

EXPT 5	Fourier Transforms
---------------	---------------------------

AIM:-

To perform frequency domain image processing using Fourier transforms for smoothing images through low-pass filtering and sharpening images through high-pass filtering in the frequency domain.

CODE:-

```
import numpy as np
import cv2
import matplotlib.pyplot as plt
from google.colab.patches import cv2_imshow

# Load grayscale image - upload 'input.jpg' first
img = cv2.imread('input.jpg', cv2.IMREAD_GRAYSCALE)

if img is None:
    print("Upload 'input.jpg' to Colab.")
else:
    # --- Smoothing Image (Low-pass filter) ---

    # Fourier transform
    f_smooth = np.fft.fft2(img)
    fshift_smooth = np.fft.fftshift(f_smooth)

    # Create a low-pass filter mask
    rows, cols = img.shape
    crow, ccol = rows//2, cols//2
    mask_smooth = np.zeros((rows, cols), np.uint8)
    r_smooth = 30 # Radius of low-pass circle
    cv2.circle(mask_smooth, (ccol, crow), r_smooth, 1, thickness=-1)

    # Apply mask and inverse DFT
```

```

fshift_filtered_smooth = fshift_smooth * mask_smooth
f_ishift_smooth = np.fft.ifftshift(fshift_filtered_smooth)
img_back_smooth = np.fft.ifft2(f_ishift_smooth)
img_back_smooth = np.abs(img_back_smooth)
img_back_smooth = np.uint8(np.clip(img_back_smooth, 0, 255))

# --- Sharpening Image (High-pass filter) ---

# Fourier Transform
f_sharpen = np.fft.fft2(img)
fshift_sharpen = np.fft.fftshift(f_sharpen)

# Create high-pass filter mask
mask_sharpen = np.ones((rows, cols), np.uint8)
r_sharpen = 30 # Radius of low-pass circle to remove from mask
cv2.circle(mask_sharpen, (ccol, crow), r_sharpen, 0, thickness=-1)

# Apply mask and inverse Fourier transform
fshift_filtered_sharpen = fshift_sharpen * mask_sharpen
f_ishift_sharpen = np.fft.ifftshift(fshift_filtered_sharpen)
img_back_sharpen = np.fft.ifft2(f_ishift_sharpen)
img_back_sharpen = np.abs(img_back_sharpen)
img_back_sharpen = np.uint8(np.clip(img_back_sharpen, 0, 255))

# --- Displaying Images ---

plt.figure(figsize=(15, 5))

plt.subplot(1, 3, 1)
plt.title('Original Image')
plt.imshow(img, cmap='gray')
plt.axis('off')

plt.subplot(1, 3, 2)

```

```
plt.title('Smoothed Image (OpenCV)')  
plt.imshow(img_back_smooth, cmap='gray')  
plt.axis('off')
```

```
plt.subplot(1, 3, 3)  
plt.title('Sharpened Image (OpenCV)')  
plt.imshow(img_back_sharpen, cmap='gray')  
plt.axis('off')
```

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

OUTPUT:-



RESULT:-

Successfully transformed images to frequency domain, applied filters, and reconstructed images demonstrating the effectiveness of frequency domain processing. Smoothing and sharpening operations were achieved by manipulating frequency components appropriately.