PROJECT #2: QM Coder, Quantization and JPEG

Issued: 02/06/2015 Due: 03/01/2015

General Instructions:

1. Assignment descriptions and all files mentioned in this homework are available under "Assignments" on DEN Website: http://den.usc.edu.
2. Please follow the submission guidelines for homework, found in the "Course Documents" folder on DEN website.

Problem 1: QM Coder (36%)

Part A: Written Question (6%)

The encoding algorithm of the QM coder can be illustrated in the following pseudo-code:

```
if encoder receives MPS
{
      A=A-Q_e;
      if A<0x8000
      {
            if A<Q_e {
                  if C+A > 0xFFFF {carry = true;}
                  C+=A; A=Q_e;
            }
            Q_e changes its state and then the value according to
      Appendix A column 4;
            renormalize();

      }

}
else if encoder receives LPS
{
      A=A-Q_e;
      if A>=Q_e {
            if C+A > 0xFFFF {carry = true;}
            C+=A; A=Q_e;
      }
      Q_e changes its state and then the value according to Appendix A
column 5;
      renormalize();
}

renormalize()
{
      if carry == true {encoder outputs 1; carry = false;}
      while (A < 0x8000)
      {
          A<<=1;
          encoder outputs MSB of C;
          C<<=1;
      }

}
```

Variable	Description	Data Type
A	Coding interval	unsigned int
C	Coding index, always points to the bottom of A	unsigned short
Qe	Probability of LPS	unsigned int

Please encode the following bit stream using the QM coder and the state transition table given below. For each new bit received at the encoder side, show the updated values of A, C, Qe, state and the MPS for the following bit stream.

- 1100010010 (MPS=0, Current State=10, A=0x10000, C=0x0000, carry = false)

State Transition Table for the QM Coder

State	Qe (Hex)	Qe (Dec)	Increase state by	Decrease state by
0	59EB	0.49582	1	S
1	5522	0.46944	1	1
2	504F	0.44283	1	1
3	4B85	0.41643	1	1
4	4639	0.38722	1	1
5	415E	0.36044	1	1
6	3C3D	0.33216	1	1
7	375E	0.30530	1	1
8	32B4	0.27958	1	2
9	2E17	0.25415	1	1
10	299A	0.22940	1	2
11	2516	0.20450	1	1
12	1EDF	0.17023	1	1
13	1AA9	0.14701	1	2
14	174E	0.12851	1	1
15	1424	0.11106	1	2
16	119C	0.09710	1	1
17	0F6B	0.08502	1	2
18	0D51	0.07343	1	2
19	0BB6	0.06458	1	1
20	0A40	0.05652	1	2
21	0861	0.04620	1	2
22	0706	0.03873	1	2
23	05CD	0.03199	1	2
24	04DE	0.02684	1	1
25	040F	0.02238	1	2
26	0363	0.01867	1	2
27	02D4	0.01559	1	2
28	025C	0.01301	1	2

29	01F8	0.01086	1	2
30	01A4	0.00905	1	2
31	0160	0.00758	1	2
32	0125	0.00631	1	2
33	00F6	0.00530	1	2
34	00CB	0.00437	1	2
35	00AB	0.00368	1	1
36	008F	0.00308	1	2
37	0068	0.00224	1	2
38	004E	0.00168	1	2
39	003B	0.00127	1	2
40	002C	0.00095	1	2
41	001A	0.00056	1	3
42	000D	0.00028	1	2
43	0006	0.00013	1	2
44	0003	0.00006	1	2
45	0001	0.00002	0	1

In the above table, S means that MPS and LPS must exchange because we made a wrong guess. For example, if MPS='1', we change MPS from `1' to `0' when the encoder encounters "S".

Note: C*Qe(Hex)=Qe(Dec) where C is a constant of value around 1.4116

Part B: Encoding using the QM coder (30%)

1. The QM coder is a binary arithmetic coder which encodes two symbols, 0 and 1. In this sub-part, you need to implement the QM coder using the pseudo-code and the state transition table from Part A (Off-the-shelf implementations not allowed). Instead of applying it to a bit stream, you have a sequence of 8-bit symbols now. Map these symbols to the sequences containing 1s and 0s only (two symbols). Four types of media files are provided to you in the '.dat' format wherein each sample is 1 byte (8 bit) long.
   a. You need to encode these media files using the QM coder and get the compression ratios. Use suitable initialization.
   b. Discuss how the mapping function influences the compression ratios. You may use different mapping functions for different media types.
2. Apply context-based QM encoders to the given four types of media (*.dat) files.
   a. Compare compression ratios for different media types using three context rules with n=1, 2, 3, where n is the number of context bits.

   Hints: You need the default context rules for the initial bits in the bit stream. It is common to set all the initial default context bits to zeroes. You still need the mapping function from the previous sub-part. (The source code for the context based QM coder is provided.)
   b. Apply some pre-processing step(s) for images to improve the compression ratio. State your algorithm in detail and discuss your results.

Problem 2: Scalar and Vector Quantization (30%)

Part A: Lloyd-Max Scalar Quantizer Design (15%)

Design a 3-bit and a 5-bit Lloyd-Max quantizer for the image set *chem.256, house.256, moon.256.*

(1) Plot the histogram of each image file.

(2) Design a 3-bit and a 5-bit Lloyd-Max scalar quantizer using a training set.

a) *Training Set:* It is sufficient to design the quantizer with a training set generated from the above three images.

b) *Initial Quantizer*: Choosing the initial representative values is also important in the quantizer design. Assuming that quantizer output value is not constrained to be an integer, please explain how the initial quantizer is chosen.

c) *Update Average Distortion*: During the Lloyd-Max iterative optimization, your iterations should stop with the following constraint:

$$\frac{MSE_{j-1} - MSE_j}{MSE_j} < \varepsilon$$

You should select a small number such as ε=0.001 to obtain a good quantizer.

d) *Report the PSNR at each iteration*. Also report the number of iterations the algorithm takes to convergence.

(3) Quantize the 3 images using the designed quantizer. Please also quantize three images (*f16.256, couple.256, elaine.256*) which were not used in the training process.

(4) Plot the histogram of 6 images after quantization. What is an indication for "good" quantization?

(5) Plot the two rate-distortion pairs in a rate (bits/pixel)-PSNR plane for each image.

(6) Plot the average code word length and PSNR obtained by a theoretically optimal entropy code. (Note: The plots of (5) and (6) should be overlapped in one figure for comparison.)

(7) Discuss any observations from the result of (1) - (6).

Note: In your program, you should use the C/C++ ONLY. You may use Matlab or Excel for plotting, but the data should be generated from your C/C++ program.

Part B: Vector Quantization (15 %)

Vector Quantization is a well-known technique for lossy data compression, pattern recognition and density estimation. For this part, you need to apply vector quantization to the image data. C-implementation of the Standard Vector Quantization and the Tree-Structured Vector Quantization is provided to you for reference. The data consists of 6 images which are separated into the training and the test set as follows:

Training Set*: *chem.256, house.256 and moon.256*

Test Set*: *f16.256, couple.256 and elaine.256.*

1. **Image Blocking**

   The image pixels are stored in a scan-line fashion. As the first step, you have to implement an image blocking function to reorder the image data. You need to read the image data block-by-block and convert that block to a vector. For example, you may divide the image into non-overlapping blocks of size,$2^n \times 2^n$ where $n \in [1, 2 \cdots)$ and transform each block to a vector of dimension $2^n \times 2^n$.

2. **Standard Vector Quantization**

   Use the training set to generate Standard Vector Quantization codebooks of sizes 16, 32 and 64. Assume that the image is divided into non-overlapping blocks of size 2x2. Compress each test image separately with the three codebooks one by one. Obtain the empirical entropy and the average distortion values for each pair. Discuss your results in detail. Repeat the whole process for block sizes of 4x4 and 8x8.

3. **Tree Structured Vector Quantization (TSVQ)**

   Generate TSVQ codebooks having fixed bitrate of 4.0, 5.0 and 6.0 bits per vector. Assume that the image is divided into non-overlapping blocks of size 2x2. Compress the test images separately with these three codebooks one by one. Obtain the empirical entropy and the average distortion values for each pair. Discuss your results in detail. Repeat the whole process for block sizes of 4x4 and 8x8. Compare TSVQ results with those obtained from the Standard VQ.

Problem 3: JPEG Compression Standard (34%)

This problem includes 3 parts. The code for Part B is provided, while in Parts A and C you need to write your own code.

Part A: Discrete Cosine Transform (DCT) and Quantization for JPEG (8%)

In this part, you are given an image "lena.raw" (size:16x16, see Figure 1b), which was resized from the original 256x256 Lena image shown in Figure 1(a). The 16x16 Lena image consists of four 8x8 blocks, and you will compute the quantized DCT coefficients for each block.

(a) Original Lena Image 256x256

(b) Down-sized Lena Image 16x16

Figure 1. Lena Image

(1) First, read in the gray scale values of all pixels in Lena image. For your reference, the C/C++ code (readraw.c/readraw.cpp) to read in raw image files has been provided.

(2) Subtract 128 from each pixel value to change the range of pixel values to [-128,127] instead of the original [0, 255]. This process is needed because the DCT is designed to work on symmetric pixel values ranging from -128 to 127.

(3) Then take the two-dimensional DCT of each 8x8 block, which is given by the following equations (N = 8):

$$D(i,j) = \frac{1}{\sqrt{2N}} C(i)C(j) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} p(x,y) \cos\left[\frac{(2x+1)i\pi}{2N}\right] \cos\left[\frac{(2y+1)j\pi}{2N}\right]$$

$$C(u) = \begin{cases} 1/\sqrt{2} & \text{if } u = 0 \\ 1 & \text{if } u > 0 \end{cases}$$

where $p(x, y)$ is the gray scale value of pixel (x,y) from part (2), and $D(i,j)$ is the DCT coefficient at coordinates (i,j).

(4) All the 8x8 blocks of DCT coefficients are now ready for compression by quantization. A standard quantization matrix Q_{50} with a quality factor of 50 for the JPEG standard is given as follows:

$$Q_{50} = \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}$$

The quantized 8x8 matrix C can be obtained by dividing each element in the DCT coefficients matrix D by the corresponding element in the quantization matrix Q, and then rounding to the nearest integer value:

$$C(i,j) = round(\frac{D(i,j)}{Q(i,j)})$$

Compute the quantized matrix for all four DCT coefficients blocks of "lena.raw" image. Discuss why this nonlinear quantization is applicable considering the characteristic of the human visual system.

(5) Different levels of JPEG compression could be obtained through the selection of specific quantization matrices. The user can decide the quality factor ranging from 1 to 100, where 1 gives the poorest image quality but highest compression and 100 provides the best quality but lowest compression. For quality level other than 50, the quantization matrix at quality factor N is calculated as:

$$Q_N(i,j) = \begin{cases} \frac{100-N}{50} Q_{50}(i,j) & N > 50 \\ \frac{50}{N} Q_{50}(i,j) & N < 50 \end{cases}$$

Write a program to find quantization matrices Q_{10} and Q_{90}. Then compute the two corresponding quantized matrices for Lena image. Compare the quantized matrices for all three quality factors (10, 50, 90) and discuss which of these matrices would be best suited for entropy coding, which is the next step for JPEG compression.

Part B: JPEG Compression Quality Factor (6%)

Public domain (http://www.ijg.org/) JPEG software **jpeg_6b.zip** is provided for your reference. It converts image files in several formats into the JPEG format (compression) and vice versa (decompression). With this software, you can use the “cjpeg” program to compress an image with JPEG scheme.

(1) Create JPEG images from the given “clock.bmp” image with 6 different quality factors: 100, 60, 40, 20, 10, 1. Report compressed file sizes, compression ratios and PSNR of JPEG images for different quality factors.

(2) Explain how the quality factor affects image quality of JPEG images in terms of the quantization matrix, referencing what you've seen in (1). Discuss any visual artifact you observe.

Part C: Post-Processing of JPEG Encoded Images (20%)

In this part, you are required to improve the subjective quality of several JPEG-encoded images. Two images ("clock.raw" and "pepper.raw") were encoded with 5 different quantization step-sizes, and the results ("clock1.jpg" to "clock5.jpg", "pepper1.jpg" to "pepper5.jpg") are included in the material folder. Develop several de-blocking algorithms to remove the blocking artifacts of these images. Several reference papers are provided.

Step #1:

Convert all the provided results (total ten images) into .raw format and calculate the PSNR with original images.

Step #2:

Finish the below de-blocking algorithms:

(1) Intuitive trial: since the blocking artifact happens along the block boundary, you may try some simple low pass filters along the boundary. Design your own method and explain it in detail. Compare the PSNR performance with Step #1.

(2) Please read the provided paper – deblocking_filter.pdf. Understand the algorithm and apply it into ten given images. Calculate the PSNR and compare them with Step #1. Also, discuss the subjective quality. Is there any inconsistency between objective quality and subjective quality? Explain your observation.

(3) Another different method to enhance the JPEG compressed image is proposed in paper – reapplying_JPEG.pdf. For this method, it is very easy to understand and implement (using **jpeg_6b.zip**). However, it can generate reasonable results. Please understand the paper and explain the algorithm in your own word. Compare the PSNR performance with deblocking filter. Discuss the advantages and disadvantages for these two methods.

Note: here the PSNR "gain" might be negative.

Appendix A: Files and Notes for HW#2

Problem 1

QM source file	qm_code.zip
Supplements	QM_state_transition_table.txt
Data files	audio.dat, binary.dat, image.dat, text.dat
References	QM_Coder.pdf; JPEG_QM.pdf

Problem 2

VQCode.zip	C implementation of the Standard VQ and TSVQ
Data files	chem.256, house.256, moon.256, f16.256, couple.256, elaine.256

Problem 3

Original images	lena.raw, pepper.raw, clock.raw, clock.bmp
JPEG images	pepper1-5.jpg, clock1-5.jpg
References	deblocking_filter.pdf
	reapplying_JPEG.pdf
	review_postprocessing.pdf
Source code	jpeg-6b.zip
	readraw.c
	readraw.cpp

Appendix B: PSNR Formula

Peak Signal to Noise Ratio (PSNR) value is defined as follows:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}\left(X'(i) - X(i)\right)^2$$

$$PSNR = 10\log_{10}\left(\frac{MAX^2}{MSE}\right)$$

Where X is the original data and X' is the data after quantization, N is the total number of pixels in the image, and MAX is the maximum possible pixel value in the image.