Note: You have one week to solve the tutorials. The theoretical part should be submitted before the lecture. The practical part has to be send to your tutor in a common compressed file format until Monday.

There will be about 10 tutorials in two blocks (1-5) and (5-10). You can get 12 points for each tutorial. You need half of the maximal accessible points in each block in order to participate in the final exam.

The code should be well structured and documented. Do not use any Matlab-Toolbox if not mentioned that you could use it. You should be able to present your work in the Mechatronik-Cip-Pool.

(6P) Exercise 1

In this exercise you will implement an edge detector. To create a working edge detector you need to perform several steps.

- Convert a RGB image to a greyscale image.
- Convolve the greyscale image with two edge detection kernels, one for horizontal edge detection one for vertical edge detection.
- Crop the padding of the convolved images to restore the original image size.
- Combine the two filtered images.
- Apply some threshold to the resulting image to sharpen the edges.

(1P) Subtask 1.1

The `conv2` function of matlabs usually zero-paddes the images for convolving. This process is actually not very exact and distorts the result of the convolution at the borders of the images. Your task is to implement a function that replicates the borders of the image with respect to the size of the actual filter kernel. As an example the image I has size \((m \times n)\) and the kernel K has size \((g \times h)\) the convolved image will have size \((m + 2 \times g \times n + 2 \times h)\).

Implement a function `padImage(img,padSize)` that performs the padding by replicating image values at the borders.

Example for a matrix padded for a kernel of size \(2 \times 2\): \[
\begin{pmatrix}
1 & 1 \\
4 & 2
\end{pmatrix} \rightarrow \begin{pmatrix}
1 & 1 & 1 & 3 & 3 & 3 \\
1 & 1 & 3 & 3 & 3 \\
1 & 1 & 3 & 3 & 3 \\
4 & 4 & 4 & 2 & 2 & 2 \\
4 & 4 & 4 & 2 & 2 & 2 \\
4 & 4 & 4 & 2 & 2 & 2
\end{pmatrix}
\]
(1P) Subtask 1.2
Implement a function `cropResult(img, cropSize)` that crops the convolved image to the correct size after the convolution. Remember that the size of the convolved image in the one-dimensional case is $s = \text{length}(X) + \text{length}(Y) - 1$.

(2P) Subtask 1.3
In the lecture the Sobel operator was introduced for edge detection. Write a script that uses your `padImage` and the `cropImage` function from the previous tasks and the Sobel operators from the lecture to filter the images provided on the course page. In a second step combine the resulting horizontal and vertical images to create the gradient image. Now write a function `imgThresh(img, thresholdValue)` that uses a vector of threshold values to sharpen the edges of your gradient images. For our purposes one threshold is sufficient. Decide carefully on a value for this threshold.

(2P) Subtask 1.4
Write a second script that uses your implementation of salt-and-pepper noise from the first exercise sheet and apply noise to the original image (Chose a very low probability for the noise generator). Now again perform the edge detection from the previous task with the Sobel operator. Plot the original gradient image and the noisy gradient image in one plot. Now think about a new threshold level which would help to reduce the amount of noise in the new gradient image. Is it possible to remove the noise solely by applying a new threshold? What other methods are available?

(6P) Exercise 2
In this exercise you will implement a Gabor filter.

(1P) Subtask 2.1
Implement the function $G_{00}$ in the following way:

$$G_{00}(x, y, \sigma_x, \sigma_y, \omega) = \frac{1}{2 \cdot \pi \cdot \sigma_x \cdot \sigma_y} \cdot \exp \left( -\frac{1}{2} \cdot \left( \frac{x^2}{\sigma_x^2} + \frac{y^2}{\sigma_y^2} \right) \right) \cdot \cos(x \cdot 2 \cdot \pi \cdot \omega); \quad (1)$$

With $x, y \in \mathbb{N}^{n \times m}$ and $\sigma_x, \sigma_y, \omega \in \mathbb{R}$.

(1P) Subtask 2.2
Implement the function $G_{nm}$ in the following way:

$$G_{nm}(x, y, \sigma_x, \sigma_y, \omega, n, m, a, K) = a^{-m}G_{00}(x', y', \sigma_x, \sigma_y, \omega) \quad (2)$$

with

$$
\begin{pmatrix}
  x' \\
  y'
\end{pmatrix} = a^{-m} \begin{pmatrix}
  \cos(\theta_n) & \sin(\theta_n) \\
  -\sin(\theta_n) & \cos(\theta_n)
\end{pmatrix} \begin{pmatrix}
  x \\
  y
\end{pmatrix} \quad (3)
$$

and

$$\theta_n = \frac{n \cdot \pi}{K} \quad (4)$$

and $x, y \in \mathbb{N}^{n \times m}$ and $\sigma_x, \sigma_y, \omega \in \mathbb{R}$ and $n, m, a, K \in \mathbb{N}$. 

Questions: alikanso91@hotmail.com and dhhjx880713@163.com
(3P) Subtask 2.3

You can generate matrices $X, Y \in \mathbb{R}^{41 \times 41}$ with the following matlab function

$$[X, Y] = \text{meshgrid}(-2:.1:2, -2:.1:2);$$

Use $X, Y$ to generate a Gabor filter kernel $g_{n, 1} \in \mathbb{R}^{41 \times 41}$ with the following parameters:

\[
\begin{align*}
\sigma_x &= 0.25 & (5) \\
\sigma_y &= 0.1 & (6) \\
m &= 1 & (7) \\
a &= 2 & (8) \\
K &= 6 & (9) \\
\omega &= 2 & (10)
\end{align*}
\]

Explain the effect of the parameters (5) ... (10) and plot figures of all 6 filter kernels $g_{0, 1}, ... g_{5, 1}$ with $n$ in range 0 to 5.

(1P) Subtask 2.4

Filter the image provided on the course website with the filter kernel $g_{0, 1}, ... g_{5, 1}$ from subtask 2.3)\(^1\). What can you observe?

\(^1\)You can use the function “conv2” in Matlab. Think about the size of the filtered image.

Questions: alikanso91@hotmail.com and dhhjx880713@163.com 3