**Project Title:  Image Processing Fundamentals**

**Project Number**: Project 6

**Course Number**: CEG 7850-01

**Student's Name**: Alex Reigle

**Date Due**: November 4, 2021\*

**Date Submitted**:  November 3, 2021

**Declaration Statement:**

I hereby declare that this Report and the Matlab codes were written/prepared entirely by me based on my own work, and I have not used any material from another Project at another department/ university/college anywhere else, including Wright State. I also declare that I did not seek or receive assistance from any other person and I did not help any other person to prepare their reports or code.  The report mentions explicitly all sources of information in the reference list. I am aware of the fact that violation of these clauses is regarded as cheating and can result in invalidation of the paper with zero grade. Cheating or attempted cheating or assistance in cheating is reportable to the appropriate authority and may result in the expulsion of the student, in accordance with the University and College Policies.

**Abstract:**

In this project I developed a function to compute the entropy of an image. I used this function to recreate the entropy calculations of Figures 8.1(a-c) in the textbook as well as the 8-bit image from textbook problem 8.9(a). In completing problems 8.9 and 8.10 from the textbook, I hand tabulated and demonstrate the Huffman encoding process and show the compression ratio and comment on the efficiency of the compression process. I then complete this again using pixel pairs and the differences between adjacent pixels. To complete 8.10 I decode the symbols for the associated encoded string using the key provided in Figure 8.8 of the textbook. I then write a program to quantify the losses in Fourier and Cosine coding for a varying number of N-largest bit allocation values, and comment on the relationship. Finally, I compress the image from Figure 8.9(a) in the text using a Haar wavelet transform technique and quantify the RMS and SNR.

**Technical Discussion:**

This project uses the MatLab built-in transformation functions to achieve fast-Fourier, cosine, and Haar wavelet transforms. However, the entropy calculation function was developed for this project by making use of the textbook equation (8-7):

In addition, the following textbook definition of compression ration was used to quantify the effectiveness of the various image compression techniques: (8-2)

To study the effects in varying losses from bit allocation methodologies, Fourier and cosine transform effectiveness was quantified using a developed root mean square error algorithm base on the following textbook equation (8-10)

Additionally, the effects were quantified with a function developed to calculate the signal to noise ratio of an image using the textbook equation (8-11), as follows:

In developing a method to compress the image from Fig. 8.9(a) from the text, I stumbled across two methods of computation. The first is the most computationally efficient and involves zeroing out the values in the ‘S’ vector (returned by the MatLab wavedec2 function) which are used to index values and locations of the non-approximation coefficient matrices. The latter is less efficient, but I chose to include it because it is more intuitive and shows the iterative reconstruction process can be completed by explicitly providing zero matrices in place of the coefficient matrices.

**Results:**

Problem 1 implemented eqn. (8-7) from the text to automatically calculate the entropy of each image passed. The first of three images passed into the function was the image from Figure 8.1(a), shown below in Fig. 1, which is calculated as having an entropy of 1.6614 bits/pixel.



Figure 1: Image from Figure 8.1(a) of page 541 in the text. The textbook reports the image to have an entropy of 1.6614 bits/pixel.

The second image (shown in Fig. 2) and third image (shown in Fig. 3) from the textbook were then processed using the same method to calculate an entropy of 8 bits/pixel and 1.566 bits/pixel, respectively.



Figure 2: Image from Figure 8.1(b) of page 541 in the text. The textbook reports the image to have an entropy of 8 bits/pixel.



Figure 3: Image from Figure 8.1(c) of page 541 in the text. The textbook reports the image to have an entropy of 1.566 bits/pixel.

Problem 2 considers the image described in textbook problem 8.9 which is shown in Table 1.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 21 | 21 | 21 | 95 | 169 | 243 | 243 | 243 |
| 21 | 21 | 21 | 95 | 169 | 243 | 243 | 243 |
| 21 | 21 | 21 | 95 | 169 | 243 | 243 | 243 |
| 21 | 21 | 21 | 95 | 169 | 243 | 243 | 243 |

Table 1: A simple 4x8 bit image, with a calculated entropy of 1.8113 bits/pixel.

Then entropy of the image is calculated to be 1.8113 bits/pixel. The encoding process is shown in Fig. 3, below, and results in the unique code symbols shown in Figure 4.

Diagram

Description automatically generated

Figure 4: The encoding process underwent to calculate the code symbols resulting from the Huffman coding process.

Diagram

Description automatically generated

Figure 5: The code symbol key resulting from the Huffman coding process.

The compressed image string is shown below in Fig. 6, and shows that the resultant image is a total of 60 bits, as opposed to the original image having 216 total bits after being converted to a binary string. This process shows a compression ratio of 3.6.

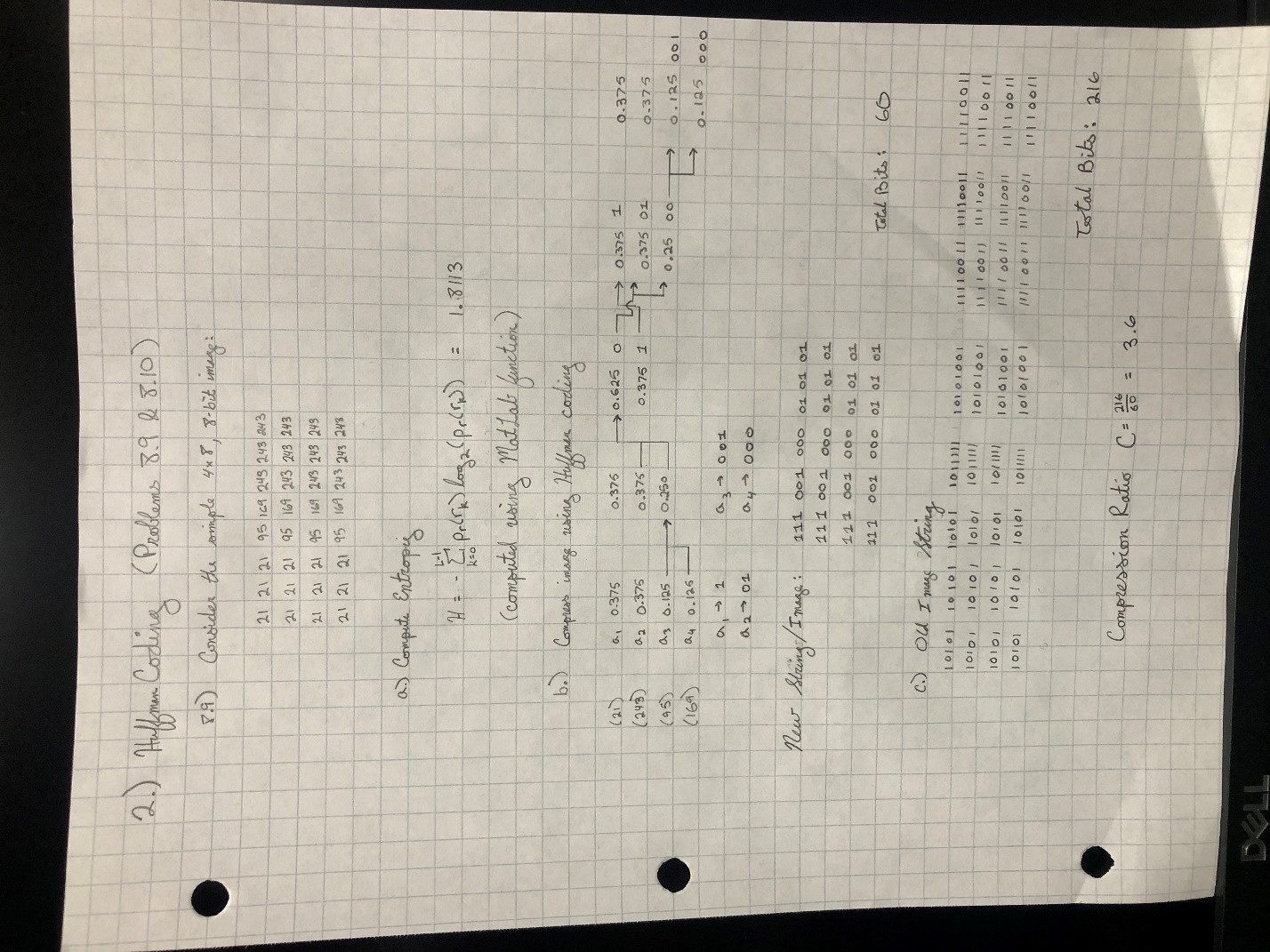


Figure 6: The binary string resulting from the Huffman compression of the image in Table 1.

Considering the Huffman encoding of pixel pairs, all code symbols have the same event probability of 0.25, resulting in a higher image entropy of 2 bits/pixel. Alternatively, the case of pixel differences results in a lower entropy of 1.5073 bits/pixel. This is expected and discussed further below.

The answer to textbook question 8.10 is as follows:

a3 a6 a6 a2 a5 a2 a2 a2 a4

Problem 3 tasked me with computing the root-mean-square (RMS) error and the mean signal-to-noise ratio (SNR) for the image in Figure 8.9(a) from the text. These values were computed for both Fourier and Cosine transformations at an N-largest coding bit allocation, with N ranging from 1 to 8. The results are captured in Table 2.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Bit Allocation | N-largest Bit | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| Fourier | RMS | 15.1949 | 14.5466 | 13.9211 | 13.3428 | 12.8006 | 12.252 | 11.821 | 0.1453 |
|  | SNR | 8.3456\*10^-36 | 1.0782\*10^-33 | 2.9129\*10^-34 | 2.5453\*10^-33 | 7.0798\*10^-33 | 7.3124\*10^-33 | 8.9315\*10^-33 | 1.3902\*10^11 |
| Cosine | RMS | 11.5227 | 11.1708 | 11.3224 | 11.6277 | 11.8596 | 11.8018 | 11.821 | 0.1453 |
|  | SNR | 3.0729\*10^-34 | 7.1317\*10^-33 | 2.0568\*10^-32 | 1.6136-10^-34 | 1.1460\*10^-32 | 1.2282\*10^-32 | 2.0355\*10^-31 | 1.3902\*10^11 |

Table 2: The tabulation of Fourier and Cosine transform SNR and RMS error for each N value 1-8.

It is clear that the 8-Largest coding bit allocation is the smallest value of N that is acceptable, as for all other values of N the SNR drops to almost zero.

Problem 4 shows the image from Figure 8.9(a) in the text being compressed using a Haar wavelet transform at scales of J=1,2,3, while zeroing out the coefficient matrices that were not the approximation coefficient matrices. Where the compression performance of each image is equal to 2J for each image, as only 1/2J pixels (the approximation coefficient matrix) are not set to zero. The results of this compression are shown in Fig. 7, below.

A person wearing a hat

Description automatically generated with low confidence

Figure 7: The Haar wavelet transform of each J=1,2,3 scale had all but the approximation coefficient matrix set to zero prior to the inverse transform. The resultant compression is shown.

The RMS error and SNR of the scale J=1 compression is 0.0078 and 4.8084x1013, respectively. The RMS error and SNR of the scale J=2 compression is 0.0841 and 4.148x1011, respectively. The RMS error and SNR of the scale J=3 compression is 0.8473 and 4.088x109, respectively. This shows a decrease in quality of image with the increase in compression ratio, as expected.

**Discussion of Results:**

Problem 1 shows results that are consistent with that of the textbook, confirming the calculations of the computeEntropy function.

Problem 2 shows the relationship between compression ratio and entropy to be directly proportional. This makes intuitive sense, because entropy is a measure of the independence of information in an image and compression is a measure of reducing a large amount of redundancy. This explains the differences in entropy between the three methods of compression from problem 2. The most successful compression (in terms of reducing the image to the shortest bit string) corresponds the pair pixel method where the image is effectively halved in size and each code symbol is equally likely. The equal likelihood of each event shows the image information is very independent, and therefore the entropy is higher. However, the Huffman coding by difference results in a higher entropy because it has some symbols that are more likely and some that are less so.

Problem 3 shows an inversely proportional relationship between RMS error and SNR, which is logical since RMS error is the denominator in the definition of SNR. The results also show that the 8-bit coding is about as low of a value for the bit allocation as can be done while maintaining good results. For better results the value can be increased, but this reduces the compression ratio and therefore computational efficiency of image transfer.

Problem 4 shows how well suited for image compression wavelet transforms are. The image can be easily compression by some scale power of 2, depending on the compression needs. The greatest advantage of wavelet transforms is that there is no requirement of a unique code symbol key, as there is in Huffman coding-from problem 2. This makes it easier to access the images across lower bandwidth networks will maintaining the important parts of an image, video, or document. However, the draw to Huffman coding is two-fold. The primary advantage of Huffman coding is that it is lossless, whereas wavelet compression is very lossy. The second advantage of Huffman coding is the requirement of a symbol code key which acts as a simple encryption and could be advantageous to some.

**Figures:**

Background pattern

Description automatically generated

Figure 1: Image from Figure 8.1(a) of page 541 in the text. The textbook reports the image to have an entropy of 1.6614 bits/pixel.

Background pattern

Description automatically generated

Figure 2: Image from Figure 8.1(b) of page 541 in the text. The textbook reports the image to have an entropy of 8 bits/pixel.



Figure 3: Image from Figure 8.1(c) of page 541 in the text. The textbook reports the image to have an entropy of 1.566 bits/pixel.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 21 | 21 | 21 | 95 | 169 | 243 | 243 | 243 |
| 21 | 21 | 21 | 95 | 169 | 243 | 243 | 243 |
| 21 | 21 | 21 | 95 | 169 | 243 | 243 | 243 |
| 21 | 21 | 21 | 95 | 169 | 243 | 243 | 243 |

Table 1: A simple 4x8 bit image, with a calculated entropy of 1.8113 bits/pixel.

Diagram

Description automatically generated

Figure 4: The encoding process underwent to calculate the code symbols resulting from the Huffman coding process.

Diagram

Description automatically generated

Figure 5: The code symbol key resulting from the Huffman coding process.

Diagram, schematic

Description automatically generated

Figure 6: The binary string resulting from the Huffman compression of the image in Table 1.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Bit Allocation | N-largest Bit | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| Fourier | RMS | 15.1949 | 14.5466 | 13.9211 | 13.3428 | 12.8006 | 12.252 | 11.821 | 0.1453 |
|  | SNR | 8.3456\*10^-36 | 1.0782\*10^-33 | 2.9129\*10^-34 | 2.5453\*10^-33 | 7.0798\*10^-33 | 7.3124\*10^-33 | 8.9315\*10^-33 | 1.3902\*10^11 |
| Cosine | RMS | 11.5227 | 11.1708 | 11.3224 | 11.6277 | 11.8596 | 11.8018 | 11.821 | 0.1453 |
|  | SNR | 3.0729\*10^-34 | 7.1317\*10^-33 | 2.0568\*10^-32 | 1.6136-10^-34 | 1.1460\*10^-32 | 1.2282\*10^-32 | 2.0355\*10^-31 | 1.3902\*10^11 |

Table 2: The tabulation of Fourier and Cosine transform SNR and RMS error for each N value 1-8.

A person wearing a hat

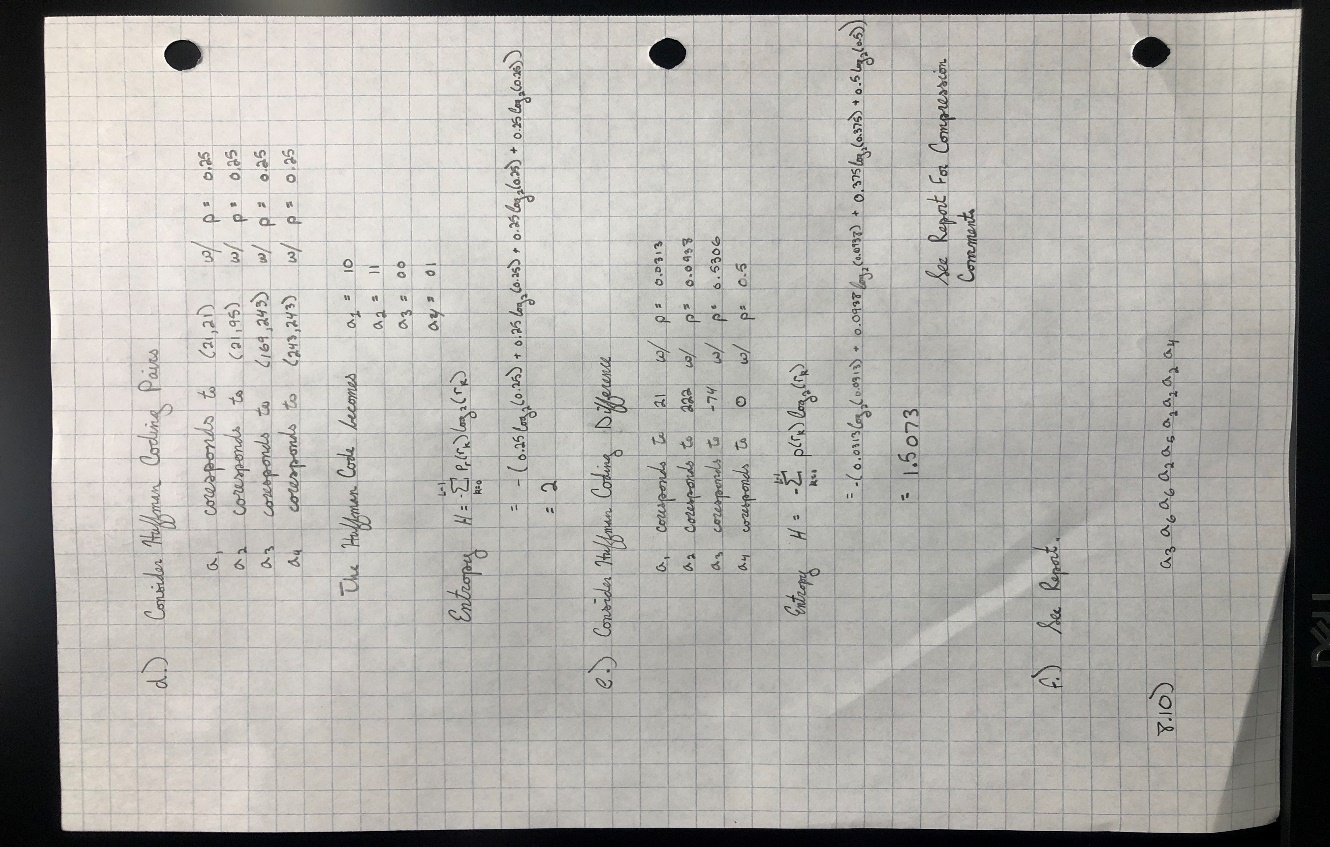
Description automatically generated with low confidence

Figure 7: The Haar wavelet transform of each J=1,2,3 scale had all but the approximation coefficient matrix set to zero prior to the inverse transform. The resultant compression is shown.

**Appendix:**

**Handwritten Solution (Problem 2):**

**Diagram

Description automatically generated**

**Written Programs:**

* main
* fourier8LargeTransform
* cosine8LargeTransform
* computeEntropy
* huffmanCompress
* rootMeanSquareError
* snrmsError
* harrWavelet2D

**Utilized Programs:**

* subplot
* imshow
* cast
* imread
* blockproc
* sort
* length
* mean
* median
* sum
* max
* min
* abs
* zeros
* size
* log2
* questdlg
* num2str
* msgbox
* disp
* nextpow2
* meshgrid
* fft2
* waverec2
* wavedec2
* pwd
* ifft2
* sqrt
* dct2
* idct2
* imhist
* sum
* struct
* real
* idwt2
* appcoef2