EXPT 2	Low pass Filter & High pass Filter (Neighbourhood Operations)

AIM:-

To implement spatial filtering techniques using mean, median, and Gaussian filters for noise reduction (low-pass filtering) and Sobel filter for edge detection (high-pass filtering) through neighborhood operations.

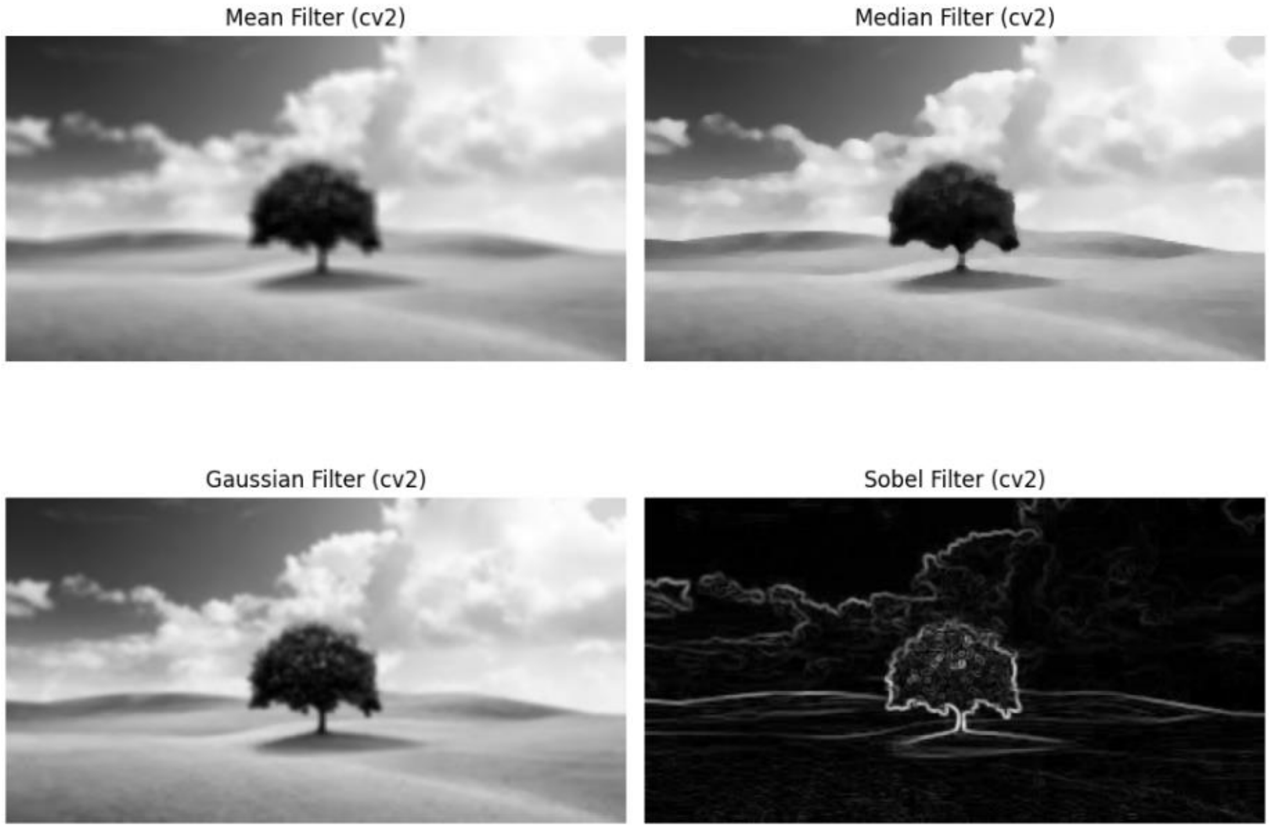
CODE:-

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
from PIL import Image, ImageFilter

def show_image(img, title="Image", cmap_type="gray"):
    plt.imshow(img, cmap=cmap_type)
    plt.title(title)
    plt.axis("off")
    # plt.show() # Do not show here as we want to display in a grid later

# A. MEAN FILTER
# WITH USING CV2 PACKAGE
img = cv2.imread("input.jpg", 0)
mean_filtered_cv2 = cv2.blur(img, (5,5))   # 5x5 kernel

# WITHOUT USING CV2 PACKAGE
img_pil = Image.open("input.jpg").convert("L")
img_array = np.array(img_pil, dtype=np.float32)

kernel = np.ones((5,5), np.float32) / 25
mean_filtered_manual = cv2.filter2D(src=img_array, ddepth=-1, kernel=kernel)  #
or implement manually
```

```
# B. MEDAIN FILTER
# WITH USING CV2 PACKAGE
img = cv2.imread("input.jpg", 0)
median_filtered_cv2 = cv2.medianBlur(img, 5)

# WITHOUT USING CV2 PACKAGE
img_pil = Image.open("input.jpg").convert("L")
median_filtered_manual = img_pil.filter(ImageFilter.MedianFilter(size=5))

# C. GAUSSIAN FILTER
# WITH USING CV2 PACKAGE
img = cv2.imread("input.jpg", 0)
gaussian_filtered_cv2 = cv2.GaussianBlur(img, (5,5), 1.0)

#WITHOUT USING CV2 PACKAGE
img_pil = Image.open("input.jpg").convert("L")
gaussian_filtered_manual = img_pil.filter(ImageFilter.GaussianBlur(radius=2))

# D. SOBEL FILTER
# USING WITH THE CV2 PACKAGE
img = cv2.imread("input.jpg", 0)
sobelx_cv2 = cv2.Sobel(img, cv2.CV_64F, 1, 0, ksize=3)
sobely_cv2 = cv2.Sobel(img, cv2.CV_64F, 0, 1, ksize=3)
sobel_cv2 = np.sqrt(sobelx_cv2**2 + sobely_cv2**2)
sobel_cv2 = np.uint8(sobel_cv2 / np.max(sobel_cv2) * 255)

# WITHOUT USING CV2 PACKAGE
img_pil = Image.open("input.jpg").convert("L")
img_array = np.array(img_pil, dtype=np.float32)

# Sobel kernels
sobel_x_kernel = np.array([[-1,0,1],[-2,0,2],[-1,0,1]])
sobel_y_kernel = np.array([[-1,-2,-1],[0,0,0],[1,2,1]])
```

```
sobelx_manual = cv2.filter2D(img_array, -1, sobel_x_kernel)
sobely_manual = cv2.filter2D(img_array, -1, sobel_y_kernel)

sobel_manual = np.sqrt(sobelx_manual**2 + sobely_manual**2)
sobel_manual = np.uint8(sobel_manual / np.max(sobel_manual) * 255)


fig, axes = plt.subplots(2, 2, figsize=(10, 8))

# Display images in the subplots
axes[0, 0].imshow(mean_filtered_cv2, cmap='gray')
axes[0, 0].set_title("Mean Filter (cv2)")
axes[0, 0].axis('off')

axes[0, 1].imshow(median_filtered_cv2, cmap='gray')
axes[0, 1].set_title("Median Filter (cv2)")
axes[0, 1].axis('off')

axes[1, 0].imshow(gaussian_filtered_cv2, cmap='gray')
axes[1, 0].set_title("Gaussian Filter (cv2)")
axes[1, 0].axis('off')

axes[1, 1].imshow(sobel_cv2, cmap='gray')
axes[1, 1].set_title("Sobel Filter (cv2)")
axes[1, 1].axis('off')

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

OUTPUT:-



RESULT:-

Successfully applied spatial filters demonstrating noise reduction capabilities of low-pass filters and edge enhancement using high-pass Sobel filter. The filters effectively processed images by considering local neighborhood pixels for various image processing tasks.