Exercise 02 for MA-INF 2201 Computer Vision WS25/26
27.10.2025
Submission on 03.11.2025

1. **Parseval's Theorem:** Consider a discrete function $f(x)$ defined on a set of N points (e.g., $x = 0, 1, ..., N-1$). Its Fourier transform $F(\omega)$ is defined using the unitary discrete Fourier transform (DFT),

$$F(\omega) = \frac{1}{\sqrt{N}} \sum_x f(x) e^{-i\omega x},$$
$$f(x) = \frac{1}{\sqrt{N}} \sum_\omega F(\omega) e^{i\omega x}$$

   where the frequencies are discrete (e.g., $\omega = \frac{2\pi k}{N}$ for $k = 0, 1, ..., N-1$). Prove the Parseval's theorem $\sum_x |f(x)|^2 = \sum_\omega |F(\omega)|^2$. Provide an exhaustive explanation for each step of your reasoning.

   (2 Points)

2. **Fourier Transform:** The Fourier transform decomposes an image into magnitude and phase components. The objective of this exercise is to investigate the relative importance of these components by swapping them between two images: `1.png` and `2.png`. You are required to:

   - Visualize the Phase and Magnitude of each image
   - Combine the magnitude from `1.png` with the phase from `2.png`. Plot the resulting image: `reconstructed_mag1_phase2.png`
   - Combine the phase from `1.png` with the magnitude from `2.png`. Plot the resulting image: `reconstructed_mag2_phase1.png`
   - Compute the Mean Absolute Differences between the original images and generated images `reconstructed_mag1_phase2.png` and `reconstructed_mag2_phase1.png`. Compare the results and analyze your findings.

   (3 Points)

3. **Filtering in Spatial and Frequency Domains:** In this exercise, we compare box and Gaussian filters (with same e.g. 9×9 filter) applied both in the spatial and frequency domains:

   - Load the image `lena.png`, implement the box filter and filter the image with your implemented filter. Plot the image. (do not use `cv2.filter2D`)
   - Load the image `lena.png`, implement the Gaussian filter and filter the image with your implemented filter. Plot the image. (do not use `cv2.GaussianBlur`)
   - Apply both filters in the frequency domain (you can use `numpy.fft`). Plot the images.
   - Compare the spatial and frequency results visually and quantitatively by computing the Mean Absolute Difference (MAD) between them. Your implementation should achieve **MAD** $< 1 \times 10^{-7}$ between spatial and frequency results.

(4 Points)

4. **Normalized Cross-Correlation (NCC):** In this exercise, we use Normalized Cross-Correlation to find corresponding points between two stereo images

   - Load the stereo image pair `left.png` and `right.png`
   - Apply NCC to find corresponding points between two images (do not use built-in stereo or template-matching functions such as `cv2.StereoBM_create` and `cv2.matchTemplate`)
   - Generate a benchmark disparity map using the built-in `cv2.StereoBM_create` function. Ensure its `blockSize` and `numDisparities` match your manual implementation.
   - Visualize both your manual NCC map and the built-in benchmark map. Quantitatively compare them by calculating the Mean Absolute Error (MAE) between the two, using the built-in map as the reference. Your manual implementation must achieve an **MAE of less than 0.7**.

   (6 Points)

5. **Canny Edge Detector:** In this exercise, you will implement the Canny Edge Detector:

   - Implement a function that performs the Canny Edge Detection algorithm (do not use built-in `cv2.Canny` function)
   - Load the image `bonn.jpg`, apply your implementation, and display the result.
   - Apply the built-in `cv2.Canny` function to the same image and display the result.
   - Compute the F1-score and the Mean Absolute Difference (MAD) between your result and OpenCV's output.
   - Your implementation should achieve at most **0.07 MAD** and at least **0.6 F1-score**.

   (5 Points)