[CMPE 362 – Spring 2016]

[Homework 4]

[Boğaziçi University]

Enes Özipek
2012400162

In this homework, we are asked to design a watermarking system and to implement the system on time domain, wavelet domain and frequency domain respectively.

1) Watermark on Time Domain

In this part, watermark image will embedded on time domain. Steps are preprocessing, watermark embedding and watermark extraction.

1.1) Preprocessing

- First read image , convert it into one dimensional array.
- Convert one dim. Array to byte array.
- Create bits array to hold bits to be embedded.
- Create hopping sequence and read sound file.
- Convert sound array to byte array. This is our *audioBytes* array.

1.2) Watermark Embedding

- Make for loop with the length of bits array holding image bits
- Obtain embed location by utilizing hopping sequence.
- Take the bit from bit array.
- If bit value is 1 , use bitor function given in pdf.
- Else use bitand function given in pdf.
- Within this loop modify *audioBytes*

1.3) Watermark Extraction

- Make loop with the bits array
- Take location from hopping sequence
- Take the bit from bit array.
- Stores all bits embedded to the sound before in a different array.
- These bits constructs our first image.
- Make mat2gray after array manipulations.
- Last step is to calculate SNR value calculation with which we are familiar from previous homework.

RESULTS:

SNR : 56.338944 MOS: 4

2) Watermark on Wavelet Domain

In this part, watermark image will embedded on wavelet domain. Steps are preprocessing, watermark embedding and watermark extraction.

2.1) Preprocessing

- First read image , convert it into one dimensional array.
- Convert one dim. Array to byte array.
- Create bits array to hold bits to be embedded.
- Create hopping sequence and read sound file.
- Convert sound array to byte array. This is our *audioBytes* array.
- Create L2, H2, H1 and fill according to DWT algorithm.
- Lastly concatenate them.

2.2) Watermark Embedding

- Make for loop with the length of concatenated array.
- Obtain embed location by utilizing hopping sequence.
- Take the bit from bit array.
- If bit value is 1 , use bitor function given in pdf.
- Else use bitand function given in pdf.
- Within this loop modify concatenated array

2.3) Watermark Extraction

- First make inverse DWT to after watermark embedding.
- Again create new L1,L2,H1,H2.
- Fill them with correct values.
- Enter to the loop with length of bit array.
- Take location from hopping sequence
- Take the bit from bit array.
- Stores all bits embedded to the sound before in a different array.
- These bits constructs our first image.
- Make mat2gray after array manipulations.
- Last step is to calculate SNR value calculation with which we are familiar from previous homework.

RESULTS:

SNR : 2,7894 MOS: 1

3) Watermark on Frequency Domain

In this part, watermark image will embedded on frequency domain. Steps are preprocessing and watermark embedding.

3.1) Preprocessing

- First read image , convert it into one dimensional array.
- Read sound file.
- Apply fft to sound array.

3.2) Watermark Embedding → MY ALGORITHM

- Create 20k bins.
- First aim is to find 5 peaks from each frame.
- Take and store indices of those peak values.
- Then embed image bits to those indices.
- Make ifft
- Lastly, calculate SNR.

RESULTS:

SNR : 206.894506 MOS: 5

3.3) Watermark Embedding → SIMPLEST ALGORITHM

- Embed bits of images to the highest points.
- For instance between 0-100Hz and 7200-10kHz are very good choice to embed the bits.
- Make ifft
- Lastly, calculate SNR.

RESULTS:

SNR : 65.212010 MOS: 4

Comparison between my algorithm and simplest algorithm

A watermark must not be embedded into insignificant region of the audio signal. Since, for instance, if a watermark is embedded in the high frequency spectrum of an audio signal, low-pass filtering can easily eliminate the watermark. Thus, it is vital to find appropriate regions to embed a watermark.

Therefore, I used the technique of embedding bits into the peak values. Those peak values are from non-overlapping frames which are divided according to bin size.

```
for i=1:N/bins
    [pxx_peaks,location]=findpeaks(real(y((i-
1)*bins+1:i*bins)),'NPEAKS',5);
    for j=1:5
    peakBits(n) = location(j);
    n = n + 1;
    end
end
% Assign watermark bits to the peak values
for i=1:4060
   y(peakBits(i))=concatImage(i);
end
% Assign watermark bits to the peak values
for i=1:36
    y(peakBits(4060)+i)=concatImage(4060+i);
end
```

When we compare the snr values of simplest algorithm and my algorithm, we see that they differ by 140. And quality of the wav file is better. And moreover, it is more safer if we consider watermark extraction.

Comments on MOS and SNR values

When I consider all results I've taken from questions , I realize that MOS values and SNR values show parallelism. In other words, If SNR value is higher then the quality of the sound is better. From the last question, when I compare the algorithms, it's also reasonable to assert that results come up as expected. Since there is consistency for overall calculations. And also considering the fact that my algorithm improves SNR value as compared to the SNR value simplest algorithm produces, I expect quality will be better. Then when I listen to sound again , I see that it supports my assert which means MOS value increases.