Lecture "Digital Signal Processing"

Prof. Dr. D. Klakow, summer term 2019

Tutorial 2

Submission deadline: 13.05.2019, 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 practical 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
  (Mon)Tutorial 1: dsp.tutorial1@gmail.com.
  (Thus)Tutorial 2: dsp.tutorial2@gmail.com.
- The subject of the letter should be [**DSP TUTORIAL 2**].
- Rename and pack the main directory:
  Ex02_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.

1 Exercise

The goal of this exercise is to implement a gaussian filter for (grey level) images.

1.1 (1.5P) Subtask

The boundary of an image is a problem for a gaussian filter. We will solve this problem by mirroring the image at the boundary.
We will assume odd, square convolution matrices K of size $h \times h$. How far must the image be mirrored if you use such a kernel?
Implement the function *imgmirror* that mirrors a strip of width w at the bounds of the matrix $M \in \mathbb{R}^{n \times m}$. You should get an output matrix $O \in \mathbb{R}^{(n+2 \cdot w) \times (m+2 \cdot w)}$.

Hint: you could use floor function here either in Matlab or Python

1.2 (2P) Subtask

Implement the function *gaussfilter*, which filters the matrix $I_1 \in \mathbb{R}^{n \times m}$ with kernel $K \in \mathbb{K}^{h \times h}$ (with standard deviation of σ and zero of μ). Use your function *imgmirror* to handle the boundaries. The matrix $I_2 \in \mathbb{R}^{n \times m}$ is the output of your function.

$$I_2(x, y) = \sum_{i=1}^{h} \sum_{j=1}^{h} I_1(x + i - a, y + j - a) K(i, j) \tag{1}$$

where $a = \lceil h/2 \rceil$.

What is the running time of your implementation, depending on K and I_1?

1.3 (1.5P) Subtask

Use the functions you implemented to denoise the corrupted picture *noisycoke.jpg*. Use a 5x5 smoothing kernel as discussed in the lecture.Vary parameter *sigma*. What do you observe? What is changed?

1.4 (2.5P) Subtask

Now, Using the separability property that we mentioned in the lecture to implement a new version of the function $gaussfilter$ .What is the running time of the new version? Compared to subatask 1.2, explain why it is faster?

2 Exercise

The operation we performed in task 1.2(applying a kernal/window function to each overlapping segment of an image/wave) is also called convolution. In pratical, the computation of convolution in time domain will be a problem (quadratic computational complexity). One can show that the convolution can be done easily in frequency domain.
Considering the continuous Fourier transfermation:

$$f(x) \xrightarrow{\mathscr{F}} \mathcal{F}[f] = \int_{-\infty}^{+\infty} f(x) e^{-i 2 \pi \omega x} dx \tag{2}$$

where ω is frequency. Prove the properties below:

2.1 (0.5P) Spatial shift:

$$\mathcal{F}[f(x-a)](\omega) = e^{-i2\pi\omega a} \cdot \mathcal{F}[f](\omega)$$

2.2 (0.5P) Convolution:

$$\mathcal{F}[(K * f)(x)](\omega) = \mathcal{F}[K](\omega) \cdot \mathcal{F}[f](\omega)$$

2.3 (0.5P) Derivative:

$$\mathcal{F}[\tfrac{\partial f(x)}{\partial x}](\omega) = i2\pi\omega \cdot \mathcal{F}[f](\omega)$$

$\star$ the properties are also hold for discrete signals, using continuous signal here is just for convenience.

3 Excercise

There is no cover about how to build a filter in the lecture. The goal of this Excercise is to give a simple example of building a first and second order derivative filter in 1-D.

3.1 (1.5P) Derivative Filter:

Considering the first order derivative of discrete signal in 1-D case as below:

$$f'[n] = \frac{f[n] - f[n-1]}{\tau} \iff \underbrace{\frac{1}{\tau}\boxed{\text{-1}}\boxed{\text{1}}}_{convolution\,kernal} * \boxed{\text{...}}\boxed{\text{f[n-2]}}\boxed{\text{f[n-1]}}\boxed{\text{f[n]}} \tag{3}$$

where $*$ is convolution and τ is the time changing from f[n-1] to f[n].

Is it a low pass filter, band pass filter or high pass filter? Write down the convolution kernel of second order derivative in 1-D following the same definition above. Where could we use this kind of filter?