Analysis of JPEG codec

Fiala Péter

Mediacommunications Technologies Laboratory

1 JPEG compression step-by-step

1.1 Block scheme

JPEG Compression is performed following the main steps described below: (see Figure 1 for illustration):

1. The RGB-coded image is transformed into YCbCr color space.

$$Y = 0.2126R + 0.7152G + 0.0722B \quad (1)$$

$$Cb = B - Y \quad (2)$$

$$Cr = R - Y \quad (3)$$

2. As the human eye is less sensitive to small scale details in the color information than to that of the luminance channel, therefore the Cb and Cr signals are downsampled (following spatial low pass filtering in order to avoid aliasing).
3. The obtained three channels are split up into blocks of equal size (8x8), called macro blocks. The next transform steps are performed for each macro block independently.
4. The macro block is transformed into spatial frequency domain by means of a spatial 2D Discrete Cosine Transform. A transformed macro block's (0,0) (left upper) pixel contains the mean level of the original macro block, while pixels with a higher index contain the weights of high frequency components.
5. The higher order components are requanized, meaning that they are coded with a reduced number of bits.
6. The obtained bit sample is further compressed using a lossless compression (Huffmann-coding), and the JPEG file contains the Huffmann-coded bitstream. Further details of the lossless compression are not targeted in the present laboratory exercise.

The losses in the JPEG transform are influenced by two parameters: The downsampling rate of the chroma channels $(1, 2, 4)$, and the requantization depth.

1.2 The Discrete Cosine Transform

The Discrete Cosine Transform transforms the N samples of the input signal x_k into N samples of the transformed signal X_n by the definition:

$$X_k = \sum_{n=0}^{N-1} x_n \sqrt{\frac{2}{N}} \cos\left(\frac{\pi}{N}\left(n + \frac{1}{2}\right)k\right), \quad k = 0, \ldots, N-1 \quad (4)$$

The DCT is easily represented as a matrix-vector product:

$$\begin{Bmatrix} X_0 \\ X_1 \\ \vdots \\ X_{N-1} \end{Bmatrix} = \left[\sqrt{\tfrac{2}{N}} \cos\left(\tfrac{\pi}{N}\left(n + \tfrac{1}{2}\right)k\right)\right] \begin{Bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{N-1} \end{Bmatrix} \quad (5)$$

where the first ($k = 0$) row contains constant values $\sqrt{\frac{2}{N}}$, the following rows contain the samples of cosine functions with increasing frequency, as displayed in figure 2. The inverse transform is defined

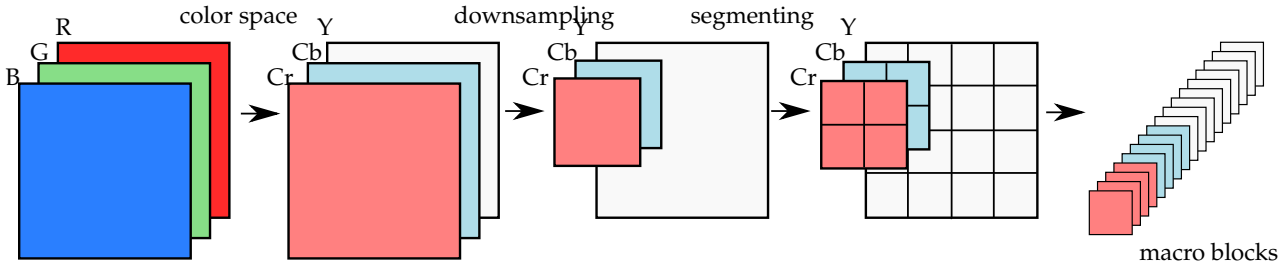

Figure 1: Block scheme of JPEG coding

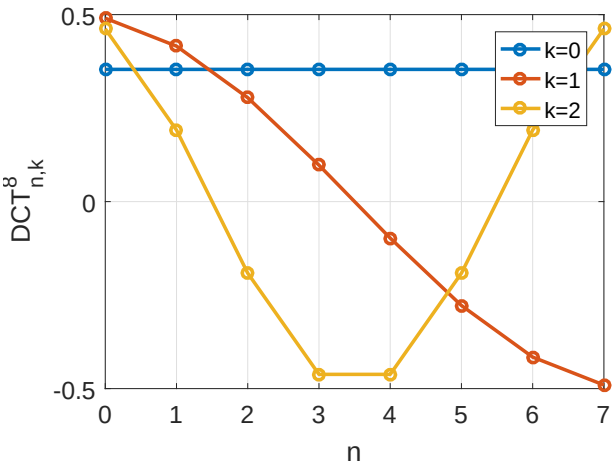

Figure 2: The first three base functions of the 8 point DCT (samples from the DCT matrix's first three rows)

by the formula

$$x_n = \sum_{k=0}^{N-1} X_k \sqrt{\frac{2}{N}} \cos\left(\frac{\pi}{N}\left(n+\frac{1}{2}\right)k\right), \quad k = 0,\ldots,N-1 \tag{6}$$

that clearly indicates that the input signal x_n is reproduced as the superposition of cosine functions of increasing frequency. As the transform is orthonormal. the DCT matrix's inverse equals its transpose.

For the case of the JPEG transform, the x_{mn} macro blocks are transformed along both dimensions:

$$x_{mn} \to X_{lk} \tag{7}$$

As a 0-th step of the transform, the pixel values of the macro block that originally vary between 0 and 255, are shifted by -128, so that an approximately zero-mean signal is subjected the cosine transform.

2 Requnatizing

The transformed macro blocks are requantized. Requnatiztazion is by definition performed by using matrices that contain increasing values for increasing indices:

$$Q_{lk} = \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix} \tag{8}$$

The transformed macroblock X_{lk} is requantized by evaluating the values $X_{lk}/(c \cdot Q_{lk})$, and rouinding them to the nearest integer. As a result, higher frequency components are stored with less precision. Parameter c defines the rate of requantization.

3 Analysis of transforming coder

3.1 Coding – Decoding

The attached Matlab files contain the skeleton of a JPEG coder and decoder, as well as a testing script, that encodes and decodes a given image file, and compares the original to the result.

1. Discuss the coding and decoding algorithm with the lecturer.
2. Implement the block that is responsible for resampling the chroma signals. This involves downsampling at encoding phase and interpolation at decoding.
   (a) Implement both methods using smoothing by a simple rect window.
   (b) Implement both methods using an ideal low-pass filter.
3. Implement the coder block that is responsible for the 8×8 DCT transform. The implementation should compute the 1D DCT-t as a matrix-vector product
   $$\mathbf{d} = \mathbf{D}\mathbf{x} \tag{9}$$
   , where $\mathbf{x}$ is the original sample vector, $\mathbf{d}$ denotes the DCT-transformed values.
   (a) How do you compute the 2D DCT with matrix $\mathbf{D}$?
   (b) How do you compute the inverse DCT? Do you need the inverse of matrix $\mathbf{D}$?
4. Implement the PSNR function that is used to compare the two images.
5. Investigate how the resampling of the chroma channels influences the subjective quality of the image. Investigate the same with the requantizazion depth.
6. Investigate the coding with blocks containing higher number of pixels $(16, 32)$. Where do You need to modify the original implementation? Can you reuse your above implemented functions for the modified algorithm?

3.2 Image manipulation in the coded domain

Some image manipulations can be performed in the coded domain. This is advantageous, as the round-off errors of the decoding and encoding process can be avoided.

1. Display a grayscale index image of the coded image. Implement a function that extracts the preview from the coded structure.
2. Compute the mirrored image in the transformed domain. Hint: The order of blocks needs to be modified (flipud, fliplr). The inner content of each block can be flipped by realizing that even DCT components are the weights of even functions, while odd index componenets are the weights of odd functions.