CSCI 576 Assignment 3

Instructor: Parag Havaldar

Assigned on 10/17/2022

Solutions due 10/31/2022 by 4:00 pm afternoon

Question 1: DCT Coding *(20 points)*

In this question you will try to understand the working of DCT in the context of JPEG. Below is an 8x8 luminance block of pixel values and its corresponding DCT coefficients.

188	180	155	149	179	116	86	96
168	179	168	174	180	111	86	95
150	166	175	189	165	101	88	97
163	165	179	184	135	90	91	96
170	180	178	144	102	87	91	98
175	174	141	104	85	83	88	96
153	134	105	82	83	87	92	96
117	104	86	80	86	90	92	103

You are expected to provide the matrix relevant to the question. You do not have to submit any scripts for them. If you would like to illustrate steps, you may provide intermediate steps for example in part 1, you may choose to provide the matrix after DCT and then the final matrix after quantization.

- Using the 2D DCT formula, compute the 64 DCT values. Assume that you quantize your DCT coefficients using the luminance quantization table K1 on page 143 of the uploaded ITU-T JPEG standard. What does your table look like after quantization? *(5 points)* Q=100
- In the JPEG pipeline, the quantized DCT values are then further scanned in a zigzag order. Ignoring your DC value, show the resulting zigzag scan AC values. *(2 points)*.
- For this zigzag AC sequence, write down the intermediary notation *(5 points)*
- For these are luminance values, write down the resulting JPEG bit stream. You will need to consult standard luminance code tables on page 150 of the ITU-T JPEG standard. *(6 points)*
- What compression ratio do you get for this luminance block? *(2 points)*

Programming on DWT Compression *(80 points)*

This programming assignment will help you gain an understanding of issues that relate to image compression using wavelets. You will read an RGB file and convert the image pixel to a DWT representation (as used in the JPEG2000 implementation) for each channel. Depending on the second parameter *n* you will decode both the representations using only *n* levels of the low pass coefficients and display the output. 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 rgb file. (file format is similar to previous assignments).
- The second parameter *n* is an integral number from 0 to 9 that defines the low pass level to be

used in your decoding. For a given *n*, this translates to using 2^n low pass coefficients in rows and columns respectively to use in the decoding process. Additionally, *n* could also take a value of -1 to show progressive decoding. Please see the implementation section for an explanation

Typical invocations to your program would look like

MyExe Image.rgb 9
This the level 9 or the highest level and as such corresponds to your entire image. Here you are making use of $2^9 = 512$ coefficients in rows and 512 coefficients in columns, which essentially is the input image itself and so the output should look just like the input.

MyExe Image.rgb 8
This is level 8 and the first level of the wavelet encoded hierarchy in rows and columns. Here you are making use of $2^8 = 256$ low pass coefficients in rows and 256 low pass coefficients in columns,

MyExe Image.rgb 1
This is level 1 and the eight level in the wavelet encoded hierarchy in rows and columns . Here you are making use of $2^2 = 4$ low pass coefficients in rows and 4 low pass coefficients in columns,

typo： 4 -> 2

Encoding Implementation

For the DWT encoding process, 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 until you reach the appropriate level

Decoding Implementation:

Once you reach the appropriate level, zero out all the high pass coefficients. Then perform a recursive IDWT from the encoded level upto level 9 which is the image level. Yu need to appropriately decode by zeroing out the unrequested coefficients (just setting the coefficients to zero) and then perform an IDWT.

Progressive Encoding-Decoding Implementation

This is when $n = -1$. In this case you will go through the creation of the entire DWT representation till level 0. Then decode each level recursively and display the output. The first display will be at level 0, then level 1 and so on till you reach level 9. You should see the image progressively improving with details.

10 frames

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.