CSCI 576 Assignment 2

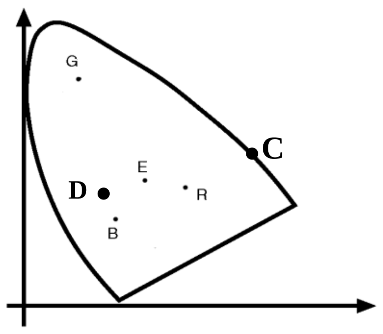
Instructor: Parag Havaldar

Assigned on 09/20/2017

Solutions due 10/20/2017 before noon

Question 1: Color Theory *(20 points)*

One of the uses of chromaticity diagrams is to find the gamut of colors given the primaries. It can also be used to find dominant and complementary colors – Dominant color of a given color D (or dominant wavelength in a color D) is defined as the spectral color which can be mixed with white light in order to reproduce the desired D color. Complementary colors are those which when mixed in some proportion create the color white. Using these definitions and the understanding of the chromaticity diagram that you have, answer the following.



- Show in the plot alongside the dominant wavelength of color D. (*2 points)*
- Do all colors have a dominant wavelength? Explain your reasoning. (*4 points)*
- Show in the plot the spectral color which is complimentary to color C. (*2 points)*
- Given the placements of R, G, B as primaries around the equiluminous point E, can you find out all points which have a value of R=0.5, explaining your reasoning. *Remember that the chromaticity space is non-linear as explained in class, meaning that although E is defined by equal contributions of R, G and B, it may not necessarily be at the centroid of R, G and B. (8 points)*
- Take the R = 0.5 locus above, how does this line map in the RGB space. Explain the mapping from the chromaticity diagram to the RGB space. (*4 points)*

Extra-Credit **Can you find the locus of points on the chromaticity diagram with R, G, B and E fixed as shown, for which R=0.75, this might not be as easy as you think!** *(30 points)*

Question 2: Generic Compression Problem *(20 points)*

The following sequence of real numbers has been obtained sampling a signal:
5.8, 6.2, 6.2, 7.2, 7.3, 7.3, 6.5, 6.8, 6.8, 6.8, 5.5, 5.0, 5.2, 5.2, 5.8, 6.2, 6.2, 6.2, 5.9, 6.3, 5.2, 4.2, 2.8, 2.8, 2.3, 2.9, 1.8, 2.5, 2.5, 3.3, 4.1, 4.9
This signal is then quantized using the interval [0,8] and dividing it into 32 uniformly distributed levels.

- What does the quantized sequence look like? For ease of computation, assume that you placed the level 0 at 0.25, the level 1 at 0.5P, level 2 at 0.75, level 3 at 1.0 and so on. This should simplify your calculations. Round off any fractional value to the nearest integral levels (*6 points)*
- How many bits do you need to transmit it? (*2 points)*
- If you need to encode the quantized output using DPCM. Compute the successive differences between the values – what is the maximum and minimum value for the difference? Assuming that this is your range, how many bits are required to encode the sequence now? Ignore the first value (*4 points)*
- What is the compression ratio you have achieved? (*2 points)*
- Instead of transmitting the differences, you use Huffman coded values for the differences. How many bits do you need now to encode the sequence? (*4 points)*
- What is the compression ratio you have achieved now? (*2 points)*

Question 3: Arithmetic Compression *(20 points)*
Consider two symbols, A and B, with the probability of occurrence of 0.8 and 0.2, respectively. For the purpose of arithmetic compression, you organize a symbol stream of A's and B's in groups of three and give each unit of three symbols an arithmetic code (Assume that each symbol occurrence is independent of previous symbol occurrences).

- How many types three symbol units are there and what are their probabilities? (*2 points)*
- Show the arrangement of these three symbol units on the unit interval [0, 1] and determine the arithmetic code for each. (*8 points)*
- What is the average code word length? Is it optimum? (*3 points)*
- How many bits are required to code the message "ABABBAABBAAABBB? (*3 points)*
- How could you do better than the above code length? (*4 points)*

Question 4: Entropy *(20 points)*
Consider a communication system that gives out only two symbols X and Y. Assume that the parameterization followed by the probabilities are $P(X) = x^2$ and $P(Y) = (1-x^2)$.

- Write down the entropy function and plot it as a function of *x*. (*4 points)*
- From your plot, for what value of *x* does the Entropy become a minimum? (*2 points)*
- Although the plot visually gives you the value of *x* for which the entropy in minimum, can you now mathematically find out the value(s) for which the entropy is a minimum? (*6 points)*
- Can you do the same for the maximum, that is can you find out value(s) of *x* for which the value is a maximum? (*8 points)*

Programming Part DCT vs DWT: (*90 points + 30 points for progressive test*)
This assignment will help you gain an understanding of issues that relate to image compression, by comparing and contrasting the frequency space representations using the Discrete Cosine Transform and the Discrete Wavelet Transform. You will read an RGB file and convert the file to an 8x8 block based DCT representation (as used in the JPEG implementation) and a DWT representation (as used in the JPEG2000 implementation). Depending on the second parameter *n* you will decode both the representations using only *n* coefficients and display them side to side to compare your results. Remember all input files will have the same format as explained to the class website. They will be of size 512x512 (intentionally square and a power of 2 to facilitate easy encoding and decoding). Your algorithm, whether encoding or decoding, should work on each channel independently.

Input to your program will be 2 parameters where:

- The first parameter is the name of the input image file. (file format description provided and is similar to the first assignment format of RGB)
- The second parameter is an integral number that defines the number of coefficients to use for decoding. The interpretation of this parameter of decoding is different for both the DCT and DWT cases so as to use the same number of coefficients. Please see the implementation section for an explanation

Typical invocations to your to your program would look like

MyExe Image.rgb 262144
Here you are making use of all the coefficients to decode because the total number of coefficients for each channel are going to be 512*512= 262144. Hence the output for each DCT and DWT should be exactly the same as the image with no loss.

MyExe Image.rgb 131072
Here you are making use of 131072 (half of the total number) of coefficients for decoding. The exact coefficients you use will vary depending on DCT or DWT. Refer to the implementation section for this.

MyExe Image.rgb 16384
Here you are making use of 16384 (1/16th of the total number) of coefficients for decoding. The exact coefficients you use will vary depending on DCT or DWT. Refer to the implementation section for this.

Implementation
Your implementation should read the input file and convert (for each channel) a DCT representation and a DWT representation.

Encoding:
For the DCT conversion, break up the image into 8x8 contiguous blocks of 64 pixels each and then perform a DCT for each block for each channel. For the DWT conversion, convert each row (for each channel) into low pass and high pass coefficients followed by the same for each column applied to the output of the row processing. Recurse through the process as explained in class through rows first then the columns next at each recursive iteration, each time operating on the low pass section.

Decoding:
Based on the input parameter of the number of coefficients to use, you need to appropriately decode by zeroing out the unrequested coefficients (just setting the coefficients to zero) and then perform an IDCT or an IDWT. The exact coefficients to zero out are different for both the DCT and DWT cases and explained next.

For a DCT, you want to **select the first m coefficients in a zig zag order for each 8x8 block** such that m = round(n/4096) where n is the number of coefficients given as input. 4096 is the number of 8x8 blocks in a 512x512 image. Thus, m represents the first few coefficients to use for each 8x8 block during decoding. So for the second test run above (n=131072), you will use m = round(131072/4096) = 32. Each block will be decoded using the first 32 coefficients in zigzag order. The remaining can be set to zero prior to decoding. For the third test run above (n=16384), you will use m = round (16384/4096) = 4.

For a DWT, you want to **select the first n coefficients in a zig zag order for the image** where n is the number of coefficients given as input. DWT encodes the entire image so, here you will use the first n coefficients in zig zag order. The remaining coefficients will be set to zero prior to decoding. The zig zag ordering naturally prioritizes the low frequency coefficients compared to the higher frequency coefficients. Remember to follow the reverse sequence when decoding, that is first column and second row.

Progressive Analysis of DCT vs DWT *(30 points)*
Here you will create an animation which will take incremental steps of decoding in order to study the output quality of your DCT vs DWT implementation. For this invocation we will put n= -1, because the incremental number of coefficients are predefined as shown below.

MyExe Image.rgb -1
This suggests the program has predefined incremental settings on the number of coefficients to use and is going to loop through them incrementally.

We have 4096 8x8 blocks in the image. So let's start by using one coefficient of each block and then incrementing it on each iteration. Total number of progressive iterations is 64

For the DCT decoding, you will use

- first iteration - the DC coefficient for each block (total 4096 coefficients)
- second iteration – the DC, AC_1 coefficient of each block (total 8192 coefficients)
- third iteration – the DC, AC_1, AC_2 coefficient of each block (total 12288 coefficients)

…
…
- sixty forth iteration – the DC, AC_1, AC_2 …. AC_{63} coefficient of each block (total 512*512= 262144 coefficients)

For the DWT decoding, you will use

- first iteration - the first 4096 coefficients in zigzag order.
- second iteration – the first 8192 coefficients in zigzag order.
- third iteration – the first 12288 coefficients in zigzag order

….

….

- sixty forth iteration – all the total 512*512= 262144 coefficients.

What should you submit?

- Your source code, and your project file or makefile. Please confirm submission procedure from the TAs. Please do not submit any binaries or data sets. We will compile your program and execute our tests accordingly.
- Along with the program, also submit an electronic document (word, pdf, pagemaker etc) for the written part and any other extra credit explanations.