**CSC 545/645 Computer Speech, Music and Images**

**Exercise No. 8, Week 12, Due April 11, 2021**

**Lossy compression**

**Goals**

1. Experiment with lossy compression using the Discrete Cosine Transform (DCT)
2. Estimate degree of compression achieved using coefficient thresholding and quantization

**Background**

Download Ex08\_dct.pde from the Exercises folder on the class server. This program performs a DCT on an image. The program converts the image to grayscale and carries out a DCT by splitting the image into 8 x 8 blocks and performing a one-dimensional DCT on the rows, then transposing the matrix and performing the one-dimensional DCT again on the rows. Unscaled coefficients are stored in a two-dimensional array (coeffs[][]). The coefficients are also scaled to the range 0 to 255 and stored in an image for visualization. An inverse transform is carried out on coeffs[][] and the result (the restored image) is stored in another image.

The PImage array (img[]) is used as follows: img[0] holds the original RGB image; img[1] holds the grayscale image; img[2] holds the scaled DCT coefficients; img[3] holds the restored image. You will write a function that will store a difference image (img[3] – img[1]) in img[4].

Key presses display the images: ‘0’ displays the color image; ‘1’ displays the grayscale image; ‘2’ displays the scaled coefficients image; ‘3’ displays the restored image; ‘4’ displays the difference image.

**Procedure**

Write the function, PImage diff(PImage img1, PImage img2),that creates an image showing the difference between two images. It will be passed the grayscale image and the restored image, and will return an image showing the absolute value difference between the two. In order to see anything in the difference image, you will probably have to scale its pixel values. A rudimentary way to do that would be to shift the pixel values left some number of bits but, as you mask and quantize coefficients, you will get values that could overflow. To deal with this, find the maximum difference value and print it to the screen so you’ll know whether you have very small differences or some sizable ones. Put in code to scale the pixel values to 255 (multiply each pixel value by 255.0/maxValue or use the map() function).

Throw away some DCT coefficients using a zone mask. The program is initially set to use zmask1, defined in the quant tab, which does not throw away any coefficients. The zone mask is applied in dct(), when the coefficients are put into the coeffs[][] matrix. Try using zmask0, then vary it, masking out more or fewer coefficients. Note how the different masks affect the difference images. Estimate your compression (assume 8 bit grayscale) – how many coefficients can you throw away and still get good quality?

Now try quantizing your coefficients using the q tables. The program is initially set to use q100, which does no quantization. Try q50 and q80 tables. Here, you’ll have to apply the quantization in dct(), when the coefficients are put into coeffs[][], but you’ll also have to restore the values in idct(), when the coefficients are used during the inverse transform (don’t forget to use integer division on the forward transform quantization). Note how quantization affects the difference image.

In both your zone mask and quantization experiments, note where the largest differences are in the restored image, and where you can see image degradation. Try several different images. Can you determine which kinds of image features degrade most easily? Can you quantify types of images that are particularly good for DCT compression, or particularly bad?