22336216_陶宇卓

实验二

要求

编写二维傅里叶变换，并且按照书上步骤进行图像处理。

原理

本实验的主要目的是通过实现和应用快速傅里叶变换（FFT）和高斯低通滤波器来处理图像。具体步骤如下：

1. **傅里叶变换**：
   - 傅里叶变换是一种将空间域信号转换为频域信号的数学工具。通过傅里叶变换，可以将图像从空间域转换到频域，从而分析图像的频率成分。
   - 快速傅里叶变换（FFT）是一种高效计算离散傅里叶变换（DFT）的方法，能够显著减少计算复杂度。
2. **高斯低通滤波器**：
   - 高斯低通滤波器是一种频域滤波器，用于去除图像中的高频噪声。其滤波器函数是一个高斯函数，能够平滑图像，保留低频成分。
   - 通过设置截止频率，可以控制滤波器的带宽，从而调整滤波效果。
3. **图像处理步骤**：
   - **图像预处理**：将输入图像转换为灰度图像，并进行零填充以适应傅里叶变换的要求。
   - **频域变换**：对预处理后的图像进行二维快速傅里叶变换，得到图像的频谱。
   - **滤波处理**：生成高斯低通滤波器，并在频域中应用该滤波器，对图像频谱进行滤波。
   - **逆变换**：对滤波后的频谱进行二维逆快速傅里叶变换，恢复到空间域图像。
   - **后处理**：对恢复后的图像进行乘以 (–1)^(x+y) 的操作，并提取最终的处理结果。

通过上述步骤，可以有效地去除图像中的高频噪声，得到平滑的图像效果。

具体实现

Python

```python
import cmath
# 一维快速傅里叶变换（递归实现）
def fft_1d(signal):
    N = len(signal)
    if N <= 1:
        return signal
    even = fft_1d(signal[0::2])
    odd = fft_1d(signal[1::2])
    T = [cmath.exp(-2j * cmath.pi * k / N) * odd[k] for k in range(N // 2)]
    return [even[k] + T[k] for k in range(N // 2)] + [even[k] - T[k] for k in range(N // 2)]

# 一维逆傅里叶变换
def ifft_1d(signal):
    N = len(signal)
    conjugated = [x.conjugate() for x in signal]
    transformed = fft_1d(conjugated)
    return [x.conjugate() / N for x in transformed]

# 二维快速傅里叶变换
def fft_2d(matrix):
    M, N = len(matrix), len(matrix[0])
    # 对每一行进行傅里叶变换
    fft_rows = [fft_1d(row) for row in matrix]
    # 转置后对每一列进行傅里叶变换
    fft_columns = [fft_1d(col) for col in zip(*fft_rows)]
    # 再次转置回原格式
    return [list(row) for row in zip(*fft_columns)]

# 二维逆傅里叶变换
def ifft_2d(matrix):
    M, N = len(matrix), len(matrix[0])
    # 对每一行进行逆傅里叶变换
    ifft_rows = [ifft_1d(row) for row in matrix]
    # 转置后对每一列进行逆傅里叶变换
    ifft_columns = [ifft_1d(col) for col in zip(*ifft_rows)]
    # 再次转置回原格式
    return [list(row) for row in zip(*ifft_columns)]
```

Python

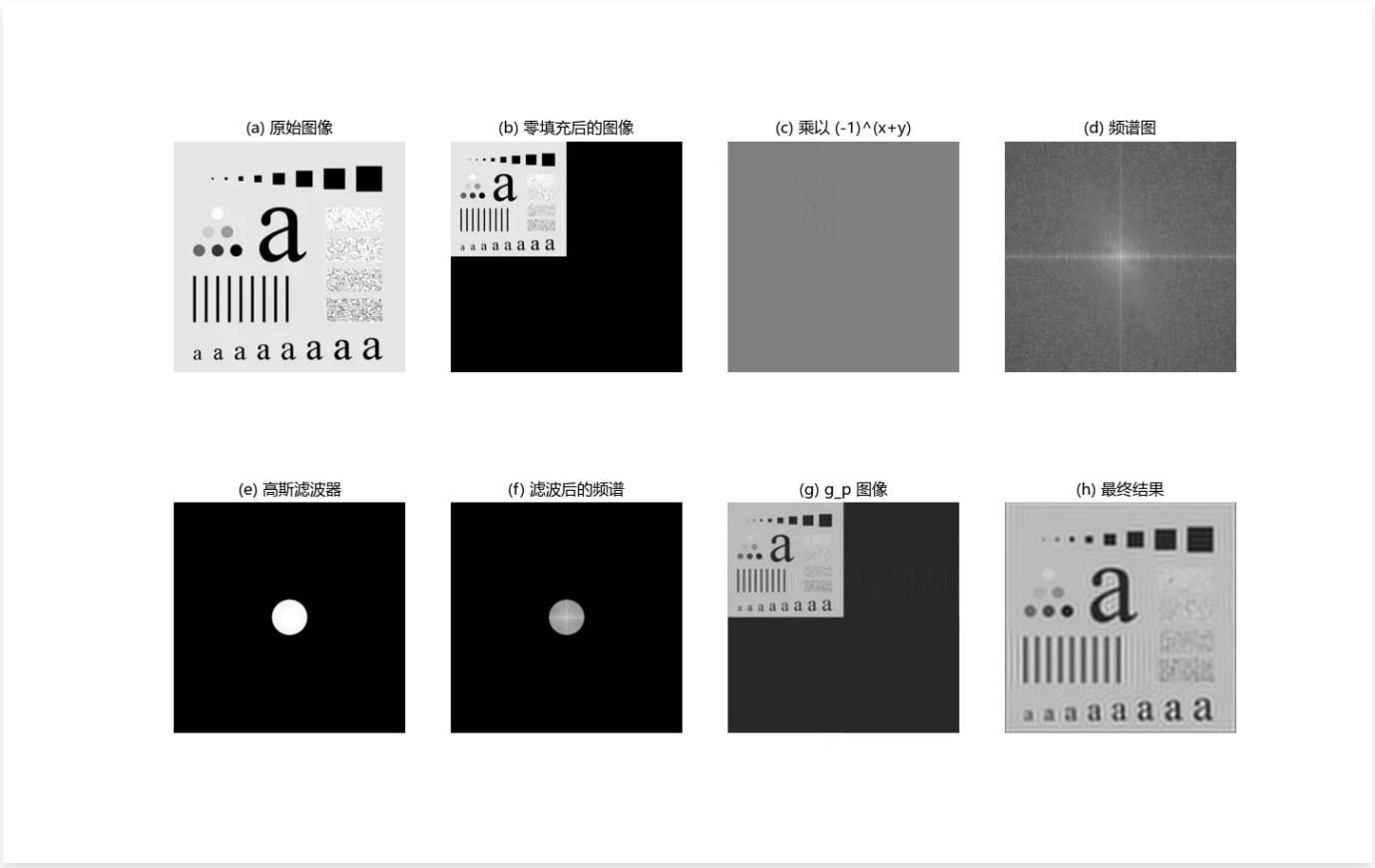
```python
import math
import FFT
import matplotlib.pyplot as plt
font = {'family': 'MicroSoft YaHei',
        'weight': 'bold',
        'size': 'larger'}
(plt.rc("font", family='MicroSoft YaHei', weight="bold"))


# 生成高斯低通滤波器
def gaussian_low_pass_filter(shape, cutoff_ratio=0.95):
    M, N = shape
    center_x, center_y = M // 2, N // 2
    sigma = min(M, N) / 4  # 高斯滤波器的标准差
    filter_matrix = [[0] * N for _ in range(M)]
    total_energy, current_energy = 0, 0

    for u in range(M):
        for v in range(N):
            distance = ((u - center_x) ** 2 + (v - center_y) ** 2) / (2 * sigma ** 2)
            filter_matrix[u][v] = math.exp(-distance)
            total_energy += filter_matrix[u][v]

    # 归一化滤波器以确保总能量比例达到cutoff_ratio
    sorted_values = sorted(filter_matrix[u][v] for u in range(M) for v in range(N))
    threshold_energy = cutoff_ratio * total_energy

    for value in sorted_values:
        current_energy += value
        if current_energy >= threshold_energy:
            threshold = value
            break

    for u in range(M):
        for v in range(N):
            if filter_matrix[u][v] < threshold:
                filter_matrix[u][v] = 0

    return filter_matrix

# 在频域中应用二维滤波器
def apply_frequency_filter(image, filter_matrix):
    fft_image = FFT.fft_2d(image)
```

```
    M, N = len(image), len(image[0])
    filtered_fft = [[fft_image[u][v] * filter_matrix[u][v] for v in range(N)] for u in range(M)]
    return FFT.ifft_2d(filtered_fft)

# 将二维矩阵乘以 (-1)^(x+y)
def multiply_by_neg1_pow_xy(matrix):
    M, N = len(matrix), len(matrix[0])
    return [[matrix[x][y] * ((-1) ** (x + y)) for y in range(N)] for x in range(M)]

# 主函数
def main():
    # 加载一个 512x512 的灰度图像（可以从文件加载或生成合成数据）
    import matplotlib.pyplot as plt
    import numpy as np
    from PIL import Image

    # 加载输入图像
    image = Image.open("第四章编程作业图.jpg").convert("L")  # 转换为灰度图像
    image = image.resize((512, 512))
    image_array = np.array(image, dtype=float)

    # (a) 原始图像
    original_image = image_array

    # (b) 零填充后的图像
    padded_image = np.pad(original_image, ((0, 512), (0, 512)), mode="constant")

    # (c) 乘以 (-1)^(x+y) 后的图像
    neg1_image = multiply_by_neg1_pow_xy(padded_image)

    # (d) 图像的傅里叶变换（频谱）
    fft_image = FFT.fft_2d(neg1_image)
    magnitude_spectrum = np.log(np.abs(fft_image) + 1)

    # (e) 高斯低通滤波器
    gaussian_filter = gaussian_low_pass_filter((1024, 1024), cutoff_ratio=0.95)

    # (f) 滤波器与频谱的乘积
    filtered_fft = [[fft_image[u][v] * gaussian_filter[u][v] for v in range(1024)] for u in range(1024)]
    filtered_magnitude_spectrum = np.log(np.abs(filtered_fft) + 1)

    # (g) 逆傅里叶变换后的图像并乘以 (-1)^(x+y)
    filtered_image = FFT.ifft_2d(filtered_fft)
```

```
    g_p = multiply_by_neg1_pow_xy(np.real(filtered_image))

    # (h) 提取最终的 MxN 结果
    # 将 g_p 转换为 NumPy 数组
    g_p_array = np.array(g_p)
    final_image = g_p_array[:512, :512]
    # 可视化结果
    plt.figure(figsize=(15, 10))

    plt.subplot(2, 4, 1)
    plt.title("(a) 原始图像")
    plt.imshow(original_image, cmap="gray")
    plt.axis("off")

    plt.subplot(2, 4, 2)
    plt.title("(b) 零填充后的图像")
    plt.imshow(padded_image, cmap="gray")
    plt.axis("off")

    plt.subplot(2, 4, 3)
    plt.title("(c) 乘以 (-1)^(x+y)")
    plt.imshow(neg1_image, cmap="gray")
    plt.axis("off")

    plt.subplot(2, 4, 4)
    plt.title("(d) 频谱图")
    plt.imshow(magnitude_spectrum, cmap="gray")
    plt.axis("off")

    plt.subplot(2, 4, 5)
    plt.title("(e) 高斯滤波器")
    plt.imshow(gaussian_filter, cmap="gray")
    plt.axis("off")

    plt.subplot(2, 4, 6)
    plt.title("(f) 滤波后的频谱")
    plt.imshow(filtered_magnitude_spectrum, cmap="gray")
    plt.axis("off")

    plt.subplot(2, 4, 7)
    plt.title("(g) g_p 图像")
    plt.imshow(g_p, cmap="gray")
    plt.axis("off")

    plt.subplot(2, 4, 8)
    plt.title("(h) 最终结果")
    plt.imshow(final_image, cmap="gray")
    plt.axis("off")
```

```
    plt.show()

if __name__ == "__main__":
    main()
```

结果分析



结果图

本实验采用高斯低通滤波器对图像进行了频域滤波操作，结果如图所示：

1. **(a) 原始图像**
   原始图像是一个标准的测试图像，包含了不同频率的细节信息以及明显的高频分量。
2. **(b) 零填充后的图像**
   通过对原始图像进行零填充，图像尺寸从 512×512 扩展到了 1024×1024。零填充的目的是为了避免频域变换中的周期性卷绕效应，同时增加频域分辨率。
3. **(c) 乘以 $(-1)^{x+y}$ 的图像**
   通过乘以 $(-1)^{x+y}$，将图像在频域中的低频分量移动到频谱中心。这一操作为后续频域滤波提供了便利。

4. **(d) 频谱图**
   对处理后的图像进行傅里叶变换后得到频谱图。图中可以观察到频率分量的分布：低频分量集中于中心位置，而高频分量分布在边缘。亮点表示幅值较大的频率分量，表明原图中包含显著的高频细节。
5. **(e) 高斯低通滤波器**
   高斯低通滤波器传递函数显示了一个实对称的中心高、边缘低的结构。这种滤波器有效地保留了图像中的低频分量（图像整体结构）并衰减了高频分量（图像细节和噪声）。
6. **(f) 滤波后的频谱**
   频谱乘以高斯低通滤波器后，高频分量显著衰减，仅保留了低频部分。可以观察到频谱亮度明显减弱，尤其是边缘区域，验证了滤波器的低通效果。
7. **(g) gp 图像**
   对滤波后的频谱进行逆傅里叶变换，并通过乘以 $(-1)^{x+y}$ 将低频分量移动回原始位置后，得到 gp 图像。可以看到，图像细节有所模糊，但整体轮廓和低频信息得到了保留。
8. **(h) 最终结果**
   从 gp 图像中裁剪出原图大小区域，得到最终的滤波结果图像。相比原始图像，高频噪声和细节得到了平滑处理，图像更为柔和，适合用于需要降低高频噪声的场景。

总结

实验结果表明，高斯低通滤波器能够有效地保留图像的低频成分，去除高频噪声和细节，实现图像的平滑处理。

- **优点**：滤波效果自然，图像边缘无明显伪影。
- **不足**：对高频信息（如边缘和细节）有一定的损失，图像的锐度降低。