**DEPARTMENT OF INFORMATION TECHNOLOGY**

**COURSE CODE: DJS22ITL603**

**COURSE NAME: Image Processing and Computer Vision Laboratory CLASS: T Y B. TECH**

**NAME: Anish Sharma ROLL: I011 DIV: IT1-1**

**EXPERIMENT NO. 5 CO/LO:** Apply Image Enhancement Techniques.

**AIM / OBJECTIVE:** To apply Spatial Filtering Enhancement Operations on a given image.

# EXERCISE

Implement spatial domain filtering such as LPF (Average and Median Filter), HPF, High Boost filters and observe its blurring or smoothening or sharpening effects on the image using PIL/OpenCV libraries:

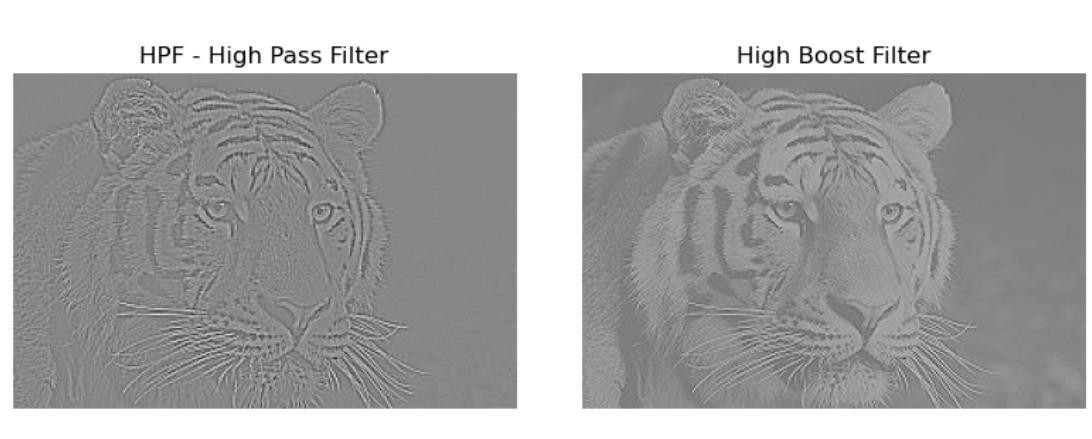
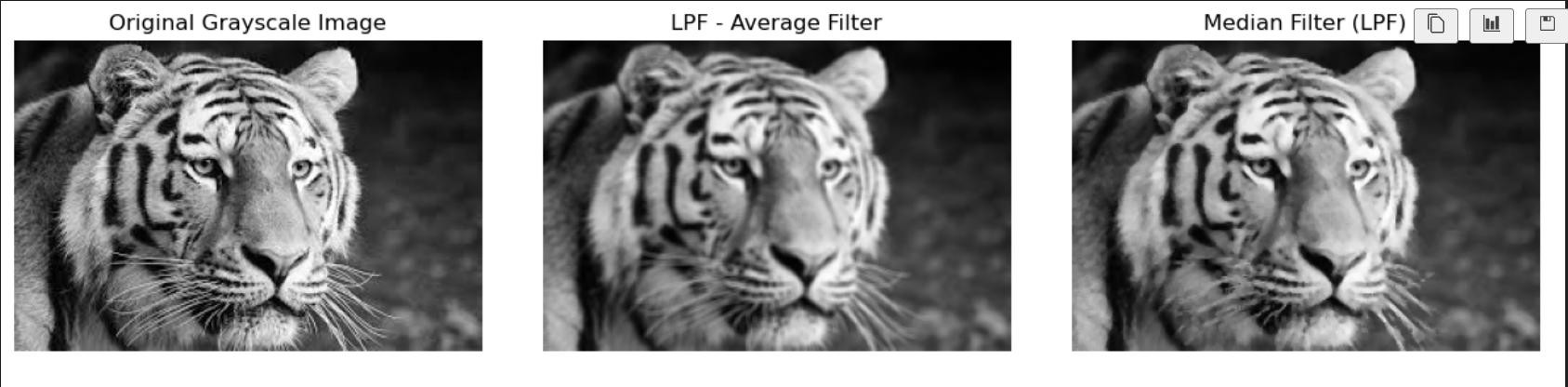
1. Read the image.
2. Convert into grayscale if it is colored.
3. Convert into the double format.
4. Use appropriate masks or filters.
5. Apply convolution operation over the given image.
6. Display the enhanced images.

|  |
| --- |
| **CODE:**  import numpy as np import cv2 from PIL import Image  import matplotlib.pyplot as plt    # Step 1: Read the image image\_path = 'images.jpg' image = Image.open(image\_path)    # Step 2: Convert the image into grayscale if it is colored image\_gray = image.convert('L') # 'L' mode is for grayscale image\_gray = np.array(image\_gray)    # Step 3: Convert the image into double format (float64) image\_gray\_double = np.float64(image\_gray) |

|  |
| --- |
| # Function to apply convolution using a given kernel def apply\_convolution(image, kernel):  return cv2.filter2D(image, -1, kernel)    # 1. Low Pass Filters (LPF) - Average Filter average\_filter = np.ones((3, 3), np.float32) / 9 # 3x3 Average Filter lpf\_image = apply\_convolution(image\_gray\_double, average\_filter)  # 2. Median Filter (LPF) median\_filter\_image = cv2.medianBlur(image\_gray, 3) # 3x3 Median Filter  # 3. High Pass Filter (HPF) - Using a simple kernel hpf\_kernel = np.array([[-1, -1, -1], [-1, 8, -1],  [-1, -1, -1]]) # HPF Kernel (Laplacian) hpf\_image = apply\_convolution(image\_gray\_double, hpf\_kernel)  # 4. High Boost Filter high\_boost\_kernel = np.array([[0, -1, 0],  [-1, 5, -1],  [0, -1, 0]]) # Simple High Boost Filter Kernel high\_boost\_image = apply\_convolution(image\_gray\_double, high\_boost\_kernel)  # Step 5: Display the enhanced images  fig, axes = plt.subplots(2, 3, figsize=(12, 8))    # Original image display axes[0, 0].imshow(image\_gray, cmap='gray') axes[0, 0].set\_title('Original Grayscale Image') axes[0, 0].axis('off')  # LPF - Average Filter axes[0, 1].imshow(lpf\_image, cmap='gray') axes[0, 1].set\_title('LPF - Average Filter') axes[0, 1].axis('off')    # Median Filter (LPF) axes[0, 2].imshow(median\_filter\_image, cmap='gray') axes[0, 2].set\_title('Median Filter (LPF)') axes[0, 2].axis('off')  # HPF axes[1, 0].imshow(hpf\_image, cmap='gray') axes[1, 0].set\_title('HPF - High Pass Filter') axes[1, 0].axis('off')    # High Boost Filter axes[1, 1].imshow(high\_boost\_image, cmap='gray') axes[1, 1].set\_title('High Boost Filter') axes[1, 1].axis('off')    # Show the plots plt.tight\_layout() |

plt.show()

**OUTPUT:**



**QUESTIONS:**

1. Write masks and give the mechanics of spatial filtering?
2. What is salt and pepper noise and how it can be removed effectively?
3. Compare smoothening and sharpening filters.

**CONCLUSION:**

In conclusion, the implementation of spatial domain filtering techniques—such as Low Pass Filters (Average and Median), High Pass Filters (Laplacian), and High Boost Filters— demonstrates the ability to manipulate images for various effects like smoothing, blurring, sharpening, and edge enhancement. The average and median filters reduce noise and smooth images, while the high-pass filter highlights edges and details, and the high-boost filter enhances the sharpness of the image. By using convolution with appropriate kernels, these filters allow for effective image enhancement, which can be useful in applications like image processing, computer vision, and noise reduction.

**REFERENCES:**

**Website References:**

1. Kaggle, “Spatial Filtering OpenCV,” Available: [https://www.kaggle.com/code/bhavinmoriya/spatial-filtering-opencv.](https://www.kaggle.com/code/bhavinmoriya/spatial-filtering-opencv)
2. OpenCV “Image Filtering,” *OpenCV Documentation*. Available: [https://docs.opencv.org/4.x/dd/d6a/tutorial\_js\_filtering.html#:~:text=As%20in%20one%2Ddi mensional%20signals,finding%20edges%20in%20the%20images..](https://docs.opencv.org/4.x/dd/d6a/tutorial_js_filtering.html#%3A~%3Atext%3DAs%20in%20one%2Ddimensional%20signals%2Cfinding%20edges%20in%20the%20images)