DIGITAL IMAGE AND VIDEO PROCESSING LAB - EXPERIMENT NO. 5

SPATIAL DOMAIN FILTERING

Submitted by Unnati Singh (21EC39027)

Overview:

This experiment focuses on designing and implementing various spatial filters such as Mean, Median, Prewitt, Laplacian, Sobel kernels (horizontal, vertical, diagonal), Gaussian Blur, and Laplacian of Gaussian. The filters are applied on a stack of grayscale images to explore their effects on image enhancement and noise reduction. Additionally, the experiment introduces a custom filter, `gaussian_unblur`, to reverse the effects of blurring through iterative convolution and correction steps.

Files Included:

1. `Exp5_21EC39027.py`: Python script containing the modular functions for spatial filtering and unblurring.
2. `Exp5_21EC39027.ipynb`: Jupyter Notebook for interactive execution and visualization of the filtering techniques.
3. `input/`: Directory containing the stack of noisy and normal grayscale images (e.g., jetplane.jpg, lake.jpg, livingroom.jpg).
4. `Spatial Domain Filtering.pdf`: PDF containing the problem statement.
5. `README.pdf`: This README file.

Functions Overview:

1. `meanfilter(input_image, kernel_size=5):`
   - **Purpose:** Applies a mean filter to smooth the images by averaging pixel values in a local neighbourhood.
   - **Output:** The filtered image stack with noise reduction and blurring.

2. `medianfilter(input_image, kernel_size=5):`
   - **Purpose:** Applies a median filter to reduce noise while preserving edges.
   - **Output:** The filtered image stack with noise removal while retaining sharper features.

3. `prewitt(input_image):`
   - **Purpose:** Applies the Prewitt filter to detect edges in the specified direction (horizontal, vertical, or diagonal).
   - **Output:** Edge-detected image emphasizing boundaries in the given direction.

4. `sobel(input_image):`
   - **Purpose:** Applies the Sobel filter for edge detection similar to Prewitt but with better noise handling.
   - **Output:** Edge-detected image highlighting horizontal, vertical, or diagonal edges.

5. `laplacian(input_image):`
   - **Purpose:** Applies the Laplacian filter to detect regions of rapid intensity change (edges).
   - **Output:** Image with detected edges in all directions.

6. `gaussian_blur(input_image, kernel_size=5, sigma=1):`
   - **Purpose:** Applies Gaussian blur to smooth the image while reducing noise and detail.
   - **Output:** Blurred image stack with a more natural smoothing effect.

7. `laplacian_of_gaussian(input_path, gaussian_kernel_size = 5, gaussian_sigma = 1.0):`
   - **Purpose:** Combines Gaussian blur and Laplacian filtering to detect edges while smoothing noise.
   - **Output:** Edge-detected images after noise reduction.

8. `gaussian_unblur(image, sigma=1, max_iterations=100, tolerance=1e-3):`
   - **Purpose:** Iteratively unblurs a Gaussian-blurred image using a series of convolution and correction steps until convergence.
   - **Output:** Corrected image that restores original sharpness by reversing the blurring effect.