课堂实验三：图像增强—频域滤波

一、实验目的

1．掌握怎样利用傅立叶变换进行频域滤波

2．掌握频域滤波的概念及方法

3．熟练掌握频域空间的各类滤波器

4．利用 MATLAB 程序进行频域滤波

二、实验内容及步骤

1．调入并显示所需的图片

2．利用 MATLAB 提供的低通滤波器实现图像信号的滤波运算，并与 空间滤波进行比较。

3．利用 MATLAB 提供的高通滤波器对图像进行处理。

4．记录和整理实验报告。

三、实验代码及结果

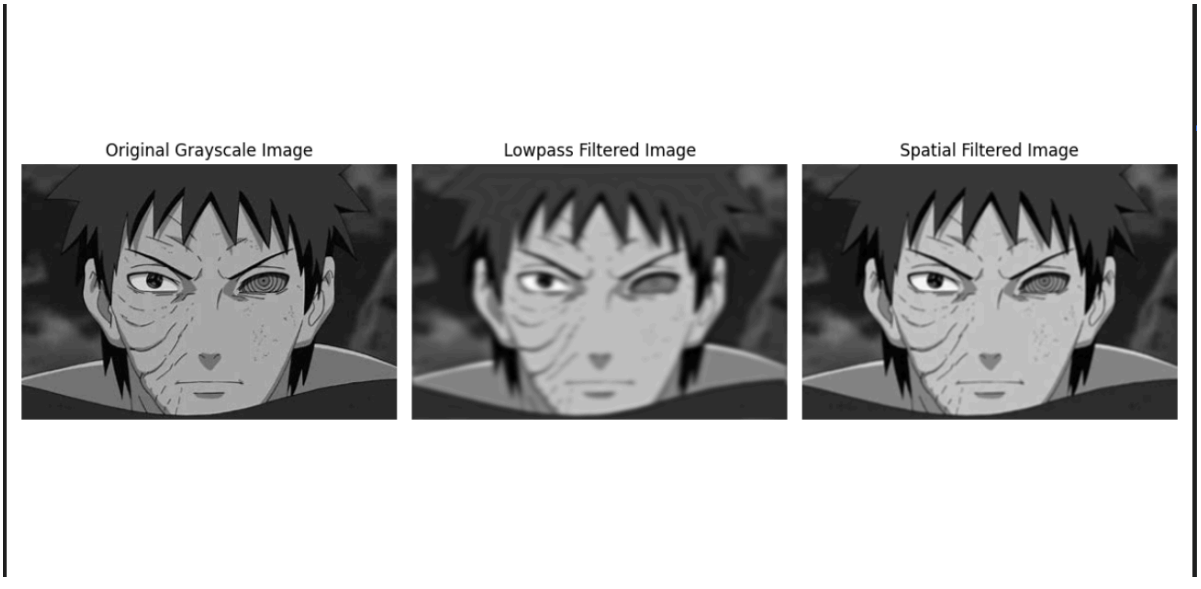
```
import cv2
import numpy as np
import matplotlib.pyplot as plt

# 1. 调入并显示所需的图片
image_path = 'daitu.jpg'  # 图像路径，根据实际情况修改
original_image = cv2.imread(image_path)
if original_image is None:
    print(f"无法读取图片，请检查路径是否正确: {image_path}")
    exit(1)
original_image_rgb = cv2.cvtColor(original_image, cv2.COLOR_BGR2RGB)
plt.figure(figsize=(8, 6))
plt.title('Original Image')
plt.axis('off')
plt.imshow(original_image_rgb)
plt.show()

# 将图像转换为灰度图
gray_image = cv2.cvtColor(original_image, cv2.COLOR_BGR2GRAY)

# 2. 利用低通滤波器实现图像信号的滤波运算
# 创建低通滤波器（巴特沃斯低通滤波器）
def butterworth_lowpass_filter(img, D0, n):
    f = np.fft.fft2(img)
    fshift = np.fft.fftshift(f)
    rows, cols = img.shape
    crow, ccol = rows // 2, cols // 2
    mask = np.ones((rows, cols), np.float32)
    for i in range(rows):
        for j in range(cols):
            d = np.sqrt((i - crow)**2 + (j - ccol)**2)
            mask[i, j] = 1 / (1 + (d / D0)**(2 * n))
```

```
    ishift = fshift * mask
    img_back = np.fft.ifftshift(ishift)
    img_back = np.fft.ifft2(img_back)
    img_back = np.abs(img_back)
    return img_back.astype(np.uint8)

# 应用低通滤波器
D0 = 30  # 截止频率
n = 2    # 滤波器阶数
lowpass_filtered_image = butterworth_lowpass_filter(gray_image, D0, n)

# 使用空间滤波进行比较（使用平均滤波器）
average_filter_3x3 = np.ones((3, 3), np.float32) / 9
spatial_filtered_image = cv2.filter2D(gray_image, -1, average_filter_3x3)

# 显示结果
plt.figure(figsize=(12, 6))
plt.subplot(1, 3, 1)
plt.title('Original Grayscale Image')
plt.axis('off')
plt.imshow(gray_image, cmap='gray')
plt.subplot(1, 3, 2)
plt.title('Lowpass Filtered Image')
plt.axis('off')
plt.imshow(lowpass_filtered_image, cmap='gray')
plt.subplot(1, 3, 3)
plt.title('Spatial Filtered Image')
plt.axis('off')
plt.imshow(spatial_filtered_image, cmap='gray')
plt.tight_layout()
plt.show()

# 3. 利用高通滤波器对图像进行处理
# 创建高通滤波器（巴特沃斯高通滤波器）
def butterworth_highpass_filter(img, D0, n):
    f = np.fft.fft2(img)
    fshift = np.fft.fftshift(f)
    rows, cols = img.shape
    crow, ccol = rows // 2, cols // 2
    mask = np.zeros((rows, cols), np.float32)
    for i in range(rows):
        for j in range(cols):
            d = np.sqrt((i - crow)**2 + (j - ccol)**2)
            mask[i, j] = 1 / (1 + (D0 / d)**(2 * n))
    ishift = fshift * mask
    img_back = np.fft.ifftshift(ishift)
    img_back = np.fft.ifft2(img_back)
    img_back = np.abs(img_back)
    return img_back.astype(np.uint8)

# 应用高通滤波器
highpass_filtered_image = butterworth_highpass_filter(gray_image, D0, n)

# 显示结果
plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
```

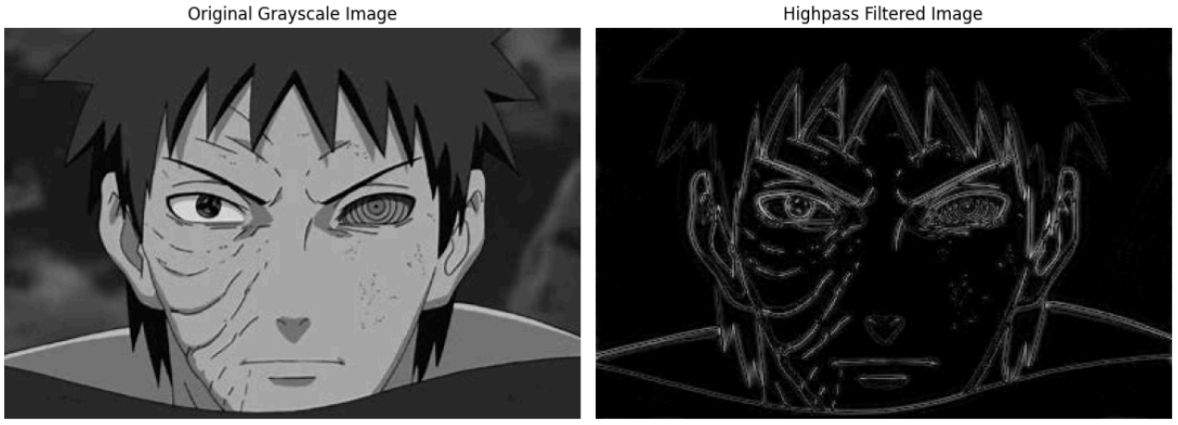
```
plt.title('Original Grayscale Image')
plt.axis('off')
plt.imshow(gray_image, cmap='gray')
plt.subplot(1, 2, 2)
plt.title('Highpass Filtered Image')
plt.axis('off')
plt.imshow(highpass_filtered_image, cmap='gray')
plt.tight_layout()
plt.show()
```

导入所需图片

低通滤波器实现图像信号的滤波运算，并与空间滤波进行比较



高通滤波器对图像进行处理



四、思考题

频域滤波的优点

频域滤波是信号处理中一种强大的工具，它通过将图像转换到频域进行处理，相较于空间滤波具有以下显著优点：

1. **计算效率高**：对于大尺寸图像，频域滤波（尤其是使用 FFT）通常比空间滤波（直接卷积）计算速度更快。例如，一个 N×N 的图像，直接卷积的计算复杂度为 O (N⁴)，而使用 FFT 的频域滤波复杂度为 O (N²logN)。
2. **易于实现特定滤波**：在频域中，可以直观地设计滤波器传递函数（如理想低通、高斯低通等），直接对特定频率成分进行操作。例如，高通滤波可以有效增强图像边缘，而这在空间域中可能需要复杂的卷积核设计。
3. **物理意义明确**：频域滤波能够清晰地分离不同频率成分，便于分析图像的频谱特性。例如，低通滤波可以去除图像中的高频噪声，保留低频细节。
4. **灵活性高**：可以同时处理多个方向的频率成分，适用于各向异性滤波。例如，在处理具有特定方向纹理的图像时，可以设计方向选择性滤波器。

频域滤波注意事项

在进行频域滤波时，需要注意以下关键问题：

1. **频谱位移**：使用 FFT 变换后，频谱的零频率分量位于四个角落，需要使用 fftshift 函数将其移至中心，便于滤波器设计和应用。
2. **振铃效应**：理想低通滤波器在频域截断陡峭，会在空间域产生振铃效应（图像边缘出现波纹状伪影）。可使用高斯低通等平滑过渡的滤波器减轻这一问题。
3. **滤波器尺寸匹配**：频域滤波器的尺寸必须与图像的 FFT 结果尺寸完全一致，否则会导致错误的滤波结果。
4. **计算精度**：由于频域滤波涉及复数运算，需要注意计算过程中的精度问题，特别是在处理大尺寸图像时。
5. **反变换后处理**：经过频域滤波并进行反变换后，可能需要对结果进行裁剪或舍入操作，以确保像素值在有效范围内（如 0-255）。