**Title:**

Implement image filtering operations: Gaussian, Median, and Bilateral filters.

**Aim:**

To apply Gaussian, Median, and Bilateral filtering techniques on images to observe noise reduction and smoothing effects using OpenCV.

**Description:**

Image filtering is a technique used to enhance or modify images by suppressing noise or highlighting specific features.

* **Gaussian Filter:**
  + It is a linear filter that uses a Gaussian function for smoothing.
  + It reduces noise and detail in the image.
  + The kernel is based on the Gaussian distribution, giving more weight to the central pixels.
* **Median Filter:**
  + It is a nonlinear filter that replaces each pixel value with the median of its neighborhood.
  + It effectively removes salt-and-pepper noise.
  + Preserves edges better than Gaussian filter.
* **Bilateral Filter:**
  + It is a nonlinear, edge-preserving, and noise-reducing smoothing filter.
  + Considers both spatial proximity and pixel intensity difference.
  + Excellent for smoothing while keeping edges sharp.

**Source Code:**

# Import necessary libraries

import cv2 # OpenCV for image processing

import numpy as np # NumPy for numerical operations

from matplotlib import pyplot as plt # For displaying images

# ---------------------- Load the Image ----------------------

image = cv2.imread('download.jpeg') # Load image from the file system

if image is None: # Check if image was loaded successfully

print("Image not found.") # If not, display error message

exit() # Exit the program

# ---------------------- Apply Gaussian Filter ----------------------

# Gaussian Blur: smooths the image by averaging pixels with a Gaussian kernel

# Syntax: cv2.GaussianBlur(src, ksize, sigmaX)

gaussian = cv2.GaussianBlur(image, (5, 5), 0) # 5x5 kernel, sigma=0 (auto)

# ---------------------- Apply Median Filter ----------------------

# Median Blur: replaces each pixel with the median of neighboring pixels

# Good for removing salt-and-pepper noise

# Syntax: cv2.medianBlur(src, ksize)

median = cv2.medianBlur(image, 5) # Kernel size must be odd and >1

# ---------------------- Apply Bilateral Filter ----------------------

# Bilateral Filter: smooths image but preserves edges

# Syntax: cv2.bilateralFilter(src, d, sigmaColor, sigmaSpace)

# d = Diameter of each pixel neighborhood

# sigmaColor = Filter sigma in color space

# sigmaSpace = Filter sigma in coordinate space

bilateral = cv2.bilateralFilter(image, 9, 75, 75) # Edge-preserving smoothing

# ---------------------- Display Results ----------------------

# List of titles to display for each image

titles = ['Original', 'Gaussian Filter', 'Median Filter', 'Bilateral Filter']

# List of corresponding image results

images = [image, gaussian, median, bilateral]

plt.figure(figsize=(10, 8)) # Set figure size for better visibility

# Loop through and plot each image in a 2x2 grid

for i in range(4):

plt.subplot(2, 2, i+1) # Define subplot position (2 rows, 2 columns)

plt.imshow(cv2.cvtColor(images[i], cv2.COLOR\_BGR2RGB)) # Convert BGR to RGB for correct display

plt.title(titles[i]) # Set subplot title

plt.axis('off') # Turn off axis ticks and labels

plt.tight\_layout() # Adjust layout to prevent overlapping

plt.show() # Display the image grid

**Output:**

The following outputs were generated after executing the program:

1. **Original Image** – Displays the unprocessed input color image.
2. **Gaussian Filtered Image** – Shows the image after applying a 5×5 Gaussian blur. Noticeable smoothness with minor edge blurring.
3. **Median Filtered Image** – The image appears clearer with salt-and-pepper noise removed. Edge sharpness is better preserved than with Gaussian.
4. **Bilateral Filtered Image** – Noise is removed while retaining crisp edges. Best balance between smoothing and detail retention.

These four images were displayed in a 2×2 grid using Matplotlib for visual comparison.

**Result:**

The Python program executed successfully using OpenCV and Matplotlib. Gaussian, Median, and Bilateral filters were applied to the original image to demonstrate the difference in smoothing capabilities and edge preservation. The bilateral filter produced the best results in terms of both noise reduction and edge sharpness.