**Exercise for MA-INF 2201 Computer Vision WS25/26**
**Submission on 26.10.2025**

## Overview

In this assignment, you are required to write code for the programming tasks in Questions 2 to 4. We recommend using Python 3.12, NumPy 2.3.3, and OpenCV 4.11.0.86. You may only use NumPy, OpenCV, matplotlib, and scikit-image (scikit-image for Question 2, Part d only). No other external libraries are permitted. Submit your Python files (.py or Jupyter notebook) with clear comments indicating which task each section addresses. Complete the provided templates for Questions 2 and 3. Provide a separate Python file for Question 4. Also submit a PDF containing your answer to Question 1 and all discussion sections.

1. **Convolution Theorem**
   Convolution is a fundamental operation in LTI (Linear Time-Invariant) systems with essential properties. Provide a formal proof for the following two properties, addressing the specified time domains:

   ### a) Distributive Property (Discrete-Time)

   Show that $f[n] * (g[n] + h[n])$ is equal to $(f[n] * g[n]) + (f[n] * h[n])$.

   ### b) Associative Property (Continuous-Time)

   Show that $(f(t) * g(t)) * h(t)$ is equal to $f(t) * (g(t) * h(t))$.

   ***(2 points)***

2. **Denoising and Optimization**

   Read the images `bonn.jpg` and `bonn_noisy.jpg` and convert them to grayscale images. The first image is the original image and the second one has been corrupted with mixed noise (Gaussian and Salt-and-Pepper noise). Your task is to denoise the noisy image according to the following instructions.

   **Implementation Note:** For each filter in parts (a), (b) and (c), you must provide two solutions:

   1. One using the built-in `cv2` library function
   2. One implemented by you from scratch (e.g., performing convolution for Gaussian, and median calculation for Median)

   Display the resulting image for each applied filter.

   (a) **Gaussian Filter:** Apply the Gaussian filter.
   (b) **Median Filter:** Apply the Median filter.
   (c) **Bilateral Filter:** Apply the Bilateral filter.

**Image Quality Evaluation:**

PSNR is a metric used to measure the quality of reconstructed or denoised images compared to the original image. It quantifies the ratio between the maximum possible signal power and the noise power. Higher PSNR values indicate better image quality (less noise/distortion).

The formula for PSNR is:

$$\text{PSNR} = 10 \cdot \log_{10}\left(\frac{\text{MAX}^2}{\text{MSE}}\right)$$

where:

- MAX is the maximum possible pixel value (255 for 8-bit grayscale images)
- MSE (Mean Squared Error) = $\frac{1}{mn}\sum_i \sum_j [I(i,j) - K(i,j)]^2$
  - $I(i,j)$ is the original image
  - $K(i,j)$ is the denoised/noisy image
  - $m, n$ are the image dimensions

**Calculating PSNR between original and denoised image using skimage:**

```python
from skimage.metrics import peak_signal_noise_ratio as psnr
psnr_value = psnr(original_image, denoised_image, data_range=255)
```

The `data_range` parameter should be set to 255 for 8-bit images (0-255 pixel values).

(d) **Performance Comparison:** Compare which filter performs the best at denoising using the PSNR metric. Calculate PSNR values using the provided skimage library code for all three filters.

(e) **Parameter Optimization:** For all three filters, write a function that iterates over relevant parameters (e.g., kernel size for Median/Gaussian; $\sigma_d$, $\sigma_r$ for Bilateral) to find the optimal parameter values that maximize PSNR. After finding the best parameters through your optimization function, report them clearly in your code as variables or constants, and use these values to generate your final denoised images. During grading, the optimization code may be commented out to verify that the reported parameters produce the claimed PSNR values.

(f) **Discussion (3-5 sentences):** The input image contains mixed Gaussian and Salt-and-Pepper noise. Based on your PSNR results and visual inspection, discuss which filter handled the noise most effectively. Explain why certain filters are theoretically better suited for specific noise types, and whether your experimental results aligned with these expectations.

*(10 points)*

3. **Integral Images**
Read the image bonn.jpg and convert it into a grayscale image.

(a) Implement a function to calculate the integral image.

(b) Implement a function to compute the mean gray value of a specific region using the integral image. The region is defined by its top left corner at (10, 10) and bottom right corner at (60, 80).

(c) Implement a function to calculate the mean gray value of the same region by dividing the sum of all pixel intensities in that region by the total number of pixels in the region.

(d) Analyze the computational complexity of both methods (b) and (c) by providing their theoretical time complexity (Big-O notation) and measuring their execution times in your implementation.

***(4 points)***

4. **Separability of Filters**
Read the image bonn.jpg and convert it into a gray image.

**Kernel 1:**

$$K_1 = \begin{bmatrix} 0.0113 & 0.0838 & 0.0113 \\ 0.0838 & 0.6193 & 0.0838 \\ 0.0113 & 0.0838 & 0.0113 \end{bmatrix}$$

**Kernel 2:**

$$K_2 = \begin{bmatrix} -0.8984 & 0.1472 & 1.1410 \\ -1.9075 & 0.1566 & 2.1359 \\ -0.8659 & 0.0573 & 1.0337 \end{bmatrix}$$

(a) Decompose each of the given kernels using the SVD class of `OpenCV`. Determine which kernel is separable.

(b) For any kernel that is not separable, create an approximation by taking only the highest singular value. Filter the image using both the original 2D kernel and the separable approximation.

(c) Compute the absolute pixel-wise difference between the results obtained using the full kernel versus the separable approximation, and print the maximum pixel error.

***(4 points)***