CS 576 – Assignment 3
Instructor: Parag Havaldar

Assigned on Monday 03/04/2024,

Solutions due on Monday 04/01/24 (April 1) by midday 12 pm noon

Late Policy: None.

Total marks = 200

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. Additionally, you will 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 N* – a factor that will decrease/increase compression as explained below. This value will range from 0 to 7.
- *DeliveryMode M* – an integer that can have a value of 1 or 2 or 3. A 1 implies baseline delivery, a 2 implies progressive delivery using spectral selection, a 3 implies progressive delivery using successive bit approximation.
- *Latency L* – 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. A high millisecond value corresponds to low band width and vice versa.

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
   Decode data of third block and display …sleep
   …
   Decode data of last 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
   …
   Decode *all blocks* using only DC, AC_1, AC_2 … AC_{63} coefficients…. 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

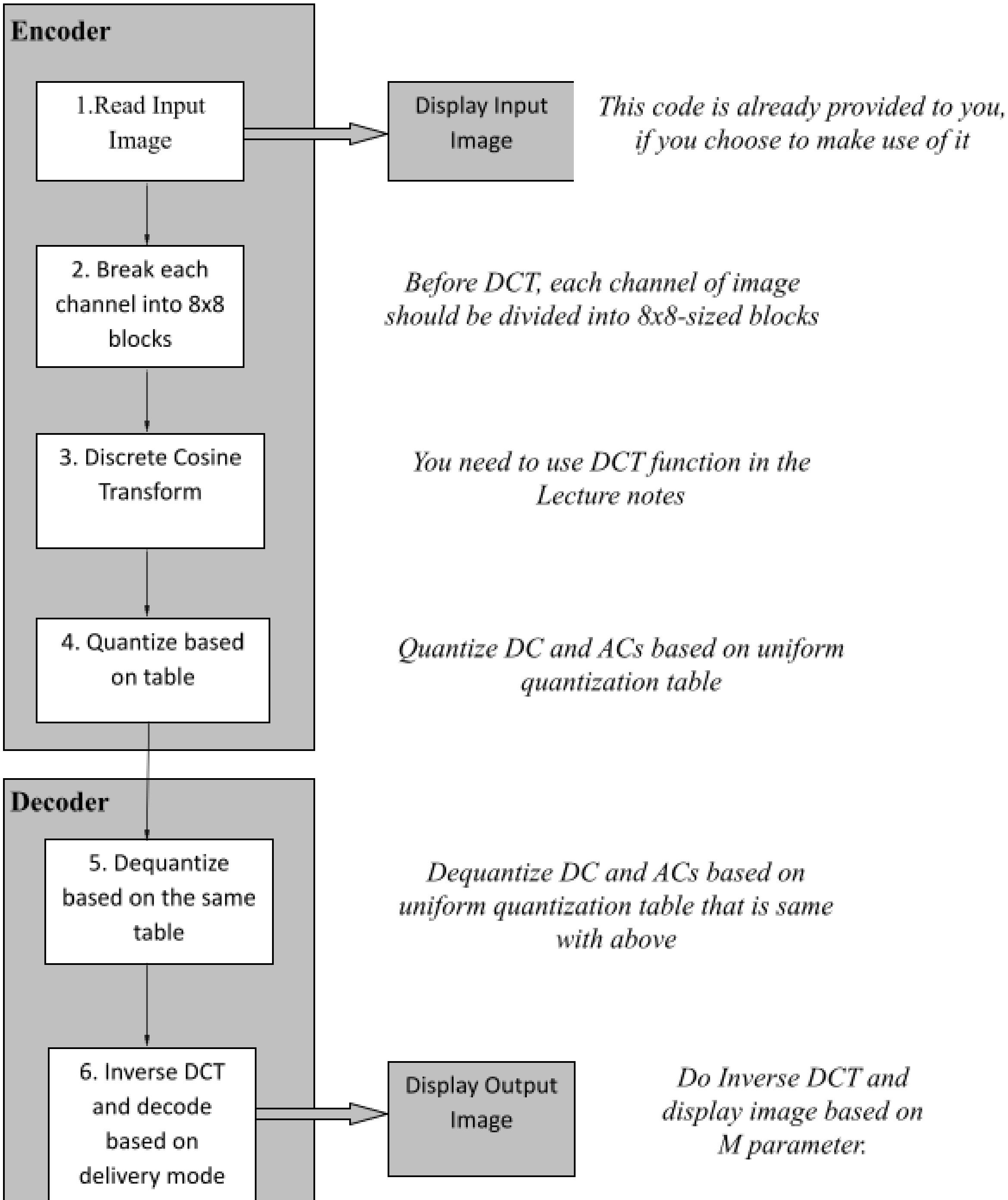
   Decode all blocks using 1st significant bit of all coefficients …Sleep

Decode all blocks using 1st, 2nd significant bit of all coefficients …. Sleep
Decode all blocks using 1st, 2nd, 3rd significant bit of all coefficients …. Sleep

What should you submit?

- Your source code, and your project file needed to compile the code. Please do not submit any binaries or images. We will compile your program and run our cases accordingly. If you need to include a readme.txt file with any special compilation instructions, that is fine too.
- You may submit your plots and analysis as a word/pdf document.

Here is the dataflow pipeline of the encoding and decoding.


Encoder
1.Read Input Image
Display Input Image
2. Break each channel into 8x8 blocks
3. Discrete Cosine Transform
4. Quantize based on table
Decoder
5. Dequantize based on the same table
6. Inverse DCT and decode based on delivery mode
Display Output Image

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 that is same with above

Do Inverse DCT and display image based on M parameter.