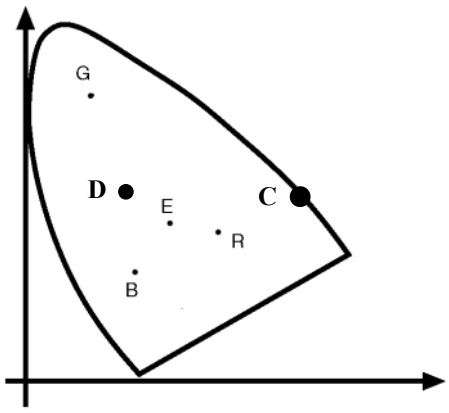
CSCI 576 Assignment 2

Instructor: Parag Havaldar
Assigned on 02/23/2022
Solutions due 03/11/2022 by NOON

Theory Questions: (50 points)

Question 1: Color Theory *(15 points)*
One of 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.



- In the side image, find the dominant wavelength of color D. Show this wavelength. *(2 points)*
- Do all colors have a dominant wavelength? Explain your reasoning *(2 points)*.
- Find the color which is complimentary to the color C. Show this color *(2 points)*.

In the next few questions, you are asked to reason about colors in the chromaticity space that have a specific amount of R (*2 + 2 + 5 points)*

- Show the locus of all points or colors that have R=0
- Show the locus of all points or colors that have R=1
- Show the locus of all points or colors that have R=0.5

Question 2: Generic Compression *(10 points)*
Consider a source that emits an alphabet with three symbols A, B, C having probability distributions as follows - P(A) = 0.625, P(B) = 0.125, P(C) = 0.25

- Construct a Huffman code and calculate its average code length. *(2 points)*
- For this three-symbol vocabulary, how many Huffman codes are possible. What are they? *(2 points)*
- Is the code you have defined optimal - give reasons! If not, what can you do to improve upon this. Show a solution by an example computing the avg. code length. *(6 points)*

Question 3: Arithmetic Compression *(10 points)*

Consider two symbols, A and B, with the probability of occurrence of 0.8 and 0.2, respectively. The coding efficiency can be improved by combining N symbols at a time (called "symbol blocking"). Say N = 3, so you are grouping symbols of 3 and giving them a unique code. (Assume that each symbol occurrence is independent of previous symbol occurrences).

- How many types of different outcomes are there and what are their probabilities? *(2 points)*
- Show the arrangement of symbols on the unit interval [0, 1] and determine the arithmetic code for the three-symbol sequence. *(6 points)*
- What is the average code word length? *(2 points)*

Question 4: Image Dithering *(15 points)*

Let's say that we have an original 12x8 image represented as 8 bits per pixel. In the normalized representation shown below, assume that 0 corresponds to white and 9 corresponds to black.

1	2	3	4	5	6	7	8	9	0	1	2
0	1	2	3	4	5	6	7	8	9	0	1
9	0	1	2	3	4	5	6	7	8	9	0
8	9	0	1	2	3	4	5	6	7	8	9
7	8	9	0	1	2	3	4	5	6	7	8
6	7	8	9	0	1	2	3	4	5	6	7
5	6	7	8	9	0	1	2	3	4	5	6
4	5	6	7	8	9	0	1	2	3	4	5

Answer the following questions You may want to code/script a process to generate the final outputs, but only final outputs are expected. Also, we have asked you to plot this 12x8 image block as a gray color image, and its processed outputs as binary black/white images. For reasons of visible clarity (because a 12x8 image block is very small) you may want to show a zoomed or magnified picture.

- Plot the image as an 8-bit gray scale map, that is - create a 12x8 image to show the original gray image block. *(2 points)*
- If you thresholded the above 12x8 image block such that all values below 4.5 were 0 and above 4.5 were 9, how does your output image B/W block look? Plot an image (*3 points*)
- We can create a better binary image output by using a dithering matrix. Compute the binary output of a dithering operation on the gray level 12x8 image using the dithering matrix D given below. Assume that the image top left coordinate indexes are [0, 0]. Show a graphical binary image plot of the dithered output. *(5 points)*

$$D = \begin{bmatrix} 6 & 8 & 4 \\ 1 & 0 & 3 \\ 5 & 2 & 7 \end{bmatrix}$$

- What if the image block's top left coordinate indexes start with [1, 1]. Show a graphical binary image plot of the dithered output. *(5 points)*

Programming on DCT based streaming (150 points)

This programming assignment will help you to understand the working of DCT and how it is used by standard compression algorithms like JPEG and MPEG. Specifically, you will implement a DCT based coder-decoder for compressing an image and simulate decoding using the baseline mode as well as progressive modes of data delivery. Your program will take as input 4 parameters and be invoked as

myProgram InputImage quantizationLevel DeliveryMode Latency

where the parameters are defined as :

- *InputImage* – is the image to input to your coder-decoder (you may assume a fixed size and format that will be described on the class website).
- *QuantizationLevel* – a factor that will decrease/increase compression as explained below. This value will range from 0 to 7.
- *DeliveryMode* – an index ranging from 1, 2, 3. A 1 implies baseline delivery, a 2 implies progressive delivery using spectral selection, a 3 implies progressive delivery using successive bit approximation.
- *Latency* – a variable in milliseconds, which will give a suggestive "sleep" time between data blocks during decoding. This parameter will be used to "simulate" decoding as data arrives across low and high band width communication networks.

Your program should display two images – the original on the left and the decoded version on the right. Here are some example input parameter invocations

1. myProgram Example.rgb 0 1 0
Here you are encoding *Example*.rgb and using a 0 quantization level, which means no quantization. You are using the baseline mode and there is no latency so you should see the whole output image almost instantaneously, taking into account computational time.

2. myProgram Example.rgb 3 1 100
Here you are encoding *Example.rgb* and using a 3 quantization level. You are using the baseline mode and there is a latency while decoding and so you should see the output data blocks appear as they get decoded.

3. myProgram Example.rgb 1 2 100
Here you are encoding *Example.rgb* and using a 1 quantization level. You are using the progressive spectral selection mode and there is a latency while decoding and so you should see the output appear in stages.

And now for the details of each part –

The Encoder

You will have to start, with an RGB image file (images will be kept on the class website). Implement jpeg-like compressor. Here you will do almost all of the JPEG compression steps except the entropy coding part (RLE for AC or DPCM for DC, entropy coding) and

producing the actual formatted bit stream. Also, the JPEG pipeline contains chroma subsampling, but I am not insisting that you convert to YCrCb and subsample Cr,Cb. The main objective here is to study the DCT and its use in encoding and decoding. So, start with the RGB image.

- For each component (R, G and B) break it into 8x8 blocks
- For each block, do a DCT on the blocks to compute the DC and AC coefficients. You will start with 64 *f(x,y)* pixels and obtain 64 *F(u,v)* frequency coefficients.
- Quantize the DC and AC coefficients with a uniform quantization table, all of whose entries are 2^N, where N is the quantization level given as a parameter above. Each table entry is the same. Quantization works by

  $$F'[u, v] = \text{round}\ (\ F[u, v] / 2^N).$$

  An entry here specifies the range of each interval. So if $N = 0$, then $2^N = 1$ and hence every interval has range 1, In this case *F'[u, v]* is the same as *F[u, v]* and there is no quantization effect.

Now you have the DCT coefficients for all the blocks for all the components computed and quantized. This is the output of the encoder.

The Decoder

The next step is to write a decoder. To decode each image block

- Dequantize all the coefficients. This is done as

  $$F[u, v] = F'[u, v] * 2^N$$

- Do an Inverse DCT on the de-quantized AC and DC coefficients to recover the image block signal. The recovered image is of course with some loss depending on *N*.

Simulating various modes of encoding-decoding

The main modes that you will be simulating are baseline, progressive encoding using spectral selection and progressive encoding using successive bit approximation explained in class. To better understand the results of this step, you will need to use the last two parameters – *Delivery Mode* and *Latency*. Latency simulates communication at different bandwidths. You may assume that all the data is not available at once for decoding but data is available in limited amounts for decoding and display depending on the latency and the mode used.

For this step you need to implement a display loop, which displays the currently decoded data. The latency parameter simulates communication for different bandwidths. This parameter simply controls the time delay (in milliseconds) between packets arriving during communication. You are not implementing any networked communication as yet, so this simulation would amount to inserting a “sleep(time)” statement as you are decoding your data blocks. This means you will have to decode data, display it, sleep for required latency time and repeat these decode-display-sleep steps for every iteration.

Here is how the decoding should proceed depending on the mode used.

1. Sequential Mode
   Each image block is encoded in a single left-to-right, top-to-bottom scan. You may assume that each latency iteration pertains to ONE BLOCK. So the process progresses as

   Decode data of first block and display …sleep
   Decode data of second block and display …sleep
   …

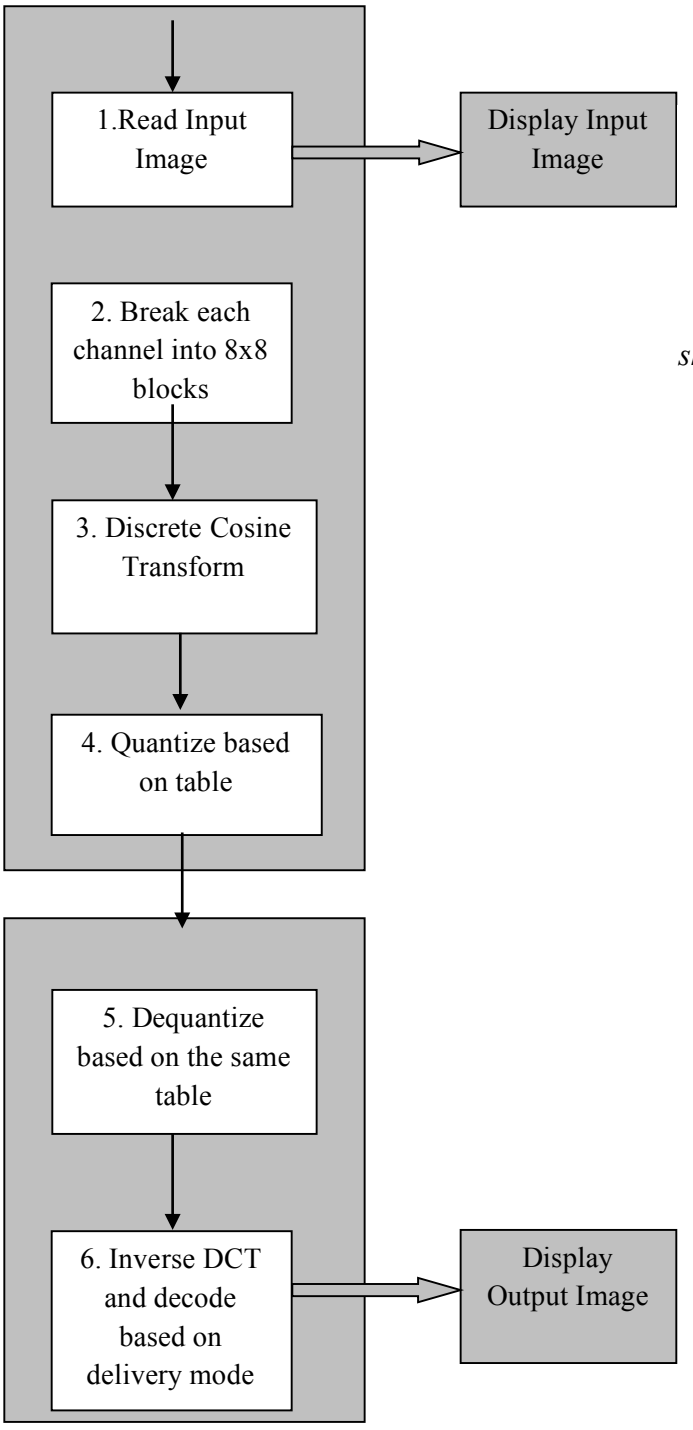
2. Progressive Mode – Spectral Selection
   The DC coefficients of every image blocks is decoded first and displayed. Next the first AC coefficients is added for all the blocks and decoded. This goes on till all the coefficients are added to the decoding process. You may assume that each latency iteration occurs after EVERY SPECIFIC DCT COFFICIENT for all blocks. So the process progresses as

   Decode *all blocks* using only DC coefficient (set rest to zero) …sleep
   Decode *all blocks* using only DC, AC_1 coefficient …. sleep
   Decode *all blocks* using only DC, AC_1, AC_2 coefficient …. sleep
   …

3. Progressive Mode – Successive Bit Approximation
   All DC and AC coefficients of all image blocks are decoded first and displayed in a successive-bit manner. So you will decode all blocks using the all the DC and AC coefficients, but only using the first significant bit of all coefficients Next, you will decode all DC and AC coefficients using the first two significant bits of all coefficients and so on. You may assume that each latency iteration occurs at EACH SIGNIFICANT BIT usage. So the process progresses as

   Decode all blocks using 1_{st} significant bit of all coefficients …sleep
   Decode all blocks using 1_{st}, 2_{nd} significant bit of all coefficients …. sleep
   Decode all blocks using 1_{st}, 2_{nd}, 3_{rd} significant bit of all coefficients …. sleep

Here is the dataflow pipeline of the encoding and decoding,



This code is already provided to you, if you choose to make use of it

Before DCT, each channel of image should be divided into 8x8-sized blocks

You need to use DCT function in the Lecture notes

Quantize DC and ACs based on uniform quantization table

Dequantize DC and ACs based on uniform quantization table

Do Inverse DCT and display image based on mode parameter