CS 576 – Assignment 2
Instructor: Parag Havaldar

Assigned on Monday 02/12/2018,
Solutions due on Friday 03/05/2018 at midday noon,

Theory Problem 1: Color Theory (*15 points*)
In a three-dimensional color space such as XYZ, any color *C* with coordinates (*X, Y, Z*) can be expressed as a linear combination of the primaries P_1, P_2, P_3 with coordinates (X_1, Y_1, Z_1), (X_2, Y_2, Z_2) and (X_3, Y_3, Z_3) respectively. This may be expressed as

$$C\,(X, Y, Z) = \alpha_1 * P_1\,(X_1, Y_1, Z_1) + \alpha_2 * P_2\,(X_2, Y_2, Z_2) + \alpha_3 * P_3 (X_3, Y_3, Z_3)$$

In this question you are asked to show that similarly, the normalized chromaticity coordinates of *C* can *also* be expressed as a linear combination of the normalized chromaticity coordinates of P_1, P_2, P_3. Proceed by answering the following:

- Find the normalized chromaticity coordinates of P_1, P_2, and P_3 in terms of given known quantities (*3 points*)
- Express the normalized chromaticity coordinates of the color *C* in terms of the chromaticity coordinates of P_1, P_2, and P_3 (*6 points*)
- Hence prove that the chromaticity coordinates of any color *C* (which is a linear combination of primaries P_1, P_2, and P_3 in XYZ color space) can be represented also as a linear combination of the chromaticity coordinates of the respective primaries. (*6 points*)

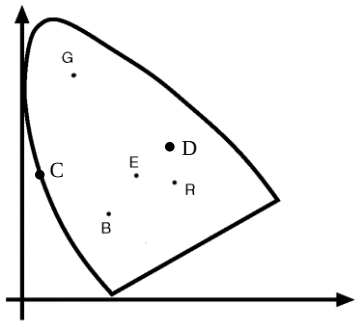
Theory Problem 2: Generic Compression Problem *(15 points)*
The following sequence of real numbers has been obtained sampling a signal:
5.8, 6.2, 6.2, 7.2, 7.3, 7.3, 6.5, 6.8, 6.8, 6.8, 5.5, 5.0, 5.2, 5.2, 5.8, 6.2, 6.2, 6.2, 5.9, 6.3, 5.2, 4.2, 2.8, 2.8, 2.3, 2.9, 1.8, 2.5, 2.5, 3.3, 4.1, 4.9
This signal 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
- How many bits do you need to transmit it?
- 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?

- What is the compression ratio you have achieved?
- Instead of transmitting the differences, you use Huffman coded values for the differences. How many bits do you need now to encode the sequence?
- What is the compression ratio you have achieved now?

Theory Problem 3: 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 G=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 G = 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)*

Programming Assignment *(100 points)*

Here you will be attempting 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. You 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 *use the same images from the last assignment*)
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" low and high band width decoding to properly evaluate the simulation of your delivery modes.

Here are some example input parameter invocations

1. *myProgram Foreman.rgb 0 1 0*

Here you are encoding Foreman.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

2. *myProgram Foreman.rgb 3 1 100*

Here you are encoding Foreman.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 in stages.

3. *myProgram Foreman.rgb 1 2 100*

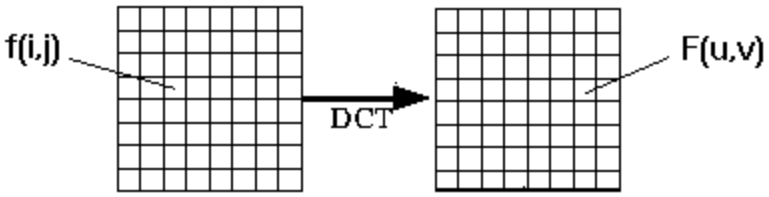
Here you are encoding Foreman.rgb and using a 1 quantization level. You are using the progressive spectral selection mode and there is a latency while decoding and so you should see the output appear in stages.

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 You

have already done this and the main objective here is to study the DCT and its use in decoding. So, start with the RGB image.

1) For each component (R, G and B) break it into 8x8 blocks
2) For each block, do a DCT (see formula in class notes) on the blocks to compute the DC and AC coefficients.



3) 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

1. Dequantize all the coefficients. This is done as

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

2. 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.

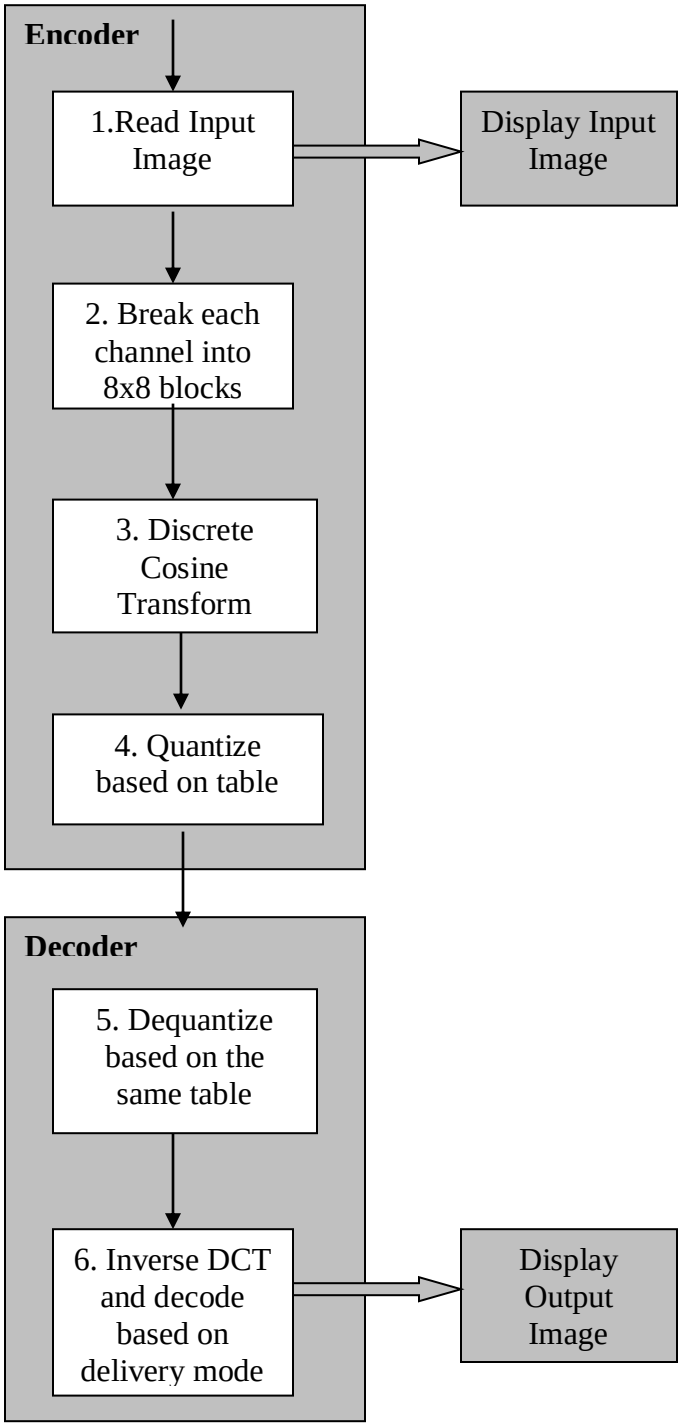
For this step you need to implement a display loop, which displays the currently decoded data. The latency parameter simulates communication for different bandwidths. This parameter simply controls the time delay (in milliseconds) between packets arriving during communication. You are not implementing 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 in the *BASELINE* mode with no latency,



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 the DCT function in the Lecture notes

Quantize DC and ACs based on a uniform quantization table

Dequantize DC and ACs based on uniform quantization table that is same with above

Do Inverse DCT and display image based on M parameter

What should you submit ?

- Your source code, and your project file or makefile, if any, using the submit program. ***Please do not submit any binaries***. We will compile your program and execute our tests accordingly.