Exercise 01 for MA-INF 2201 Computer Vision WS24/25
13.10.2024
Submission on 20.10.2024

In this assignment, you are required to write code for the programming tasks listed below. The tasks should be completed using the opencv library, similarly to Exercise 0. Please submit your python file (*.py* or a Jupyter notebook) with clear comments indicating which part refers to which task. In addition, you are required to submit a separate file called *tasks 3_6.pdf* with answers for the theoretical questions 3 and 6.

1. **Rectangles and Integral Images**
Read the image *bonn.png* and convert it into a gray image.

- Compute and display the integral image without using the function *integral*.
- Compute the mean gray value of the image by:
  - Summing up each pixel value in the image, i.e. $\frac{1}{R}\sum_{p\in\mathcal{R}} I(p)$
  - Computing an integral image using the function *integral*
  - Computing an integral image with your own function
- Select 7 random squares of size 100×100 within the image and compute the mean gray value using the three versions. Output the runtime of this task for the three versions in seconds using *time*.

(*3 Points*)

2. **Histogram Equalization**
Read the image *bonn.png*. Convert the image into a gray image and perform histogram equalization:

- Using *equalizeHist*.
- Using your own implementation of the function *equalizeHist*.

and display both results. Compute the absolute pixelwise difference between the results and print the maximum pixel error.
(*2 Points*)

3. **Convolution Theorem**
Proof that convolutions are in the continuous case associative.
(*2 Points*)

4. **2D Filtering**
Read the image *bonn.png*. Convert the image into a gray image and display it. Filter the image with a Gaussian kernel with $\sigma = 2\sqrt{2}$

- Using *GaussianBlur*.
- Using *filter2D* without using *getGaussianKernel*.

- Using *sepFilter2D* without using *getGaussianKernel*.

and display the three results. Compute the absolute pixelwise difference between all pairs (there are three pairs) and print the maximum pixel error for each pair.
(*2 Points*)

5. **Multiple Gaussian Filters**
Read the image *bonn.png*. Convert the image into a gray image and display it. Filter the image:

- twice with a Gaussian kernel with $\sigma = 2$
- once with a Gaussian kernel with $\sigma = 2\sqrt{2}$

and display both results. Compute the absolute pixelwise difference between the results, and print the maximum pixel error.
(*1 Points*)

6. **More on Convolution**
Proof that convolution two times with a Gaussian kernel with standard deviation σ is the same as convolution once with a Gaussian kernel with standard deviation $\sqrt{2}\sigma$
(*2 Points*)

7. **Denoising**
Read the image *bonn.png*. Convert the image into a gray image, add 30% (the chance that a pixel is converted into a black or white pixel is 30%) salt and pepper noise, and display it. Filter the image by

- a Gaussian kernel
- stack blur the image by *stackBlur*.
- Bilateral filter *bilateralFilter*.

and display the three results. Select the filter size from the range $[1, 3, 5, 7, 9]$ that minimizes the mean gray value distance to the original image.
(*3 Points*)

8. **Separability of Filters**
Read the image *bonn.png*. Convert the image into a gray image.

- Filter the images using the two 2D filter kernels given blow
- Use the class SVD of OpenCV to separate each kernel. For the first kernel, use an approximation by taking only the highest singular value. For the second kernel, use an approximation by taking the first two highest singular values. Filter the images with the obtained 1D kernels and display the results.
- Compute the absolute pixel-wise difference between the results of (a) and (b), and print the maximum pixel error.

kernel 1

$$\begin{bmatrix} 0.0113 & 0.0838 & 0.0113 \\ 0.0838 & 0.6193 & 0.0838 \\ 0.0113 & 0.0838 & 0.0113 \end{bmatrix} \tag{1}$$

kernel 2

$$\begin{bmatrix} -1.7497 & 0.3426 & 1.1530 & -0.2524 & 0.9813 \\ 0.5142 & 0.2211 & -1.0700 & -0.1894 & 0.2550 \\ -0.4580 & 0.4351 & -0.5835 & 0.8168 & 0.6727 \\ 0.1044 & -0.5312 & 1.0297 & -0.4381 & -1.1183 \\ 1.6189 & 1.5416 & -0.2518 & -0.8424 & 0.1845 \end{bmatrix} \tag{2}$$

(*5 Points*)