Lab 6 – JPEG Compression

Goal: Introduction of principles of the JPEG baseline coding system.

Introduction

The JPEG standard provides a powerful compression tool used worldwide for different applications. This standard has been adopted as the leading lossy compression standard for natural images due to its excellent compression capabilities and its configurability. In this lab we will present and implement the basic concepts used for JPEG coding and experiment with different coding parameters.

JPEG baseline coding algorithm (simplified version) [1]

The JPEG baseline coding algorithm consists of the following steps:

1. The image is divided into 8×8 non-overlapping blocks.
2. Each block is level-shifted by subtracting 128 from it.
3. Each (level-shifted) block is transformed with Discrete Cosine Transform (DCT).
4. Each block (of DCT coefficients) is quantized using a quantization table. The quantization table is modified by the "**quality**" $q \geq 1$ factor that controls the quality of compression. Increasing q results in improved compression at the expense of more data loss.
5. Each block (of quantized DCT coefficients) is reordered in accordance with a zigzag pattern.
6. In each block (of quantized DCT coefficients in zigzag order) all trailing zeros (starting immediately after the last non-zero coefficient and up to the end of the block) are discarded and a special End-Of-Block (EOB) symbol is inserted.

Remark: In the real JPEG baseline coding, after removal of the trailing zeros, each block is coded with Huffman coding, but this issue is beyond our scope. Furthermore, in each block all zero-runs are coded and not only the final zero-run. Each DC coefficient (the first coefficient in the block) is encoded as a difference from the DC coefficient in a previous block [5].

Preliminary report

Answer the following questions:

1. Why was DCT chosen as transform domain for JPEG? What are the advantages of DCT over other transforms, e.g. DFT?

   Hint: how does DFT and DCT treat a periodic extension of a signal and what are consequences of these extension methods on image border processing?

2. Use the unnormalized DCT for this question:

$$X_N^{(DCT)}[k] = 2\sum_{n=0}^{N-1} x[n]\cos\left(\frac{\pi k(2n+1)}{2N}\right) \quad \text{(i)}$$

   The unnormalized DCT can be efficiently computed using FFT as seen in [6]. Prove:

$$X_N^{(DCT)}[k] = 2Re\left\{e^{-\frac{j\pi k}{2N}}\tilde{X}_{2N}^{(DFT)}[k]\right\}\ (ii) \quad \text{(ii)}$$

   where $\tilde{X}_{2N}^{(DFT)}$ is the length $2N$ one dimensional DFT of the zero padded signal:

$$\tilde{x}[n] = \begin{cases} x[n], & n = 0, \dots, N-1 \\ 0, & n = N, \dots, 2N-1 \end{cases}$$

   Hint: see [6].

3. (*Programming*) Implement DCT using both the direct method (i) and FFT (ii).
   a. write functions ***my_dct(x)*** and ***my_dct_fft(x)***, computing the DCT of the input signal ***x*** following (i) and (ii), respectively. Notes:
      - You may use FFT package functions (***scipy.fft.fft***, etc.) in your solution.
      - Follow the normalization mode used in ***scipy.fftpack.dct***, ***scipy.fft.dct*** and make sure your functions output the same results.
      - Bonus (5pts): Implement without loops.

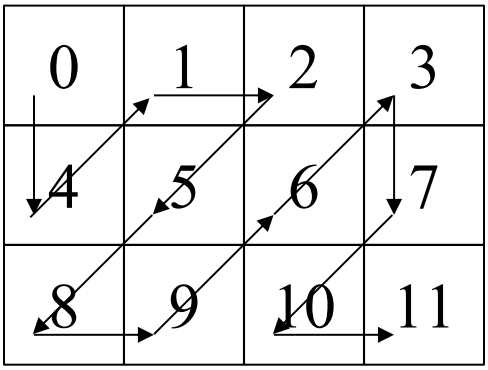
b. Generate a random 1D signal ***x*** of length $N = 16$. Plot on the same graph:
   - DCT using ***my_dct(x)***
   - DCT using ***my_dct_fft(x)***
   - DCT using ***scipy.fftpack.dct(x)*** setting norm='ortho'.

c. Verify numerically that the 3 signals are equal (error below 10^{-10}).

4. (*Programming*) Implement zigzag ordering pattern of a matrix of arbitrary size. Namely, write a function ***zigzag(M,N)*** which returns a list of the indices of a **row-stacked** $M \times N$ matrix, ordered according to the zigzag pattern, starting at the upper left point, going downwards first and then right (see example).

   Note: Although slightly different from the one shown in lectures, we use this pattern for easier implementation.

   Example: Consider a 3×4 matrix ($M = 3, N = 4$):



   The indices of the row-stacked matrix are given in every matrix cell. Reordering the indices according to the required pattern would result in:

$$[0,4,1,2,5,8,9,6,3,7,10,11]$$

   Therefore, calling ***zigzag(M,N)*** should return the above list of indices.

   Test your function for various values of M, N and specifically make sure your function works properly for the 4×4, 8×8, 16×16 cases as these will be used in the experiment.

Description of the experiment

Open the Jupyter notebook supplied for Lab 6 and follow the instructions, use image of your choice.

Final report

Submit the results of testing and demonstrations from the previous section of the experiment, with the image of your choice. Explanation your results in each step and submit all the graphs and images.

Good luck!

References

[1] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*. Prentice-Hall, Inc., 2002, Second Edition (Library Dewey number 621.368 GON).

[2] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*. Addison-Wesley Publishing Company, Inc., 1992 (Library Dewey number 621.368 GON).

[3] A. K. Jain, *Fundamentals of Digital Image Processing*. Prentice-Hall, Inc., 1989 (Library Dewey number 621.368 JAI).

[4] G. K. Wallace, “The JPEG still picture compression standard,” *IEEE Trans. Consumer Electron.*, vol. 38, no. 1, pp. xviii-xxxiv, February 1992.

[5] G. K. Wallace, “The JPEG still picture compression standard,” *Commun. ACM*, vol. 34, no. 4, pp. 30-44, April 1991.

[6] J. Makhoul, "A fast cosine transform in one and two dimensions," in IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. 28, no. 1, pp. 27-34, February 1980, doi: 10.1109/TASSP.1980.1163351.