Lecture ”Digital Signal Processing”
Prof. Dr. D. Klakow, summer term 2014

Tutorial Nr. 2
Submission: 26.05.2014 by e-mail to your tutor

Note: You have one week to solve the tutorials. The theoretical part should be submitted before the lecture.
The code should be well structured and documented. Do not use any Matlab-Toolbox if not mentioned that you could use it.
You are allowed to hand in your practical solutions in groups of two student.

1 Exercise

You will implement some basic parts of JPEG-Coding in this exercise. Please find the file “cola.jpg” on our website as an example image and include the “blockproc.m” into your solution’s directory.
Furthermore, add a textfile “answers” which contains answers to the questions appearing in this exercise.

About blockproc
It is common to implement the general function blockproc. This function splits the given matrix \( I \in \mathbb{R}^{n \times m} \) into submatrices with size \( b \in \mathbb{N}^2 \) and applies the delivered function \( \text{fun} : \mathbb{R}^b \to \mathbb{R}^b \) to each of the submatrices. Finally, it returns a matrix \( O \in \mathbb{R}^{n \times m} \) out of all the submatrices. Find an example usage in the “blockproc.m” file.
(You can use “@” to deliver functions as reference; Matlab-Help: “function_handle”)

\[
\text{function } \ [O] = \text{blockproc}(I, b, \text{fun})
\]

1.1 (2P) Subtask

The first step in the process is to apply a color transformation, which takes the image from \( RGB \) to \( YC_bC_r \) color space. The following matrix-vector multiplication transforms an \( RGB \) vector into a \( YC_bC_r \) vector. Implement the function colortrans, that transforms a given \( RGB \) — image to a \( YC_bC_r \) — image and vice versa.

\[
\begin{pmatrix}
Y \\
C_b \\
C_r
\end{pmatrix} =
\begin{pmatrix}
0 \\
128 \\
128
\end{pmatrix} +
\begin{pmatrix}
0,299 & 0,587 & 0,114 \\
-0,168736 & -0,331264 & 0,5 \\
0,5 & -0,418688 & -0,081312
\end{pmatrix}
\begin{pmatrix}
R \\
G \\
B
\end{pmatrix}
\]  

(1)

1.2 (2P) Subtask

Use the colortransform from subtask 1.1 to transform “cola.jpg” to the \( YC_bC_r \)—color space. Now decompose the source image and the resulting image into their respective channels
and display all of them in a single window (6 pictures) for comparison.

(To show a matrix as an image with “imshow”, you have to convert its components to unsigned 8bit integers first, using “uint8”)

1.3 (2P) Subtask

An important step in the JPEG compression is to quantize blocks of the image. To do so, a function, that quantizes a matrix $M \in \mathbb{R}^{8 \times 8}$ using a given matrix $Q \in \mathbb{R}^{8 \times 8}$, comes in handy. Implement such a function by the name `quanmat` using the following common quantization matrix:

(A quantization in this case is a component-wise division of the two mentioned matrices)

$$Q = \begin{pmatrix}
8 & 16 & 19 & 22 & 26 & 27 & 29 & 34 \\
16 & 16 & 22 & 24 & 27 & 29 & 34 & 37 \\
19 & 22 & 26 & 27 & 29 & 34 & 34 & 38 \\
22 & 22 & 26 & 27 & 29 & 34 & 37 & 40 \\
22 & 26 & 27 & 29 & 32 & 35 & 40 & 48 \\
26 & 27 & 29 & 36 & 38 & 46 & 56 & 69 \\
27 & 29 & 35 & 46 & 46 & 56 & 69 & 83 \\
\end{pmatrix}$$

(2)

1.4 (2P) Subtask

The goal of this subtask is to treat the luminance channel $Y$ of the image according to the JPEG pipeline. Use the method “blockproc” and a block-size of $8 \times 8$ pixels to first apply a blockwise discrete cosine transform to the $Y$ channel. Now follow up with the method `quanmat` from subtask 1.3 to quantize the resulting matrix. Lastly, apply the inverse discrete cosine transform. The result will be referred to as $Y'$.

Display the original $Y$ channel, the result after the quantization, $Y'$ as well as the difference between $Y$ and $Y'$ in one window (4 pictures).

What do you observe? How do these steps help with compression? Do we lose quality?

(Make sure to have a function “newY” that transforms $Y$ to $Y'$ in one go, for later use. For the DCT you can use “dct2” and “idct2”)

1.5 (2P) Subtask

Another step in JPEG compression is downsampling of the $CbCr$—channels by a factor of $w$ in vertical and horizontal direction. Downsampling here means to assign the whole block of width and height $w$ the mean of all values in the given block. Implement a function that downsamples a given matrix given a blocksize $w \times w$.

How can this be used for compression?
1.6 (2P) Subtask

Using the previously obtained methods apply a lightweight JPEG compression, by first applying a color transform, computing $Y'$ and downsampling the $C_bC_r$–channels. Now transform back to the $RGB$ color space.

Display the source image as well as 3 more images using the following block sizes for the downsampling: 8, 16, 32