Homework #6

Due Date: 11:59pm Friday 2 May 2014

In this homework you will write a C++ program that can compress images using the Discrete Cosine Transform (DCT). Your program will read the image on `stdin` and outputs the resulting image on `stdout`, and get in the required operation on the command line.

Please present a report containing your answers as well as a zip file containing all your code.

1. [4 points] Implement the functions `mul_8x8()`, `dct_block()` and `idct_block()` to compute the Discrete Cosine Transform (DCT) and the Inverse DCT on blocks of 8×8 pixels. The 8×8 transform matrix A is as defined in the lecture slides.

   Try the functions on the small images `msg1.pgm` and `msg2.pgm` and compute the SNR to make sure they are working correctly and that the Inverse DCT indeed reconstructs the original DCT with minimal error (high SNR).

2. [3 points] Implement the functions `quantize_block()`, `dequantize_block()` and `compress_dct()` to compress an image using a JPEG-like approach i.e.

   - Compute the DCT of each 8×8 block of pixels in the image.
   - Quantize the resulting DCT coefficients using the a quantization matrix.
   - De-quantize the quantized DCT coefficients i.e. imitate how the coefficients will reach the decoder
   - Compute the Inverse DCT to reconstruct the image from the approximated DCT coefficients.

   The standard quantization table is defined as:

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

   and the quantization process is defined as:

$$s_{i,j}=\left\lfloor \frac{y_{i,j}}{q_{i,j}}+0.5 \right\rfloor$$

where $y_{i,j}$ is the DCT coefficient and $q_{i,j}$ is the corresponding entry in Q, the de-quanitzation process is defined as:

$$\hat{y}_{i,j} = s_{i,j} \times q_{i,j}$$

To obtain different "quality factors" from the standard quantization table above, we simply divide the quantization table by the quality factor, and thus the quantization process becomes:

$$s_{i,j} = \left\lfloor \frac{y_{i,j} \times \text{quality}}{q_{i,j}} + 0.5 \right\rfloor$$

so higher "quality" values correspond to lower quantization errors and better visual quality.

Note: *before* computing the DCT on the input blocks, subtract 128 from each pixel, and add it back *after* computing the inverse DCT.

3. [2 points] Try the DCT compression and quantization on the four images (sena, sensin, earth, omaha) using different quality values (*quality* = 0.1, 0.5, 1, 10), and compute the SNR of the reconstructed image. What do you notice? Include sample output quantized images in your report.

Command Line

You need to modify the main file `hw06.cpp` to include the required functionality. Your program should be named `hw06`, and should be called as follows:

- To compute DCT on a block of 8×8 pixels and output the DCT coefficients on the stdout:

  ```
  ./hw06 -dct_block < input.ppm
  ```

  where the input 8×8 image is called `input.ppm` and the output is written to `stdout`. For example, to compute the DCT of `msg1.ppm`, you could run:

  ```
  ./hw06 -dct_block < msg1.pgm
  ```

- To compute the Inverse DCT on a block of 8×8 pixels and output the reconstructed pixels on the stdout:

  ```
  ./hw06 -idct_block < input.ppm
  ```

  where the input 8×8 image is called `input.ppm` and the output is written to `stdout`. For example, to compute the Inverse DCT of the DCT of `msg1.ppm`, you could run:

  ```
  ./hw06 -dct_block < msg1.pgm | ./hw06 -idct_block
  ```

- To compress an input image using DCT compression/quantization and output the SNR and the reconstructed image on stdout:

  ```
  ./hw06 -dct_compress q < input.ppm
  ```

  where q is the quality factor, the input image is called `input.ppm` and the output is written to `stdout` where the first line contains the SNR and the rest contain the output PPM file. For example, to compress the image `sena.ppm` with quality factor $q = 2$ you could run:

  ```
  ./hw06 -dct_compress 2 < sena.pgm
  ```

Instructions

- All code should be implemented in C++ under Linux.
- Please submit your homework in one zip file named as follows: *CMPN206.HW##.FirstName.LastName.zip*, so for example if your name is Mohamed Aly and this is homework #1, then the file name should be *CMPN206.HW01.Mohamed.Aly.zip*.
- Please include all your code and sample output in the zip file, with a README file to explain what you did. Failure to follow these instructions will cause deductions from your grade.
- You are allowed to discuss the problems among yourselves. However, **copying** any part of the code will result a grade of **ZERO**. No exceptions.

Grading

- 9 points: requirements above
- 1 point: submission instructions