Communication and Information Engineering Program
Information Theory and Coding (CIE 425) Fall 2019
Project Part 1

Image Compression

Construct a team of 4 members and work together on the following tasks:

JPEG
compression

You are required to develop a JPEG encoder and decoder any grey-scale image of your choice. You can read it, e.g., in Matlab using the following command: Image_pixels = imread('Test.jpg','jpeg'); The variable Image_pixels in Matlab (after executing the imread command) will contain NxM pixels with values between 0 (black) and 255 (white), i.e., 8-bit grey scaled pixels. For simplicity, you can crop the image to be a square image, e.g., NxN

Some notes on the Discrete cosine transform (DCT) that you might find useful

1- Please use the DCT basis functions provided in the text book

2- Divide the DCT coeffs at (u=0,v=0) by 64 3- Divide the DCT coeffs with either u = 0, or v = 0 (not including the (0,0) coeff) by 32 4- Divide the remaining DCT coeffs by 16
5- To perform the IDCT operation, you should just multiply each DCT coefficient by the corresponding basis function, and then add them without any further scaling The above notes are to ensure that the output after performing DCT and IDCT restores the original pixel values. You should check on a single 8x8 block your DCT and IDCT codes and make sure that this is the case

You are required to perform the following steps using your own developed code:

1- Divide the image into blocks of 8x8 pixels 2- Perform DCT on each block (develop your own DCT, don't use the ready-made function in Matlab) 3- Perform the quantization step per 8x8 block using the below mentioned

quantization tables (You should try both tables and compare the compression ratio and performance of each)

4- Transform each block from 2-D into 1-D vector using the following pattern

5- Use run-length encoding to compress the stream of zeros that may results due to the quantization (use your own developed code) 6- Develop your own Huffman encoder function in Matlab to encode the final stream into a further compressed stream.

Now you should have completed the encoding operation, let's start the decoder 7- Use your developed Huffman decoder to decode the Huffman encoded stream 8- Perform run-length decoding (use your own developed code) 9- Transform the 1-D vector into groups of 8x8 matrices 10- Multiply each group by the quantization table 11- Perform IDCT using your own developed function on each 8x8 pixel group 12- Combine the 8x8 pixel groups into a single image and save it back to a file named 'Compressed_Image'. You can use the following command to save the compressed image pixels: imwrite(Compressed_Image _pixels, Compressed_Image.jpg','jpeg'); 13- Compare the original image with the compressed image when using each of the quantization tables used in step (3) 14- Repeat the steps from 1-13 when using a different DCT block size, e.g., 16x16 instead of 8x8. Develop two new Quantization tables that fit the new block size. Make sure to perform the correct scaling in the DCT operation. 15- Compare the 4 codecs that you developed (two different DCT block sizes with two

quantization tables each) in terms of the following:

a. Amount of achieved compression b. Number of floating-point operations (additions, subtractions,

multiplications, and divisions) required to perform the encoding and decoding c. Quality of the compressed image when compared to the original image. Choose a metric of your choice, e.g., means squared error (MSE), or any other metric.

Deliverables for JPEG compression:

1- Your own developed JPEG encoder and decoder MATLAB (or other language) code

with all the previous steps clearly implemented as indicated above. The correctness of the code will be graded. 2- Your code should be modular, readable, and re-usable. This will also be graded. 3- A documentation of your code is required. In the documentation you should explain

clearly all your used modules (functions) and their corresponding inputs, outputs, internal variables, etc., and how they map to the JPEG algorithm. 4- The images before and after compression. After compression you should have 4

versions of the image (two DCT block sizes x two quantization tables)

JPEG 2000-like compression

Starting from the same image used in JPEG, perform the following tasks:

1- Perform DWT (analysis phase) on the image using 2 different decompositions 2- For each decomposition, perform 2 different quantization levels. Each DWT sub-band (e.g., LL, LH, etc.) should use a different number of bits. You are free to quantize each sub-band using the method you find appropriate. 3- Perform entropy coding on the quantized stream after converting from 2D to 1D.

You can use the Huffman code from the JPEG part.

At the decoder, you are required to:

1- Perform entropy decoder 2- Perform the inverse DWT, i.e., the synthesis (reconstruction) phase

Deliverables for JPEG 2000-like compression:

1- Your own developed encoder and decoder MATLAB (or other language) code with all

the previous steps clearly implemented as indicated above. The correctness of the code will be graded. 2- Your code should be modular, readable, and re-usable. This will also be graded. 3- A documentation of your code is required. In the documentation you should explain

clearly all your used modules (functions) and their corresponding inputs, outputs, internal variables, etc., and how they map to the compression algorithm. 4- The images before and after compression. After compression you should have 4 versions of the image (two DWT decompositions x two quantization levels)

Deadline:
Sunday 3/11
11:59 PM