CS 576 – Assignment 3
Instructor: Parag Havaldar

Assigned on Mon 10/21/24,
Solutions due on Monday 11/11/24 by 5:00 pm

Late Policy: None. Any assignment submitted after 5 pm will not be graded. If multiple assignments are submitted, the last one before the deadline will be graded

Understanding DCT vs DWT *(100 points + 100 points for progressive analysis)*

This assignment will help you gain an understanding of issues that relate to image compression by comparing the frequency space representations using the Discrete Cosine Transform and the Discrete Wavelet Transform. You will read an RGB file with dimensions 512x512 as input. You will convert the data 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 by side to compare your results. You may assume all input files will have the same RGB format as previous assignments and 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 and size is similar to the second assignment)
- The second parameter is an integral number *n* that defines the number of coefficients to use for decoding. *n* will be a power of 4 whose values range from 4096 … 262144. The interpretation of this parameter for decoding is different in the inverse DCT and inverse DWT cases so as to use the same number of coefficients. Please see the implementation section for an explanation
- For the progressive encoding analysis *n* = -1, OR *n* = -2, see description below

Output of your program will be two images displayed side by side – firstly, encoded-decoded using *n* DCT coefficient and secondly, encoded-decoded using *n* DWT coefficients.

Typical invocations 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 65536
Here you are making use of 65536 (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 ($1/16^{th}$ 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.

Encoding Implementation:
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 a 512x512 image, there should be a total of 64x64 (=4096) blocks. For the DWT conversion, convert each row (for each channel) into low pass and high pass coefficients taken pairwise 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*=65536), you will use *m* = round(65536/4096) = 16. Each block will be decoded using the first 16 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. You may assume that *n*>4096, since we need at least 1 coefficient per block.

For a DWT, you want to **select *n* coefficients that correspond to the lower left square** where *n* is the number of coefficients given as input. DWT encodes the entire image so, here you will use the *n* coefficients as described and set others to zero prior to decoding. Remember to follow the reverse sequence when decoding, that is first column decode and then row decode.

Progressive Analysis of DCT vs DWT *(100 points – 40+60)*
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.

Part A: For this invocation we will put *n* = -1, You will show progressive iterations happening on DCT and DWT as discussed below. The number DCT progressive iterations will be 64 (one iteration for each additional coefficient - DC, AC_1 … AC_{63}) while the number of DWT iterations will be 10 (with the number of coefficient being power of 4 – 1, 4, 16 …262144)

For the DCT progressive decoding, we have 4096 blocks each of size 8x8. Start by using one coefficient of each block and then incrementing it on each iteration. Total number of progressive iterations you will be 64 and will proceed as follows:

- 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 decode iteratively using coefficients in powers of 4, for a total of 10 iterations. Each iterations using the lower left size of $2^k x 2^k$ where k is the iteration number, starting with k=0.

- first iteration – k =0, use one coefficient, the lower left corner in the DWT representation. coefficients shown below. Set the rest to zero
- second iteration – k=1, for a total of 4 (2x2) in the lower left corner, set the rest to zero.
- third iteration – k=2, for a total of 16 (4x4) in the lower left corner, set the rest to zero.
- fourth iteration – k=3, for a total of 64 (8x8) in the lower left corner, set the rest to zero.
- ….
- ….
- ninth iteration – k=8, for a total of 65536 (256x256) in the lower left corner, set the rest to zero.
- tenth iteration – k=9, for a total of 262144 (512x512) which is all the coefficients

Part B: For this invocation we will put *n* = -2. The progressive comparison in part A did show a progressive implementation of both compression schemes but was not an equivalent quality comparison because of the different number of iterations – 64 for DCT and 10 for DWT. In this invocation, you will need to make both iteration = 64 so that you can visually evaluate the quality for each iteration together. Propose a way to select appropriate coefficients in the progressive DWT decoding so as to have the number of coefficients match the progressive DCT decoding. For best results, you will need to lower pass in both rows and columns, followed by either lowpass in rows high pass in columns, and vice versa with finally high pass in both rows and columns. In both cases you need 64 iterations with

- first iteration - total 4096 coefficients
- second iteration –total 8192 coefficients
- third iteration –total 12288 coefficients

…

…

- sixty forth iteration – total 512*512= 262144

The DCT coefficients for every pass will be the same as part A. You need to decide on the DWT coefficients for every pass. Grading will favor better quality of outputs. ***Please submit a word/pdf explaining your choices of coefficients here and why they would work well.***

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 of progressive decoding part B.