**Lab 02: Image Filtering**

**Lab Title:** Image Filtering in Python

**Lab Outcomes**

After completing this lab, students will be able to:

1. Understand the fundamental concepts of image filtering.
2. Implement various spatial filters using OpenCV and NumPy.
3. Apply low-pass (smoothing) and high-pass (edge detection) filters.
4. Analyze the impact of different filters on images.
5. Develop custom filters for specific applications.

**Introduction**

Image filtering is a fundamental technique in image processing used to enhance or modify images. Filters can be classified into:

* **Low-pass filters (Smoothing):** Reduce noise and blur images.
* **High-pass filters (Edge detection):** Enhance edges and details.
* **Custom filters:** Designed for specific image processing tasks.

Filters operate using **convolution**, where a kernel (small matrix) slides over an image to compute a weighted sum of pixel values. Some commonly used filters include:

* **Gaussian Blur:** Removes noise by averaging pixel values with a Gaussian function.
* **Median Filter:** Replaces each pixel with the median of neighboring pixels.
* **Sobel & Laplacian Filters:** Detect edges in images.

Python provides libraries like **OpenCV** and **NumPy** to implement these techniques efficiently.

**Activities**

**Activity 1: Reading and Displaying an Image**

**Instructions:**

1. Import necessary libraries.
2. Read an image using OpenCV.
3. Display the original image using OpenCV.

import cv2

import matplotlib.pyplot as plt

# Load an image

image = cv2.imread('image.jpg')

image = cv2.cvtColor(image, cv2.COLOR\_BGR2RGB) # Convert to RGB for correct display

# Display the image

plt.imshow(image)

plt.axis('off')

plt.show()

**Activity 2:**

**Instructions:**

1. Read an image.
2. Apply a Gaussian Blur using OpenCV.
3. Display the filtered image.

blurred = cv2.GaussianBlur(image, (5,5), 0)

plt.imshow(blurred)

plt.axis('off')

plt.show()

**Activity 3: Applying Median Filter**

**Instructions:**

1. Read an image.
2. Apply a Median Filter.
3. Display the result.

median\_filtered = cv2.medianBlur(image, 5)

plt.imshow(median\_filtered)

plt.axis('off')

plt.show()

**Activity 4: Applying Edge Detection using Sobel Filter**

**Instructions:**

1. Convert an image to grayscale.
2. Apply the Sobel filter.
3. Display the result.

gray = cv2.cvtColor(image, cv2.COLOR\_RGB2GRAY)

sobelx = cv2.Sobel(gray, cv2.CV\_64F, 1, 0, ksize=5)

plt.imshow(sobelx, cmap='gray')

plt.axis('off')

plt.show()

**c,**

**Instructions:**

1. Convert an image to grayscale.
2. Apply the Laplacian filter.
3. Display the result.

laplacian = cv2.Laplacian(gray, cv2.CV\_64F)

plt.imshow(laplacian, cmap='gray')

plt.axis('off')

plt.show()

**Activity 6: Applying a 3x3 Filter Using a Custom Kernel**

**Instructions:**

1. Define a 3x3 kernel matrix.
2. Apply the kernel to an image using OpenCV.
3. Display the filtered image.

import numpy as np

# Define a 3x3 sharpening kernel

kernel = np.array([[0, -1, 0],

[-1, 5, -1],

[0, -1, 0]])

# Apply the filter

filtered\_image = cv2.filter2D(image, -1, kernel)

plt.imshow(filtered\_image)

plt.axis('off')

plt.show()

**Activity 7: Convolution on a 2D Grayscale Image**

**Instructions:**

1. Convert an image to grayscale.
2. Define a custom kernel.
3. Perform convolution using OpenCV.
4. Display the filtered image.

# Convert image to grayscale

gray\_image = cv2.cvtColor(image, cv2.COLOR\_RGB2GRAY)

# Define a 3x3 edge detection kernel

kernel = np.array([[-1, -1, -1],

[-1, 8, -1],

[-1, -1, -1]])

# Apply convolution

filtered\_gray = cv2.filter2D(gray\_image, -1, kernel)

plt.imshow(filtered\_gray, cmap='gray')

plt.axis('off')

plt.show()

**Home Assignment**

**Task 2: Custom Kernel for Edge Detection**

**Scenario: Define a custom kernel for edge detection and apply it to an image.**

**Task 3: Adaptive Smoothing Filter**

**Scenario: Implement an adaptive smoothing filter where the kernel size depends on image noise levels.**

**Task 4: Combining Filters**

**Scenario: Apply a sequence of filters (Gaussian Blur + Sobel) and analyze the results.**

**Task 5: Noise Removal Using Bilateral Filtering**

**Scenario: Given a noisy image, apply a bilateral filter to preserve edges while reducing noise.**

**Task 6: Implementing High-Pass Filtering**

**Scenario: Design a high-pass filter using a custom kernel and apply it to an image.**

**Task 7: Image Gradient Magnitude Calculation**

**Scenario: Compute the gradient magnitude using both Sobel X and Sobel Y filters.**

**Task 8: Object Detection Using Edge Detection**

**Scenario: Apply edge detection to identify objects in an image.**

**Task 9: Design a Filter for Enhancing Text in an Image**

**Scenario: Given an image with text, apply appropriate filtering techniques to improve text visibility.**

**Task 10: Real-Time Filtering on Webcam Feed**

**Scenario: Capture live video and apply a selected filter in real time.**