HW SET #1 (DUE BEFORE CLASS ON JAN 27, WED)

Problem 1 (30 points) The 2D DCT is defined as

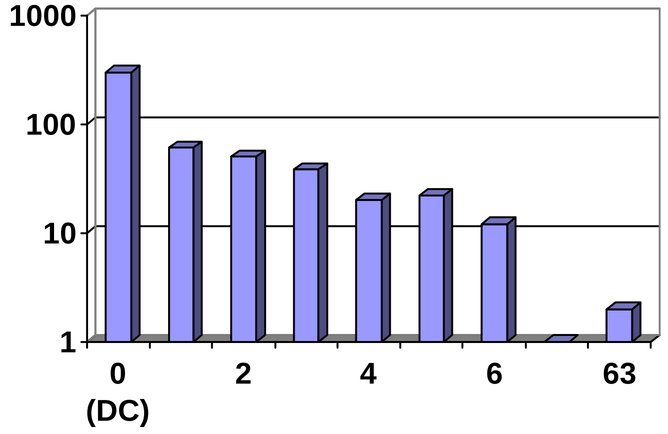
$$\mathbf{Y} = \mathbf{C}^T \mathbf{X} \mathbf{C}$$

where $\mathbf{X}$ is an $N \times N$ image block, $\mathbf{Y}$ contains the $N \times N$ DCT coefficients, and $\mathbf{C}$ is an $N \times N$ matrix defined as

$$C_{mn} = k_n \cos\left[\frac{(2m+1)np}{2N}\right] \text{ where } k_n = \begin{cases} \sqrt{1/N} & \text{when } n = 0 \\ \sqrt{2/N} & \text{otherwise} \end{cases}$$

and $m, n = 0, 1, \cdots N-1$. Plot the basis functions for $N = 4, 8, \text{and } 16$ (as shown in Lecture 2: JPEG). *Hint:* Each basis function can be obtained by computing $\mathbf{X}$ with one element in $\mathbf{Y}$ to be one and all the rest to be zero.

Problem 2 (40 points) Write a program to compute DCT. Do not use software libraries or packages, as their definition of DCT may not be the same as above. The input to the program will be three video sequences available on the web site. Please see the `readme` file for the format of these video sequences. In general, each video sequence is composed of 10 pictures. Each picture is composed of the Y component at the full resolution, followed by the U and V components at the quarter resolution (downsampling in both the horizontal and the vertical directions). One pel is represented by one byte (`unsigned char` in C) and has value 0…255. The scanning order for each picture is from left to right and then top to bottom. With the block size 8×8, the output of your program should be a sequence of 64 floating point numbers, each representing the energy (mean square value) of the corresponding DCT coefficient, in the zigzag scan order as given in Lecture 2: JPEG. Instead of printing these numbers, plot them (in logarithmic scale) versus the coefficient index for each sequence and for Y, U, and V. In other words, you will have 9 plots that look like:

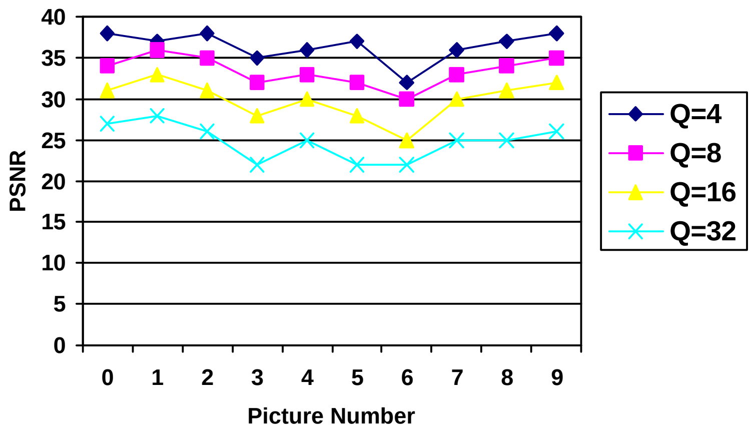


(OVER→)

Problem 3 (30 points) Use the same video sequences as in the previous problem. In addition to DCT, now write quantization and IDCT into your code. Use the quantizer and dequantizer defined in Lecture 2: JPEG (The rounding operator should be rounding to the nearest integer). Try quantization stepsize 4, 8, 16, and 32. After IDCT, round off the resulting pixel values to the nearest integer from 0 to 255. Let $F(x,y)$ be one picture in the original sequence and $\hat{F}(x,y)$ be the corresponding picture in the resulting sequence. The PSNR for this picture is defined as

$$10 \cdot \log_{10} \frac{255^2}{\frac{1}{\text{no. of pel}} \sum_{x,y} \left(F(x,y) - \hat{F}(x,y) \right)^2}$$

Plot the PSNR versus the picture number, for Y, U and V, and for each sequence. In other words, you will have 9 plots that look like:


PSNR
Picture Number
Q=4
Q=8
Q=16
Q=32