Anitha Thanam

Zh9479

# Program #3

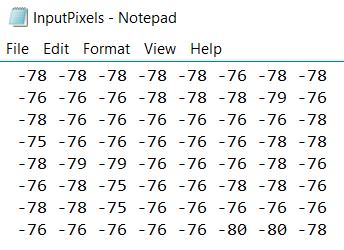
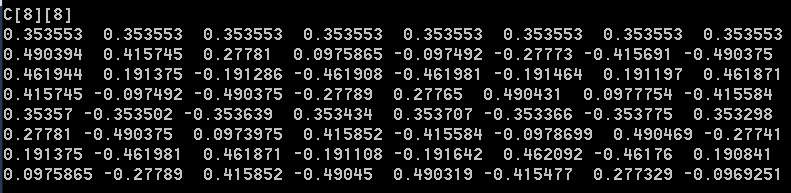
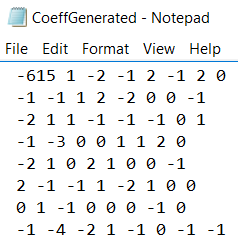
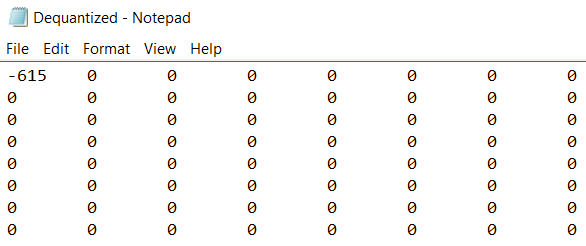
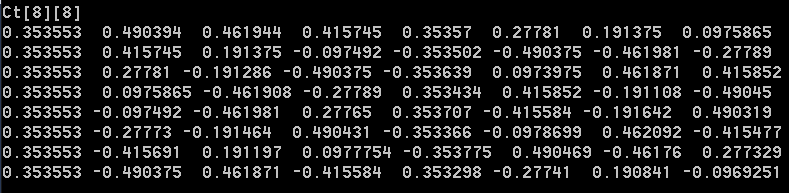
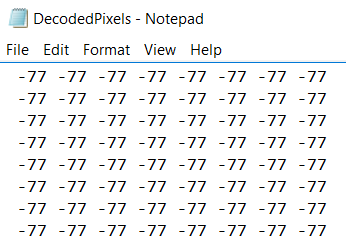
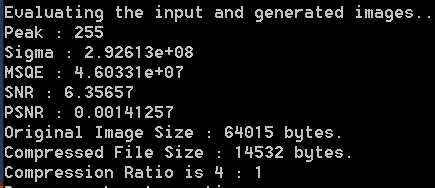
# CS 6715 Data Compression

**A description of algorithm.**

Main method in DCT.cpp is the entry point of the application and it does the following:

* Read the input image and determine the pixels.
* Initializes the 8x8 matrix co-efficients
* Calls the DCTEncoder method in dctCoder.cpp to encode using the generated co-efficient matrices. Post the forward transform, values less than 15 are eliminated. This THRESHOLD value is configured in stdafx.h file.
* Calls the quantize method in Quantizer.cpp  
  Here, a simple uniform quantization method is used where the delta value is pre calculated and configured in stdafx.h file. This value is then divided with each values.
* Calls the HuffmanEncoder method in HuffmanCoder.cpp to compress.  
  A minimum variance Huffman tree is constructed from the distinct values of the quantized file content and their frequencies and probabilities are calculated. And, codewords for each value is generated.   
  For each value, the codeword is packed in binary format to the compressed file.
* Calls the HuffmanDecoder method in HuffmanCoder.cpp to de-compress  
  Using the existing codewords, the compressed file is unpacked.
* Calls the dequantize method in Quantizer.cpp  
  All the values are dequantized using the same delta value.
* Calls the DCTDecoder method in dctCoder.cpp to decode  
  Inverse transform is applied to the dequantized values.
* Write the decoded pixels to an image file
* Evaluate the input and generated images by calling EvaluateImages method in QuantitativeMeasure.cpp  
  Compression ratio, Sigma, MSQE, SNR, PSNR are evaluated between input image and restored image.

**Description of how the program performed on the two assigned input files**

1. A description of the contents of each input file - what is the image a picture of?   
     
   Both images are grayscale images with 320x200 pixels, each pixel ranges from 0 to 255 (uses 1 byte per pixel). Rose.pgm has an image of a rose and 3 leaves in a black background. Mouse.pgm has a front view of a white mouse in a gray/black background.
2. For the first block in the **encoder** : Output the original pixel values after scaling,  output the transform coefficients, output the quantized coefficients.    
     
     
     
     
     
   
3. For the first block in the **decoder** : Output the quantized coefficients, output the transform coefficients, output the pixel values. (This is for debugging purposes). 5pts  
     
     
     
     
     
   
4. The achieved compression ratio (after losslessly encoding the coefficients).   
     
   
5. A description of the quality of the reconstructed images.  Could you see any errors?   
     
   The quality of the reconstructed images is directly proportional to the selected threshold value. Eliminating more lower co-efficient, the quality becomes lower and increasing the compression ratio.  
     
   For example,  
     
   Original image  
     
      
   Recontructed image with Threshold 25  
     
     
   Reconstructed image with Threshold 50  
   