Lecture "Digital Signal Processing"

Prof. Dr. D. Klakow, summer term 2019

Tutorial 1

Submission deadline: 6.05.2019, 10:15

Submission Instructions:
You have one week to solve the tutorials.
The code should be well structured and documented. Do not use any Matlab-Toolbox or Python external libraries if not mentioned that you could use it.

- You are allowed to hand in your solutions in groups of two students.
- The theoretical part should be submitted before the lecture.
- For the practical tasks please submit files via the email address
  Tutorial 1: dsp.tutorial1@gmail.com Tutorial 2: dsp.tutorial2@gmail.com
- The subject of the letter should be [**DSP TUTORIAL X**]. **X** is the tutorial/assignment number.
- Rename and pack the main directory:
  Ex01_matriculationnumber1_matriculationnumber2.zip.

The directory that you pack and submit should contain the following files:

- code files;
- file "answers.pdf" which contains answers to the questions appearing in the exercise sheet;
- file "README" that contains an information on all team members:
  name
  matriculation number
  email address.

- Note: If you use Jupyter Notebook, you don't have to submit "answers.pdf".You can write your theoretical answer in the markdown area

1 Exercise

You will implement some basic parts of JPEG Compression (slide 23 chapter 2) in this exercise.
Note: You will first build the building blocks of JPEG compression Please find the file "cola.jpg" on our website as an example image and include the "blockproc.m" (For Python "Utility.py") into your solution's directory.

Furthermore, add a pdf file "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 $fun : \mathbb{R}^b \to \mathbb{R}^b$ 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")

```
function [O] = blockproc(I, b, fun)
```

For Python:

```
from Utility import blockproc
Output = blockproc(I, b, fun)
```

1.1 (2P) Optimal color space

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 two functions *colortrans* and *invcolortrans*, that transform a given $RGB-$image to a YC_bC_r-image and YC_bC_r-image to $RGB-$image respectively.

$$\begin{pmatrix} Y \\ C_b \\ C_r \end{pmatrix} = \begin{pmatrix} 0 \\ 128 \\ 128 \end{pmatrix} + \begin{pmatrix} 0,299 & 0,587 & 0,114 \\ -0,168736 & -0,331264 & 0,5 \\ 0,5 & -0,418688 & -0,081312 \end{pmatrix} \cdot \begin{pmatrix} R \\ G \\ B \end{pmatrix} \tag{1}$$

1.2 (2P) Downsample

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 *downsample* that downsamples a given matrix given a blocksize $w \times w$.
Also implement a function *upsample* that upsample a matrix. Use w = 4

1.3 (2P) Quantization

An important step in the JPEG compression is to quantize blocks of a 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. A quantization in this case is a component-wise division of the two mentioned matrices. Implement such a function by the name *quanmat* using the following common quantization matrix:

$$Q = \begin{pmatrix} 8 & 16 & 19 & 22 & 26 & 27 & 29 & 34 \\ 16 & 16 & 22 & 24 & 27 & 29 & 34 & 37 \\ 19 & 22 & 26 & 27 & 29 & 34 & 34 & 38 \\ 22 & 22 & 26 & 27 & 29 & 34 & 37 & 22 \\ 22 & 26 & 27 & 29 & 32 & 35 & 40 & 22 \\ 26 & 27 & 29 & 32 & 35 & 40 & 48 & 58 \\ 26 & 27 & 29 & 36 & 38 & 46 & 56 & 69 \\ 27 & 30 & 35 & 46 & 46 & 56 & 69 & 83 \end{pmatrix} \tag{2}$$

1.4 (2P) DCT

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 8×8 pixels to first apply a blockwise discrete cosine transform to the Y channel and downsampled C_bC_r. Now follow up with the method *quanmat* from subtask 1.3 to quantize the resulting matrices. This coefficients matrices are labeled as Y', C_b', C_r'. Lastly, apply the inverse discrete cosine transform(include inverse-quantization by elementwise multiplication). The result will be referred to as $newY, newC_b, newC_r$.

Display only original Y channel, $newY$ as well as the difference between Y and $newY$ in one window .

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

Hint: Make sure to have two functions "dct_quant" that transforms Y to quantizded coefficient of DCT of Y and and "idct_quant" that works complete opposite in one go, for later use. For the DCT you can use "dct2" and idct2". For Python Python: from scipy.fftpack import dct or ..idct

1.5 (2P)Encoding

The final step of JPEG compression is encoding. Apply Huffman encoding on Y', C_b', C_r'(They are the quantized DCT coefficient) individually. For simplicity, just flatten the matrix into a vector by concatenating all the rows together and then apply Huffman encoding (Note: You dont have to do zigzag ordering and RLE that is used in the jpeg compression).

Write two functions *encodemat* and *decodemat* for further utilization. You can generate the code table based the data you have. You can use *huffmanenco,huffmandeco,huffmandict* in Matlab. For Python use *dahuffman* 0.2 (pip install dahuffman).

Write the length of each encoded channels Y', C_b', C_r' in answers.pdf.

1.6 (2P) Decompressoin

Using the previously obtained methods you already applied a light weight JPEG compression, by first applying a colortransform, downsampling the C_bC_r-channels, applying

DCT in all the channels and finally encoding the DCT coefficients using Huffman encoding. Now transform back to the RGB color space and show the image. Explain why the quality is reduced during compression in 2 sentences
Use w = 4 in downsampling and upsampling step.