EXP NO: 5	USE OF FOURIER TRANSFORM FOR FILTERING THE IMAGE

AIM:

To implement Fourier, transform for analysing the frequency domain of an image and applying filtering operations for enhancement and noise reduction.

THEORY:

Fourier Transform is a mathematical tool that converts a spatial domain image into its frequency domain representation. In the frequency domain, low frequencies represent smooth regions, and high frequencies represent edges and noise.

• Forward Fourier Transform: Converts the image to frequency domain using FFT.
• Filtering: By applying masks, low-pass filters smooth the image, while high-pass filters enhance.
• Inverse Fourier Transform: Converts the filtered image back to spatial domain.

REQUIREMENTS:

- Python 3.x
- OpenCV
- Matplotlib
- NumPy

ALGORITHM:

Step 1: Start

Step 2: Read the input image using cv2.imread() and convert it to grayscale.

Step 3: Apply Fourier Transform using numpy.fft.fft2().

Step 4: Shift the zero-frequency component to the center using numpy.fft.fftshift().

Step 5: Create a mask for filtering:

a. Low-pass filter: keep low frequencies, block high frequencies.

b. High-pass filter: block low frequencies, keep high frequencies.

Step 6: Apply the mask to the Fourier transformed image.

Step 7: Perform inverse shift and inverse Fourier transform to reconstruct the filtered image.

Step 8: Display the original, spectrum, and filtered images using matplotlib/cv2.imshow().

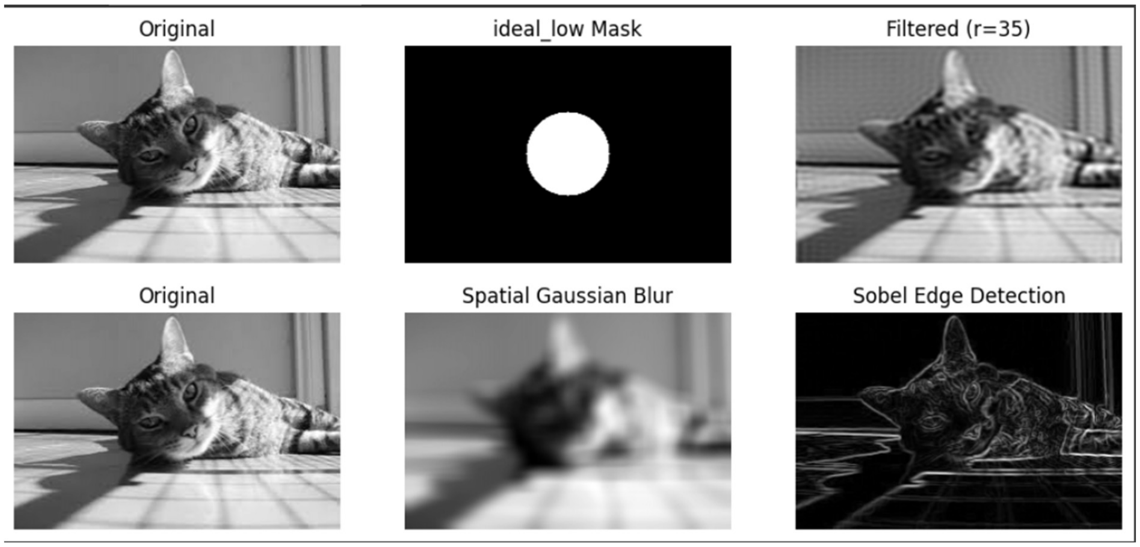
Step 9: End

CODE:

```
import cv2
import numpy as np
import matplotlib.pyplot as plt

# Load image in grayscale
img = cv2.imread('/content/grayscale image.jpg', 0)

# Perform FFT
f = np.fft.fft2(img)
fshift = np.fft.fftshift(f)   # Shift zero frequency component to the center
magnitude_spectrum = 20 * np.log(np.abs(fshift))

# ---------------- Low Pass Filter ---------------- #
rows, cols = img.shape
crow, ccol = rows//2 , cols//2   # center

# Create a mask with high value (1) at low frequencies
mask = np.zeros((rows, cols), np.uint8)
r = 50  # radius for low-pass filter
cv2.circle(mask, (ccol, crow), r, 1, thickness=-1)

# Apply mask
fshift_low = fshift * mask
f_ishift_low = np.fft.ifftshift(fshift_low)
img_low = np.fft.ifft2(f_ishift_low)
img_low = np.abs(img_low)

# ---------------- High Pass Filter ---------------- #
mask_high = np.ones((rows, cols), np.uint8)
cv2.circle(mask_high, (ccol, crow), r, 0, thickness=-1)
```

```
fshift_high = fshift * mask_high
f_ishift_high = np.fft.ifftshift(fshift_high)
img_high = np.fft.ifft2(f_ishift_high)
img_high = np.abs(img_high)

# ---------------- Show Results ---------------- #
plt.figure(figsize=(12, 8))

plt.subplot(231), plt.imshow(img, cmap='gray')
plt.title('Original Image'), plt.axis('off')

plt.subplot(232), plt.imshow(magnitude_spectrum, cmap='gray')
plt.title('FFT Magnitude Spectrum'), plt.axis('off')

plt.subplot(233), plt.imshow(mask, cmap='gray')
plt.title('Low-pass Mask'), plt.axis('off')

plt.subplot(234), plt.imshow(img_low, cmap='gray')
plt.title('Low-pass Result (Blurred)'), plt.axis('off')

plt.subplot(235), plt.imshow(mask_high, cmap='gray')
plt.title('High-pass Mask'), plt.axis('off')

plt.subplot(236), plt.imshow(img_high, cmap='gray')
plt.title('High-pass Result (Edges)'), plt.axis('off')

plt.show()
```

OUTPUT:



RESULT:

The Fourier Transform was applied to the image, successfully converting it to the frequency domain. Frequency-based filtering, such as low-pass and high-pass filters, was performed, and the inverse transform produced images with smoothed regions or enhanced edges, demonstrating effective frequency-domain image processing.