CS 490/590 Introduction to Multimedia Networking, Fall 2019

Programming Assignment 2 – Simple JPEG Compressor / Decompressor
Assigned: November 12, 2019
Due: 11:59:59pm, Wednesday, December 4, 2019

You are allowed to work with one other person on this assignment. Make sure both your names are on everything you turn in (including the source code). You will both receive the same grade.

In this project, we will be implementing a basic JPEG compression algorithm. Specifically, we will be implementing the DCT Transform and Quantization for a grayscale image. You will be implementing both the compressor and decompressor.

Input File Format
Our compression algorithm will work with PGM files. PGM files can be generated by using the *ppmtopgm* program. PPM files can be generated by using either *giftopnm* or *djpeg* from the pbmplus libraries. The format for PGM files are:

P5\n
<xsize> <ysize>\n
255\n
[xsize*ysize bytes of grayscale data, left to right, top to bottom]

For ease of implementation, we will only use grayscale images for this project.

Compressed File Format
Your compressed file will take the following format (which we will refer to as the .dct format):

MYDCT\n
<xsize> <ysize>\n
Qvalue\n
[xsize/16 * ysize/16 blocks of DCTcoefficients]

Each block will be encoded into the file as:
x_offset(in pixels) y_offset(in pixels)
DCT values zig-zag reordered and
The DCT values will be written out as %5d

xsize and *ysize* will be multiples of 16 and are specified as ASCII characters. We will use input files that are *only* multiples of 16. Qvalue is a *floating point* value specified in ASCII characters.

Compressor (Encoder)
Your program will be named "myDCT.c" and will be compiled to an executable named either *myDCT* or *myDCT.exe*. The program will take 4 input parameters:

myDCT <input image> <quantfile> <qscale> <output file>

As previously mentioned, the input image will be in PGM format. The quantfile, will be specified as a matrix in a text file with 8 values per input line (8 lines total). The *qscale* will be a floating point value specified by the user in ASCII.

Using the input file, the compressor will process the file using 16x16 macroblocks, as in the JPEG standard. Each macroblock will then be processed as 4 8x8 blocks. The blocks will be processed in left to right, top to bottom order as in the JPEG standard. Each block will then be converted to the frequency domain using a DCT transform. The transform is:

DCT:

$$F(u,v) = \frac{C_u}{2}\frac{C_v}{2}\sum_{x=0}^{7}\sum_{y=0}^{7} f(x,y)\cos\left[\frac{(2x+1)u\pi}{16}\right]\cos\left[\frac{(2y+1)v\pi}{16}\right]$$

$$C_u = \begin{cases} \frac{1}{\sqrt{2}} & if\ u = 0 \\ 1 & if\ u > 0 \end{cases} \qquad Cv = \begin{cases} \frac{1}{\sqrt{2}} & if\ v = 0 \\ 1 & if\ v > 0 \end{cases}$$

Obviously, this transform is not integer based. As a result, you should use double values for the transform.

After the DCT transform, the values within the array will range from -2047 to 2048. These values will then need to be quantized and written out to the output file. If the quantization matrix is *qt* and the dct matrix is *dctmatrix*, the resulting integer *quantcoeff* matrix will be determined by:

quantcoeff[x][y] = round(dctmatrix[x][y] / (qt[x][y]*qscale))

where x and y range from 0 to 7.

The quantization values will then have a range, depending upon the quant table, roughly [-200, 200]. To get this range into an unsigned 8-bit value we will be doing two things. First, we will crop the values of the coefficients to the range [-127,128]. That is, anything between 128 and 200 will be remapped to 128. Anything between -200 and -127 will be mapped to -127. Second, we will offset the coefficient by 127. That is, we will add 127 to each of the coefficients to map the range of [-127,128] to [0,255].

After the cropping and offsetting of values is complete, we will write the coefficients out to the outputfile, one byte (in binary) at a time zig-zag re-ordered . Notice, our file format will not lead to any compression. In fact, it will be much larger!

Decompression (Decoder)

Your program will be named "myIDCT.c" and will be compiled to an executable named either *myIDCT* or *myIDCT.exe*. The program will take 3 input parameters:

myIDCT <input image> <quantfile> <output file>

The input format will be a properly formatted "MYDCT" file. The quantfile, will be specified as a matrix in a text file with 8 values per input line (8 lines total).

The program will then decompress the file and write out a PGM file, basically reversing the process of compression / encoding. For the inverse DCT, you will use the following equation:

IDCT:

$$f(x,y) = \frac{1}{4}\sum_{u=0}^{7}\sum_{v=0}^{7} C_u C_v F(u,v)\cos\left[\frac{(2x+1)u\pi}{16}\right]\cos\left[\frac{(2y+1)v\pi}{16}\right]$$

$$C_u = \begin{cases} \frac{1}{\sqrt{2}} & if\ u = 0 \\ 1 & if\ u > 0 \end{cases} \qquad Cv = \begin{cases} \frac{1}{\sqrt{2}} & if\ v = 0 \\ 1 & if\ v > 0 \end{cases}$$

The output of the IDCT may result in values that are NOT in the range [0,255]. In this case, you will need to crop the values. Any values less than 0 will be mapped to 0. Any values greater than 255 will be remapped to 255. The values will also need to be rounded to an integer using the *round()* function.

The overall steps for decompression involve:

1) Resetting the offset of the pixel range to [-127,128]
2) Multiplying the coefficients with the quantization matrix values (and qvalue)
3) Performing the IDCT
4) Rounding and cropping coefficients
5) Writing out the PGM file

What to turn in

You are required to turn in the following via D2L:

1) A short description of your programs. In particular, you must describe at a high-level, the algorithm that you used for DCT and quantization, including the data structures that you used.
2) Well-commented code with sufficient detail for us to understand the various code segments that you are using. Your names must appear at the top of the source code listing in comments
3) Several examples that demonstrate that your program is working. This might be in the form of print statement from your program and the output file.
4) A description of what works, what doesn't, and what kinds of testing you did to ensure that your program is correct.

Other useful tidbits

- Your program should be written in C or C++. We will be grading your assignments on a cs.pdx.edu box.
- You can work in groups of up to 2.
- You can assume a maximum file size of 640x480.
- Start small! The programs can be broken into many smaller pieces. Test each piece before moving on.
- To convert a gif to pgm use:

```
giftopnm [gif filename] | ppmtopgm > [pgm filename]
```

- In order to view the pgm file, you may need to convert it back into a gif file for viewing (depending on the computer you are using). To convert a PGM back to GIF, use the following:

```
pgmtoppm 1,1,1 inputfilename | ppmtogif > outputfilename
```

  The 1,1,1 tells pgmtoppm to use all white for 255 and black for 0.