HOMEWORK ASSIGNMENT

NAZARBAYEV UNIVERSITY | SCHOOL OF SCIENCE AND TECHNOLOGY

PROJECT 5

In this project, you will be working on developing an image compression and decompression algorithm which will be evaluated according to a specific metric combining execution time and compression ratio. The assignment will allow you to advance your understanding of image compression, associated mathematical background as well as Matlab, C/C++, or Python programming skills.

DUE DATE

Tuesday, 17th of April

METHOD OF DELIVERY

Assignment deliverables should be submitted via Moodle to the course instructor before the due date.

LEVEL OF COLLABORATION ALLOWED

Collaboration is not allowed on this assignment – each student should perform the assignment individually.

ESTIMATED TIME FOR COMPLETION

30 hours

ADDITIONAL SUPPORT

Please contact the course instructor if you need any assistance or have any concerns about this assignment.

GRADING CRITERIA

- 60% - implementation, functionality and documentation of the compressor and decompressor
- 40% - based on student ranking provided using a metric defined in the assignment specification

ASSIGNMENT DETAILS

Efficient storage and transmission of image data using compression is an essential subject in image processing. This assignment fortifies the knowledge of image compression with relevant mathematical background, as well as Matlab/C/C++/Python programming skills.

You are required to develop a compression algorithm. Your compressor and decompressor code can be tested on a set of 16 8-bit grayscale bitmap images which can be downloaded from this link (https://drive.google.com/file/d/1Sh_SX7vXUyvu119l4epUuCEyGPUhNa1j/view?usp=sharing). Bitmap images are uncompressed and reflect the actual size of an image, which you should use for calculating your compression ratio (*size of input image / size of compressed image*). The implementation and functionality of the algorithm as well as associated documentation contribute 60% to the total assignment grade. You are welcome to implement any of the compression algorithms which you have studied in class (Huffman, Golomb, Arithmetic, bit-plane, run-length coding, LZW). For this a student is encouraged to study the presented algorithms beyond the scope of the in-class lectures. Understanding advantages and disadvantages of the methods allows making the most optimal choice (for example LZW requires good programming skills, and RLC works best for certain intensity patterns which might or might not be present in an image). Importantly, students are encouraged to use combination of compression algorithms in order to obtain the best results. However, a student is still in position of obtaining full grade out of 60% component for implementing a single compression technique correctly.

There are two fundamental criteria defining the quality of image compression – compression ratio and time required for a single image compression. These two will comprise the metric by using which your compressor quality will be evaluated. You need to measure the ratio and time spent per image compression yourself and provide these numbers. There is a requirement on the compression time per image *no more than 2 minutes*. The evaluation of the work will be accomplished on a second dataset possessed by the course instructor and not available to you. The numbers for compression ratio and compression time provided will also be confirmed on the second dataset. Compressor quality defined at this stage will generate a student ranking within the class and corresponding grading scheme associating the ranks with grades out of 40% which the component contributes to the total assignment grade. The ranking would be done sorting from highest to lowest:

Score = (size of input image / size of compressed image) / (1.0 + time)

This year we are allowing to use **lossy** compression. You can use any method you want (for example remove high frequency components using FFT or disregarding the least significant bits in the image), yet you should take into account that lossy compression lowers the quality of the image. One of the objective

fidelity criterion is root mean square error (see equation (1)), N and M are dimension of a matrix. Such that we can still obtain good quality images after compression we allow you to have root mean square error (RMSE) on average to be **5**.

$$RMSE = \sqrt{\frac{1}{NM}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}(\hat{f}(x,y) - f(x,y))^2} \qquad (1)$$

16 images are provided in the archive *images.zip* archive as well as a starter code for MATLAB/C/C++/Python program. Remember that you work with 8-bit grayscale format only. If you have chosen MATLAB, you are required to submit two m-file:

1. *robt310_yoursecondname_compress.m* - this will write a compressed image to a specified file;
2. r*obt310_yoursecondname_decompress.m* – this will accept the filename of the compressed image file, decompress and save it as a BMP file.

Each function should follow the input/output structure as follows:
robt310_yoursecondname_compress(input_filename, output_filename);
robt310_yoursecondname_decompress(input_filename, output_filename);

Example usage in MATLAB:
>> robt310_yoursecondname_compress("test.bmp", "test.pku");
>> robt310_ yoursecondname_decompress("some_othertest.pku", "decompressed.bmp");

For C/C++ you should submit all your code and a build script (for example Makefile). As for Python please list in a separate file which additional packages you used with the exception of NumPy and OpenCV.
For every language we will have a test script/program that will check your solution. For decompression routine make sure that you save it as BMP eventhough the extension would be different (for example .dat).
Please note, that your compressor function should record the data into a file with '.pku' extension in addition to the regular output parameters, while decompressor function should be able to read it. Remember, that in this assignment we can do **lossy** compression.