Indian Institute of Technology Delhi

ELL786 Multimedia Systems: Assignment 1

Deepak: 2019MT10685

Part 1 Experiment 1

Method

In this part, we have converted a .txt file into a binary string of length M. Then we generated a a random binary error pattern of length M with hamming weight d such that the non-zero entries are uniformly distributed across M bits. After that we XOR the above error pattern with the message bits. This resulted in noise in the message. Then we decoded the binary string to text string and calculated the error(i.e., percentage of modified characters with respect to the input file). We then repeated the same experiment with different values of d.

Observations

- As we increase d, the percentage of modified characters with respect to the input file increases. This is quite obvious since on increasing d, we are inducing more noise and therefore more error.
- On taking the XOR operation, if the bit value at index i of random binary error pattern is 0, the it will not change the bit value of original binary string but if it is 1, then it will flip the bit value in original binary string. As d is number of 1s in random binary error pattern. Therefore on increasing d, there will be more 1s in random error pattern and this in turn will flip more bits in original binary string. Therefore, on increasing d, error increases.

Part 1 Experiment 2

Method

In this part, we repeated Experiment 1 with some modifications in it. After converting the .txt file intro binary string, we compressed the binary string using various coding techniques: 1) Huffman Coding, 2) Extended Huffman Coding and 3) Arithmetic Coding. After compressing the data, we induced some noise by using the same method of experiment 1 and then analysed the error with respect to different values of d.

Observations

1) Huffman Coding

- In the binary string there were only 2 letters(0 and 1) in the alphabet. Therefore, after calculating frequency of both the letters and applying Huffman Encoding technique, one letter was encoded as '0' and the other letter was encoded as 1. Therefore, no compression overall.
- Since there was no compression, the error with respect to d was same as Experiment 1. Error increased on increasing the value of d (as this would result in more noise in data).

2) Extended Huffman Coding

- In the Extended Huffman coding, we used n-bit per symbol. therefore there were total n^2 letters. We then calculated the frequency of each letter via iterating over the string and the applied Huffman coding using the greedy approach via Huffman Tree. This resulted in compression of binary string. The compression was 9.74% at n=4 and it increased on increasing n (this is because on increasing number of bits per symbol, many letters will have 0 frequency).

- Since the dictionary may not contain all the letters after inducing noise, we couldn't decode the binary string with noise and therefore cannot compute the exact error. However, we can intuitively say that the error increased on increasing the value of d.

3) Arithmetic Coding

- In the Arithmetic coding, we used n-bit per symbol. therefore there were total n^2 letters. We the calculated the probability and cumulative probability of each letter and applied the Arithmetic coding technique to compress the data. The compression was 11.63% at n=5 and it increased on increasing n (this is because on increasing number of bits per symbol, many letters will have 0 frequency)

- Since the dictionary may not contain all the letters after inducing noise, we couldn't decode the binary string with noise and therefore cannot compute the exact error. However, we can intuitively say that the error increased on increasing the value of d.

Part 2 Experiment 1

In this part, we applied Discrete Wavelet Transform on a grey-scale png image of dimension 512x512 and analysed how much compression can be achieved without a prominent loss. After taking the Take the 2-D DWT of the image, we computed the average energy of each sub-band. After that we we are going to "pruned" some of the coefficients in each sub-band by keeping those with the highest energy and selecting a sub-band specific threshold and discarding the rest. We experimented this with different thresholds and level of decomposition. We also varied the number of coefficients retained and analysed the result

Observations

Variiying Threshold

On decreasing the threshold, image quality increases.

Image at threshold = average :

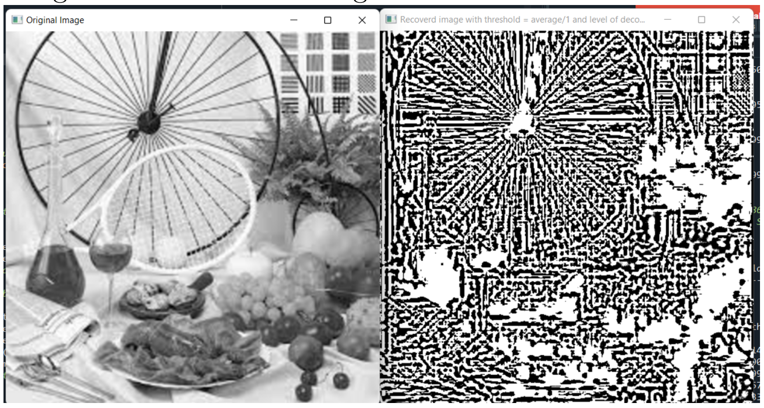
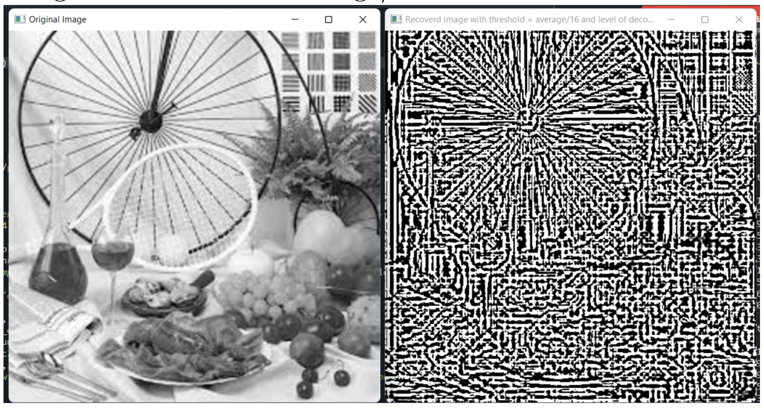

Original Image
Recoverd image with threshold = average/1 and level of deco...

Image at threshold = average/16 :


Original Image
Recoverd image with threshold = average/16 and level of deco...

Variying Level of decomposition

On increasing the level of decomposition, image quality decreases.

Image at level of decomposition = 3

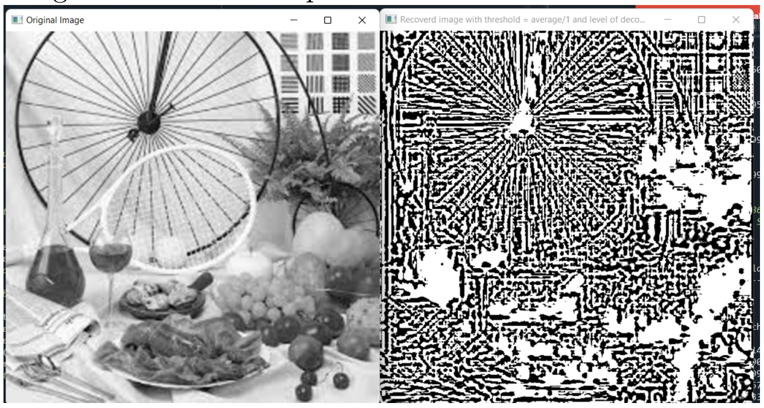
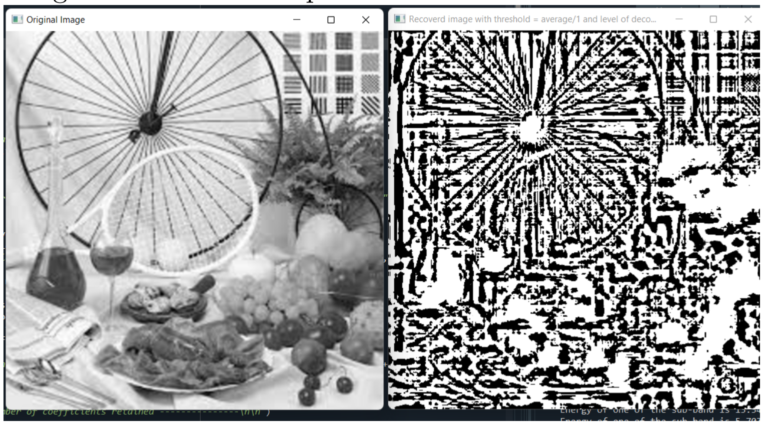

Original Image
Recoverd image with threshold = average/1 and level of deco...

Image at level of decomposition = 4


Original Image
Recoverd image with threshold = average/1 and level of deco...

Variiying number of coefficients retained

On decreasing the number of coefficients retained, image quality decreases.

Image at number of coefficients retained = 50 percent

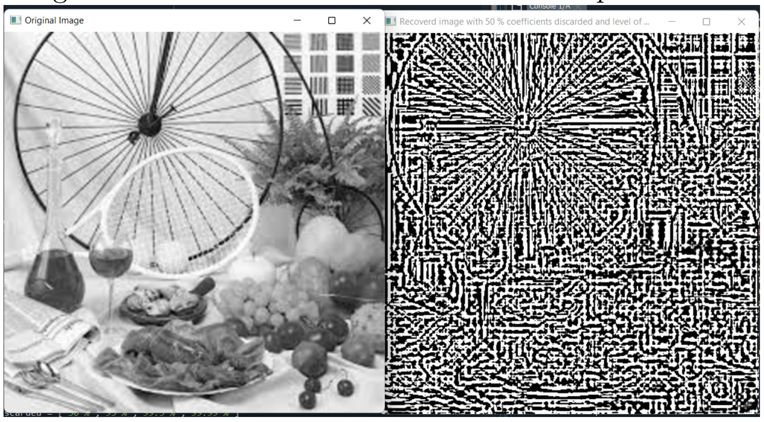
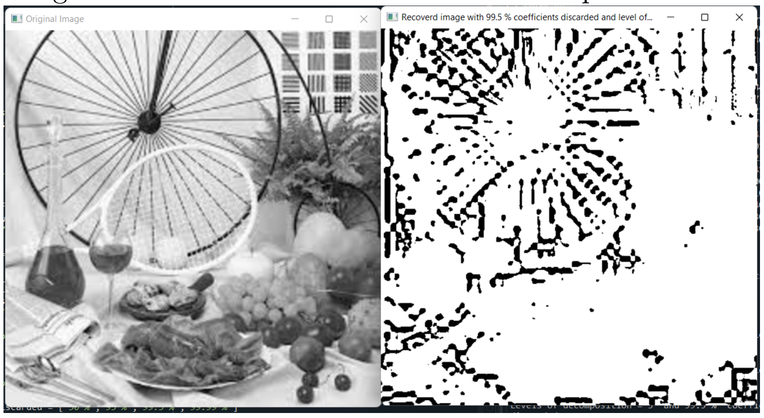

Original Image
Recoverd image with 50 % coefficients discarded and level of ...

Image at number of coefficients retained = 5 percent


Original Image
Recoverd image with 99.5 % coefficients discarded and level of...