EXP NO: 4	IMPLEMENT THE VARIOUS LOW PASS AND HIGH PASS FILTERING MECHANISMS

AIM:

To implement and understand the working of various low pass and high pass filters for image smoothing and edge enhancement using OpenCV.

THEORY:

Filtering is used in image processing to enhance or suppress specific features.

• Low Pass Filters (LPF): Used for image smoothing or blurring by removing high-frequency components.
Examples: Average filter, Gaussian filter, Median filter.

• High Pass Filters (HPF): Used for edge detection and sharpening by preserving high-frequency.
Examples: Laplacian filter, Sobel filter, High-pass kernel.

REQUIREMENTS:

- Python 3.x
- OpenCV
- Matplotlib
- NumPy

ALGORITHM:

Step 1: Start

Step 2: Read the input image using cv2.imread().

Step 3: Convert the image to grayscale using cv2.cvtColor().

Step 4: Apply low pass filters:

a. Average filter using cv2.blur().

b. Gaussian filter using cv2.GaussianBlur().

c. Median filter using cv2.medianBlur().

Step 5: Apply high pass filters:

a. Laplacian filter using cv2.Laplacian().

b. Sobel filter using cv2.Sobel().

Step 6: Display the filtered images using cv2.imshow().

Step 7: Wait for key press and close all windows using cv2.waitKey(0) and cv2.destroyAllWindows().

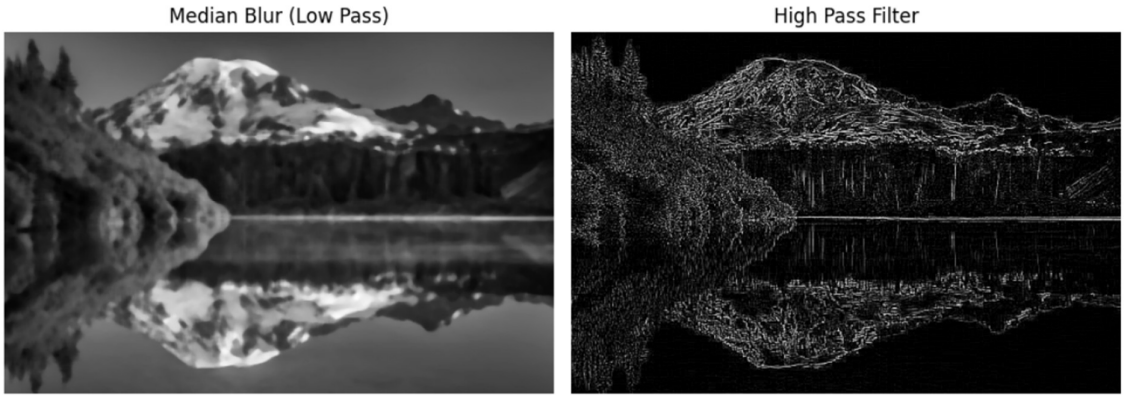
Step 8: End

CODE:

```
# ✅ Step 2: Import required libraries
import cv2
import numpy as np
import matplotlib.pyplot as plt
from google.colab import files


# ✅ Step 3: Upload image
print("📤 Please upload an image file:")
uploaded = files.upload()
image_path = next(iter(uploaded))


# ✅ Step 4: Load image in color and grayscale
img_color = cv2.imdecode(np.frombuffer(uploaded[image_path], np.uint8), cv2.IMREAD_COLOR)
img_color = cv2.cvtColor(img_color, cv2.COLOR_BGR2RGB)  # Convert BGR to RGB for matplotlib
img_gray = cv2.cvtColor(img_color, cv2.COLOR_RGB2GRAY)


# ✅ Step 5: Apply Filters on grayscale image


# Low Pass Filters
blur_avg = cv2.blur(img_gray, (7, 7))
blur_gaussian = cv2.GaussianBlur(img_gray, (7, 7), 0)
blur_median = cv2.medianBlur(img_gray, 7)


# High Pass Filter using kernel
high_pass_kernel = np.array([[-1, -1, -1],
                             [-1,  8, -1],
                             [-1, -1, -1]])
high_pass = cv2.filter2D(img_gray, -1, high_pass_kernel)
```

```
# ✅ Step 6: Plot All Results
plt.figure(figsize=(15, 10))

plt.subplot(2, 3, 1)
plt.title("Original (Color)")
plt.imshow(img_color)
plt.axis('off')

plt.subplot(2, 3, 2)
plt.title("Averaging (Low Pass)")
plt.imshow(blur_avg, cmap='gray')
plt.axis('off')

plt.subplot(2, 3, 3)
plt.title("Gaussian Blur (Low Pass)")
plt.imshow(blur_gaussian, cmap='gray')
plt.axis('off')

plt.subplot(2, 3, 4)
plt.title("Median Blur (Low Pass)")
plt.imshow(blur_median, cmap='gray')
plt.axis('off')

plt.subplot(2, 3, 5)
plt.title("High Pass Filter")
plt.imshow(high_pass, cmap='gray')
plt.axis('off')

plt.tight_layout()
plt.show()
```

OUTPUT:



RESULT:

Low-pass filtering successfully smoothed the image by reducing noise and blurring fine details, while high-pass filtering enhanced edges and fine details, making boundaries and sharp features more prominent in the processed images.