

To measure the performance of the audio Watermarking, we have used several modification of the song, related to the work done in Digital Speech processing. Three types of errors can occur:
- a watermarked is detected although the file has not been watermarked.
-a watermarked is not detected although the signal has been watermarked. (Called Error A)
-the signal has been watermarked but the wrong data Is get from the signal. (Called Error B)
We will only be interested on the two last errors, are they are the ones sensible to signal modification. The test have been done on several radio files (the simplest and cleanest audio files have been able to get).

a. Pitch Modification

Modification (unit: tone)	Error A	Error B
Pitch +1	96%	49%
Pitch +2	98%	56%
Pitch -1	94%	47%
Pitch -2	99%	53%

As we can see the watermarked is not efficient at all against pitch modification. This could be easily explained by the fact that the watermarking algorithm input its data in the frequencies domain. And this is this domain that is modified by the watermark.

b .Encoding

Encoding qualities (MP3) (in Kbits)	Error A	Error B
256	0%	0%
128	0%	0%
64	2%	0%
32	13%	6%

The better the quality of the recording is, the less likely the watermark will be destroyed. This results show two things: -the watermark algorithm does well against encoding, with a relatively rate of error.

-the MP3 is a high fidelity encoding, as the watermarked is not detoriated for high bitrates.

c. Echo addition

Echo addition	Error A	Error B
Attenuation coeff= 0.60	31%	6%
Attenuation coeff= 0.30	14%	3%
Attenuation coeff= 0.10	3%	1%

The results seem quite logical, the higher the attenuation coefficient is, the more error we will get, as more DFT or DCT coefficients are modified. Nevertheless for a common and real echo (0,10), the algorithm succeeds in keeping the watermarked information.

4. How to perform better

a. Silent detections

A simple addition I have done, is detecting when the speaker is not speaking (i.e.: when the energy in the signal, is lower than 5 times the mean energy in the entire signal). The robustness of the algorithm is then better especially with MP3 encoding at low bitrates (reduces from 13 to 7%).

b.Against pitch modification.

One simple thing can be done to almost annihilate all the effects from the pitch modification. We just have to detect how much the pitch has been changed (using our pitch detection algorithm in HW3) , and change it back to the original value. Then the error rate tends to zero.

5. Conclusion

In this work I have been able to manipulate the audio speech signal and discovering the watermarking and his possibilities and uses. Moreover I have been able to directly link it to the lecture by using several analyzing tools (fourrier, auto correlation, pitch estimation, ….).