Image Compression

Overview

For this CS517 programming assignment, use the images from this dataset: Kodak Lossless True Color Image Suite. Convert to grayscale using inbuilt MATLAB or Python function, if needed. Use zero padding if there is a need for padding on the last column/row side.

In addition to the dataset, use images uploaded on Google Drive. The analysis table should be based on all 24 images, but you may show only some sample images in the report.

Part 1: JPEG Compression

For analysis, you can vary: the number of coefficient parameters sent, sub-image/block size, normalization/quantization matrix, etc. By default, consider all coefficients sent unless the sequence of zeroes encountered, 8 x 8 size block size, and use the standard quantization matrix as provided in the book/lecture slides.

It is recommended to use color image jpeg compression by applying RGB -> YUV (or, say, YCbCr) conversion, as mentioned in the slides. You may also refer to YUV - Wikipedia. For PSNR calculations (in the case of color images), use the RGB color space only.

For other block sizes quantization matrices (as needed for analysis), resize the standard quantization matrix and use rounded integer values. Mention the obtained quantization matrices.

For no. of coefficient parameters variation, use 1 (i.e., only 1 coefficient sent only), 3, 6, 10, 15, 28 and all in zigzag order. Consider 8x8 block size and standard Q matrix. Show the PSNR table for all images and average (+normalized) PSNR variations via graph, considering both grayscale and color images as input. Also, show the curves for the compression ratio.

Try to have a common plot showing both the average compression ratio and average PSNR. Try to prepare the report accordingly.

Codes:

JPEG_encoder:

Input:

image filename, block size, # of Coefficient parameters sent, and Color/grayscale binary input.

Output:

JPEG encoded integer array. This shall also be written in some file, e.g. img1_jpgcode.txt.

JPEG_decoder:

Input:

all 4 inputs, as mentioned above, JPEG encoded integer arrays.

Output:

Reconstructed image, RMSE, PSNR, and compression ratio.

Assume the standard Q matrix is known and could be hard coded.

Part 2: LZW Compression

For this part, you may consider only grayscale images. In addition to the dataset, use images uploaded on Google Drive. To make it easier for all of you, ignore the class discussion of considering 7-bit inputs and 8-bit (LZW) codes, assume now input image pixel values are 8-bit only (i.e., range 0-255) and codes are stored by you as short/integers.

For mathematical analysis, we will assume code size to be as specified in that part, e.g., 12-bit codes or 10-bit codes. Assume the code table size is 4096 by default, i.e., 12-bit codes, and also assume the image is compressed fully, i.e., there is no sub-images/block extraction unless specified. If the dictionary gets filled (e.g., all 12-bit codes are made), then assume no new codes will be added.

(Ref: book, slides, weblink1, …) Include in the report your analysis of images uploaded on Google Drive. The compression ratio achieved, entropy, max compression achievable, etc., your observation on some images. Note again: as LZW is lossless, PSNR etc., analysis is unnecessary.

Codes:

LZW_encoder:

Input:

image filename, block size, Codes size.

Output:

LZW coded file, max value of any code, i.e. used, # of codes, compression ratio achieved, Avg. length of encoded pixels, Entropy.

(LZW coded filename, e.g. img1_lzwcode.txt. One row for one block and assume the first row has 3 integers: height and width values of the image and block size)

Assumptions/constraints:

Block size will be 4, 8, 16, 32, 64, 128, and 256. And -1 indicates the whole image.

Codes size is in bits and is mandatorily greater than 8. E.g. 9, 10, 11, 12, 16.

Each block is separately coded (i.e. no impact of one block on the other in terms of coding).

LZW_decoder:

Input:

LZW coded filename (Image height, image width info in file first row).

Output:

Reconstructed grayscale image.