西安交通大學

XI'AN JIAOTONG UNIVERSITY

数字图像处理实验报告

项目名称：图像的频域滤波

姓名：XXX

班级：XXXXXXX

学号：XXXXXXXXXX

摘　要

频域滤波综合运用傅里叶变换和卷积性质，把图像变换到频域对图像进行处理，是一种广为有效的图像滤波手段，经常被运用到图像噪声滤除和图像轮廓处理上。

本文基于 sciPy、numPy 和 scikit-image 等 Python 矩阵分析和信号处理有关组件，自行编写了高斯高通、高斯低通、巴特沃斯高通、巴特沃斯低通、拉普拉斯高通和锐化掩模滤波等滤波函数，分别对给定图像进行滤波处理，并对测试结果进行分析，最后比较并讨论了空域滤波器与频域滤波器这两者的异同。

一 题目 1

1.1 题目

频域低通滤波器：设计低通滤波器包括巴特沃斯和高斯（选择合适的半径，计算功率谱比），平滑测试图像 test1 和 test2，并分析各自优缺点。

1.2 解答

频率域中的滤波过程可总结为：

（1） 已知一幅大小为$M \times N$的输入图像$f(x,y)$，通过

$$P = 2M,\ Q = 2N$$

分别得到填充零后的尺寸$P,\ Q$；

（2） 使用零填充、镜像填充或复制填充，形成大小为$P \times Q$的填充后的图像

$f_p(x,y)$；

（3） 将$f_p(x,y)$乘以$(-1)^{x+y}$，使得傅里叶变换位于$P \times Q$大小的频率矩形的中心；

（4） 计算上步得到的图像的 DFT，即得$F(u,v)$；

（5） 构建一个实对称滤波器的传递函数$H(u,v)$，其大小为$P \times Q$，中心在

$\left(\frac{P}{2},\ \frac{Q}{2}\right)$处；

（6） 采用对应像素相乘得到$G(u,v) = H(u,v)F(u,v)$，即

$$G(i,k) = H(i,k)F(i,k) \quad i = 0,1,...,M-1,\ k = 0,1,...,N-1;$$

（7） 计算$G(u,v)$的 IDFT 得到滤波后的大小为$P \times Q$的图像：

$$g_p(x,y)=\{\mathrm{Re}[\mathscr{F}^{-1}[G(u,v)]]\}(-1)^{x+y};$$

（8） 从$g_p(x,y)$的左上象限提取一个大小为$M\times N$的区域，得到与输入图像大小相同的滤波后的结果$g(x,y)$。

1.2.1 高斯低通滤波器

高斯低通滤波器的传递函数有如下形式

$$H(u,v)=e^{-\frac{D^2(u,v)}{2D_0^2}}$$

其中$D(u,v)$为频率域中的点(u,v)到$P\times Q$矩形中心的距离，即

$$D(u,v)=\sqrt{\left(u-\frac{P}{2}\right)^2+\left(v-\frac{Q}{2}\right)^2}$$

而D_0为截止频率。

以下，我们取$D_0=50$，直接使用 sciPy 和 numPy 等相关组件编写函数，对 test1 和 test2 进行频率域的高斯低通滤波，结果分别如下所示。

针对 test1：

图1　test1 高斯低通滤波

test1 滤波前后的频谱图像如下图所示。

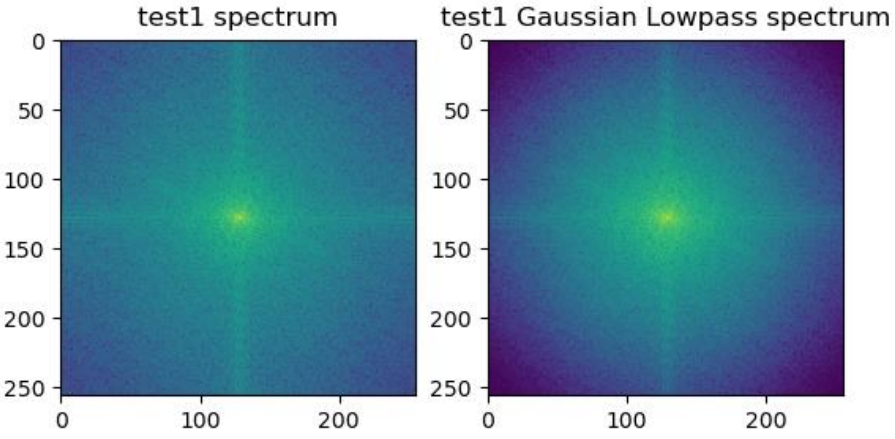

图2　test1 高斯低通滤波前后频谱

计算出功率谱比为$p=0.9869657$。

针对 test2：

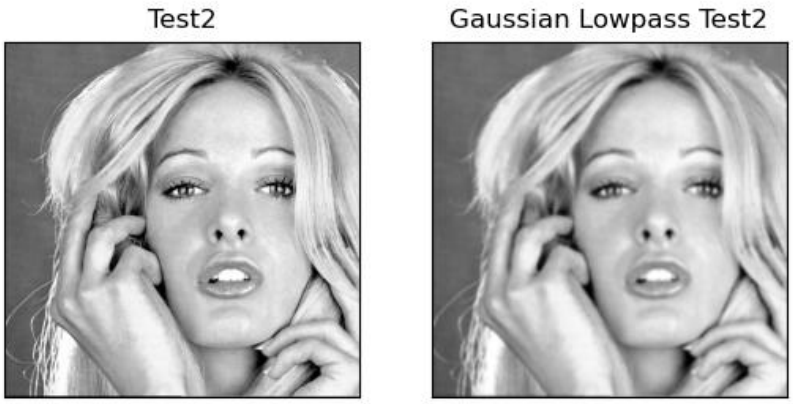

图3　test2 高斯低通滤波

test2 滤波前后的频谱图像如下图所示。

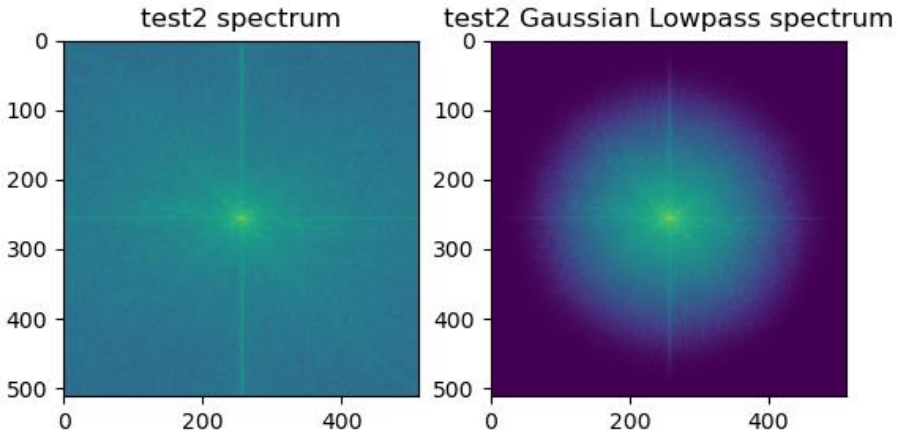

图4　test2 高斯低通滤波前后频谱

计算出功率谱比为$p=0.9870436$。

经高斯低通滤波，两幅图像均明显变得平滑，使用高斯低通滤波器没有振铃效应（与理想低通滤波器相比）。且截止频率半径D_0越小，图像越模糊，对D_0的具体选择取决于图片想要达成怎样的滤波效果。

1.2.2 巴特沃斯低通滤波器

截止频率位于距矩形中心D_0处的 n 阶巴特沃斯低通滤波器的传递函数为

$$H(u,v)=\frac{1}{1+\left[\frac{D(u,v)}{D_0}\right]^{2n}}$$

其中$D(u,v)$定义同前。

以下取$D_0=50$，阶数$n=6$，分别对 test1 和 test2 进行巴特沃斯低通滤波，其结果如下所示。

针对 test1：

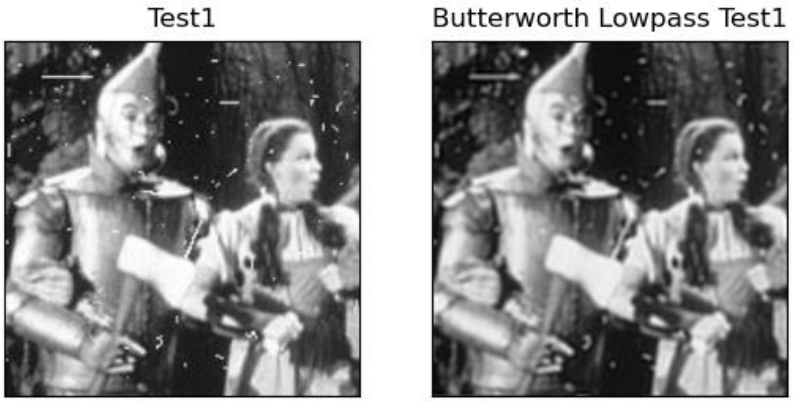

图5　test1 巴特沃斯低通滤波

test1 滤波前后的频谱图像如下图所示。

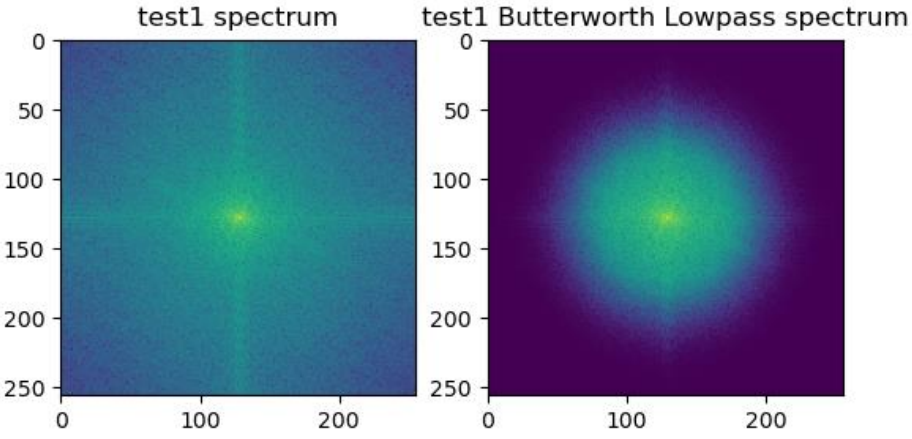

图6　test1 巴特沃斯低通滤波前后频谱

计算出功率谱比为$p=0.9944839$。

针对 test2：

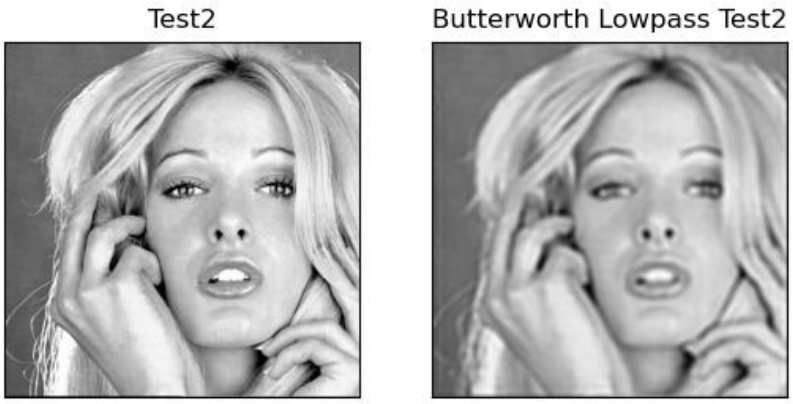

图7　test2 巴特沃斯低通滤波

test2 滤波前后的频谱图像如下图所示。

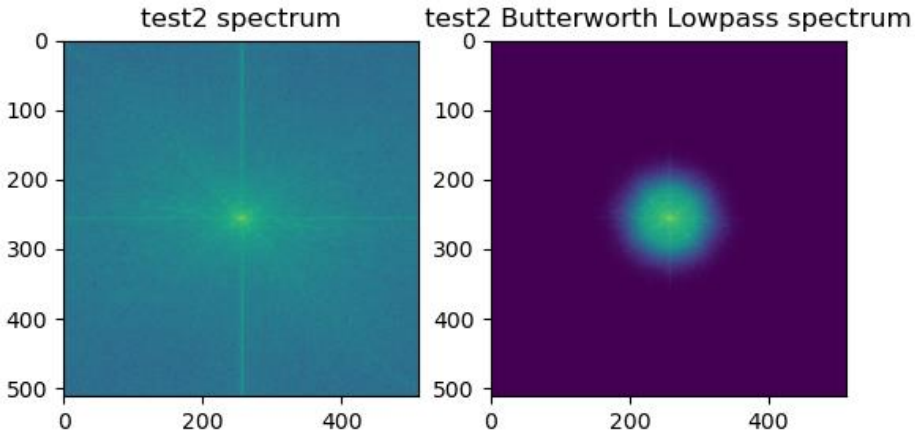

图8　test2 巴特沃斯低通滤波前后频谱

计算出功率谱比为$p=0.9884526$。

可见，使用巴特沃斯低通滤波器对图像仍然有良好的低通滤波器效果，且依然是D_0越小，滤波效果越明显，图像越模糊。巴特沃斯低通滤波器的阶数 n 越大，其滤波频谱的过渡带越陡峭，过渡区域越窄，而较高阶数的巴特沃斯低通滤波器会略微有一定的振铃现象。

综合高斯滤波器和巴特沃斯滤波器的滤波效果来看，注意到理想低通滤波器的滤波非常尖锐，则高斯低通滤波器的滤波则非常平滑，而低通滤波器介于两者之间，当巴特沃斯低通滤波器的阶数较高时，十分接近于理想低通滤波器，而阶数较低时，则接近于高斯低通滤波器。

二 题目 2

2.1 题目

频域高通滤波器：设计高通滤波器包括巴特沃斯和高斯，在频域增强边缘。选择半径和计算功率谱比，测试图像 test3 和 test4，并分析各自优缺点。

2.2 解答

一般而言，高通滤波器传递函数$H_{HP}(u,v)$可由低通滤波器传递函数$H_{LP}(u,v)$推导得到，公式为：

$$H_{HP}(u,v)=1-H_{LP}(u,v)$$

2.2.1 高斯高通滤波器

高斯高通滤波器的传递函数为

$$H(u,v)=1-e^{-\frac{D^2(u,v)}{2D_0{}^2}}$$

其中D_0为截止频率。

以下选取$D_0=50$，分别对 test3 和 test4 进行高斯高通滤波，结果如下。

针对 test3：

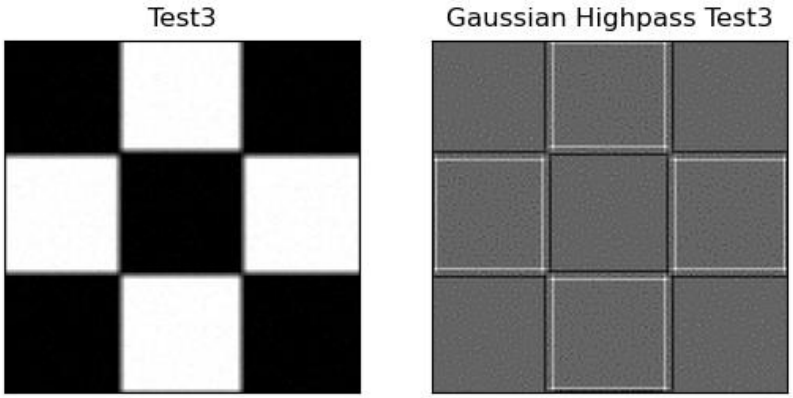

图9　test3 高斯高通滤波

test3 滤波前后的频谱图像如下图所示。

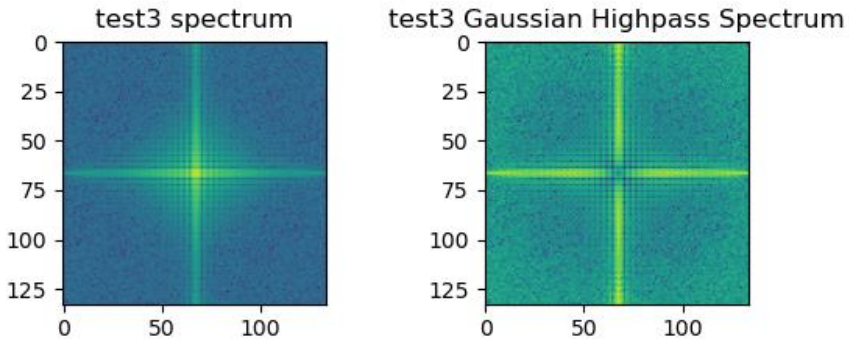

图10　test3 高斯高通滤波前后频谱

计算出功率谱比为$p=0.00026782378$。

针对 test4：

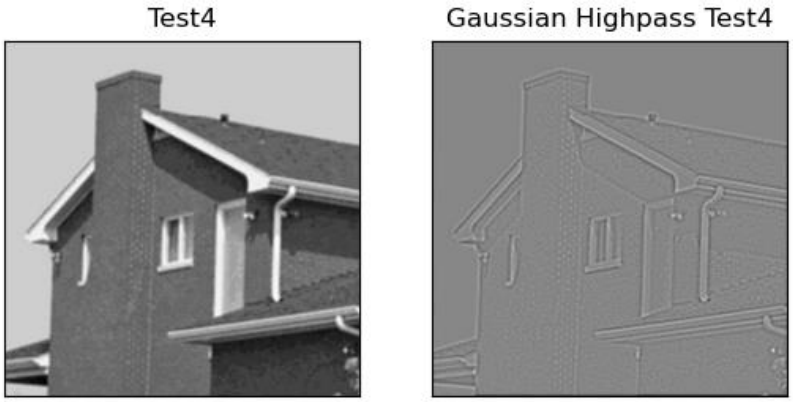

图11　test4 高斯高通滤波

test4 滤波前后的频谱图像如下图所示。

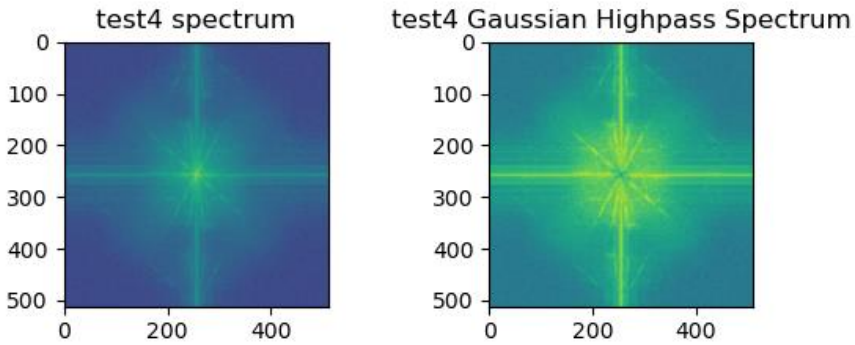

图12　test4 高斯高通滤波前后频谱

计算出功率谱比为$p=0.00151070757$。可见高斯高通滤波器能有效滤除图像的低频分量，提取图像的高频边缘特征。且D_0越大，图像的边缘特征提取越清晰。

2.2.2 巴特沃斯高通滤波器

巴特沃斯高通滤波器的传递函数为

$$H(u,v)=\frac{1}{1+\left[\frac{D_0}{D(u,v)}\right]^{2n}}$$

其中 n 为滤波器阶数。

以下选取$D_0=50$，$n=5$，分别对 test3 和 test4 进行巴特沃斯高通滤波，结果如下。

针对 test3：

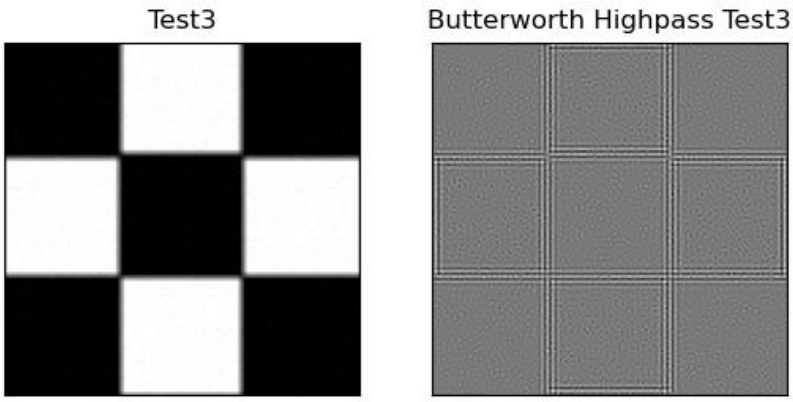

图13　test3 巴特沃斯高通滤波

test3 滤波前后的频谱图像如下图所示。

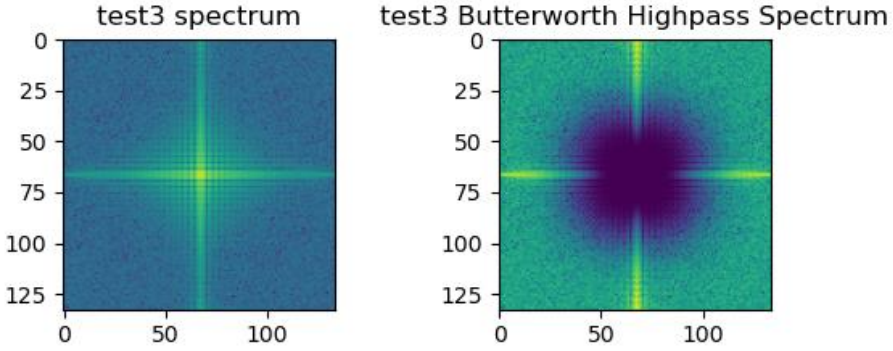

图14　test3 巴特沃斯高通滤波前后频谱

计算出功率谱比为$p=0.00032997577$。

针对 test4：

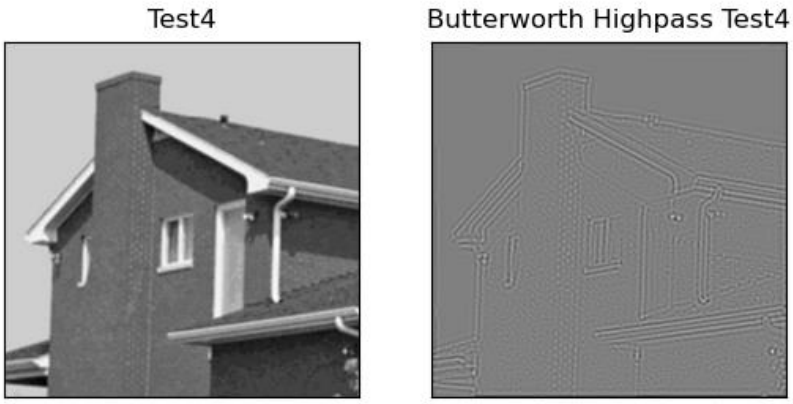

图15　test4 巴特沃斯高通滤波

test4 滤波前后的频谱图像如下图所示。

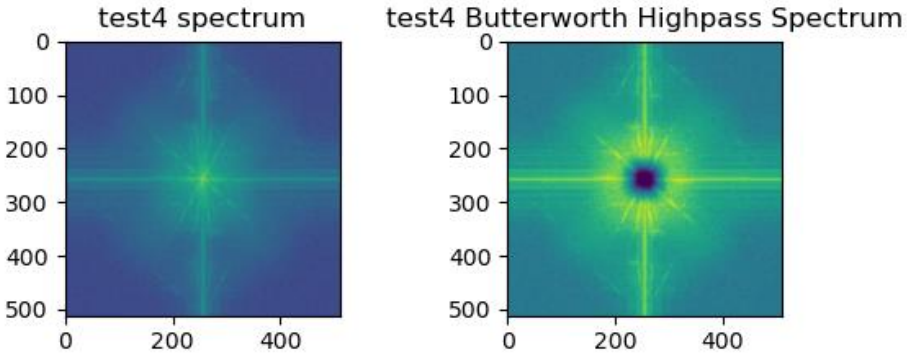

图16　test4 巴特沃斯高通滤波前后频谱

计算出功率谱比为$p=0.00205337992$。

由上述结果可见，一个巴特沃斯 5 阶高通滤波器提供了从低频到高频的尖锐但平滑

的过渡，这种滤波特性介于理想滤波器和高斯滤波器之间。以上实验的高斯高通滤波效果略微由于巴特沃斯高通滤波，但D_0太大时，高斯滤波后图像可能变得过暗，可能需要对图像进行阈值处理。

三 题目 3

3.1 题目

其他高通滤波器：拉普拉斯和 Unmask，对测试图像 test3 和 test4 滤波，并分析各自优缺点。

3.2 解答

3.2.1 拉普拉斯滤波器

拉普拉斯滤波器的传递函数为：

$$H(u,v) = -4\pi^2 D^2(u,v)$$

其中$D(u,v)$定义同前。

以下分别对 test3 和 test4 使用拉普拉斯滤波器，结果如下所示。

针对 test3：

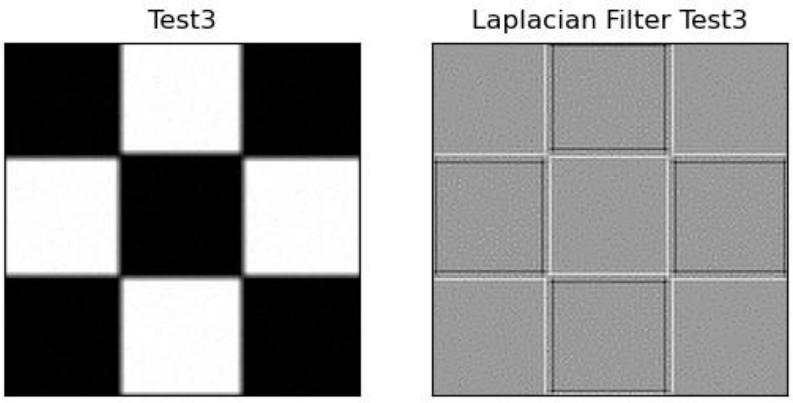

图17　test3 拉普拉斯滤波

test3 滤波前后的频谱图像如下图所示。

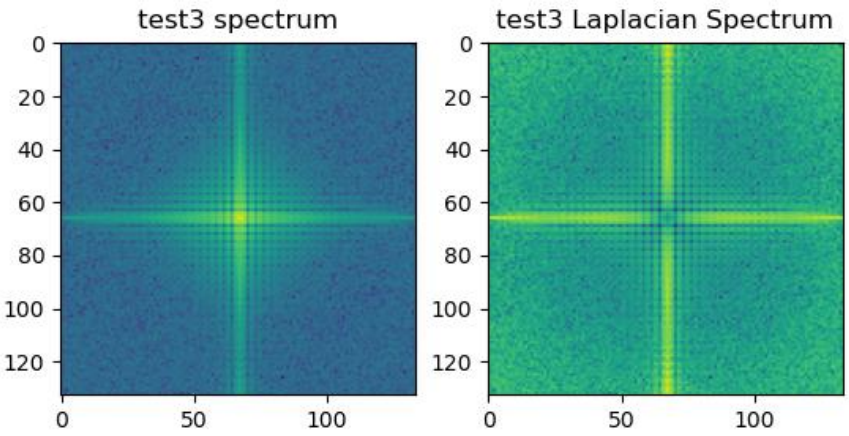

图18　test3 拉普拉斯滤波前后频谱

针对 test4：

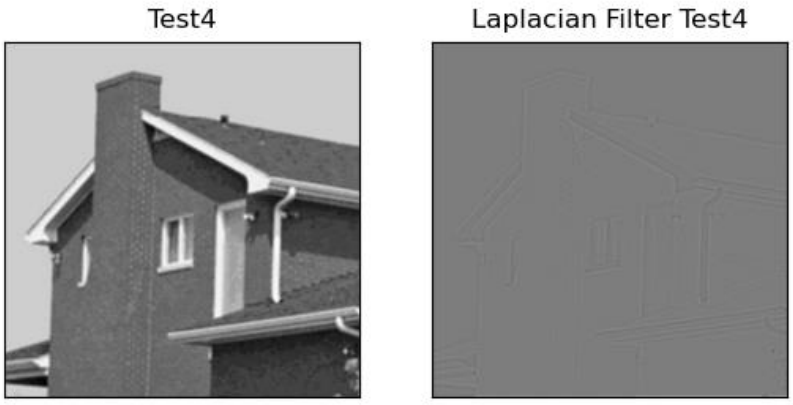

图19　test4 拉普拉斯滤波

test4 滤波前后的频谱图像如下图所示。

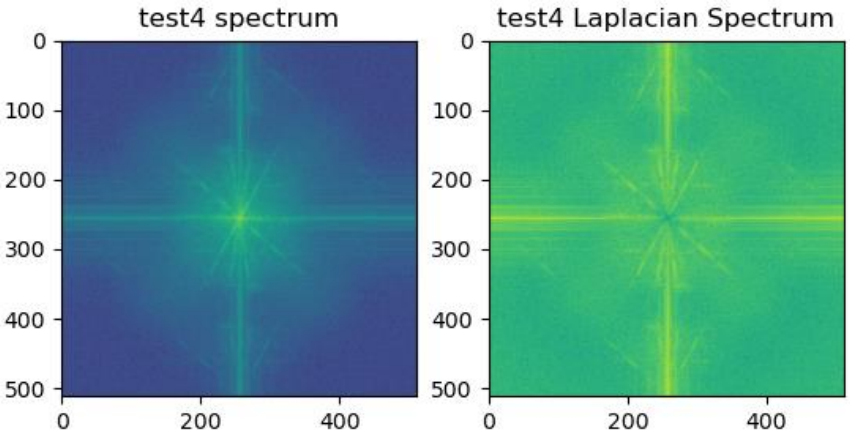

图20　test4 拉普拉斯滤波前后频谱

可见，使用拉普拉斯高通滤波器滤波后，图像的边缘有一定的突出，但滤波效果整体不如高斯高通和巴特沃斯高通滤波，滤波后图像可能有部分杂乱的线条。

3.2.2 Unsharp Masking 滤波器

Unsharp Masking 高通滤波器使用如下表达式：

$$g(x,y)=\mathcal{F}^{-1}\{[k_1+k_2*H_{HP}(u,v)]*F(u,v)\}$$

以下取$k_1=1$，$k_2=1$，高通滤波器选取高斯高通滤波器，其中$D_0=50$。

分别对 test3 和 test4 使用 Unsharp Masking 滤波器，结果如下所示。

针对 test3：

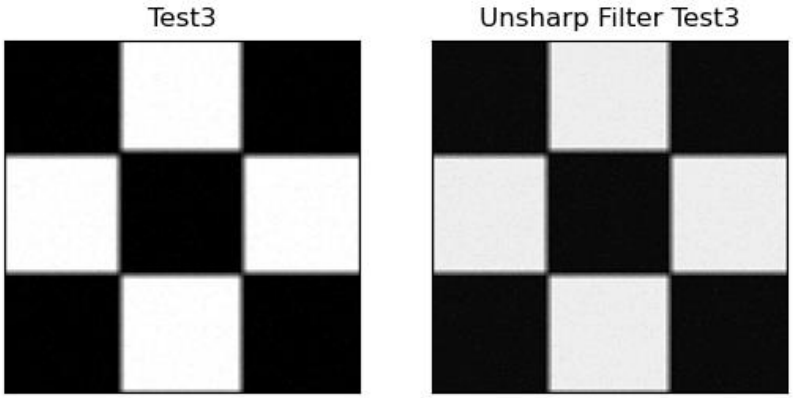

图21　test3 Unmask 滤波

test3 滤波前后的频谱图像如下图所示。

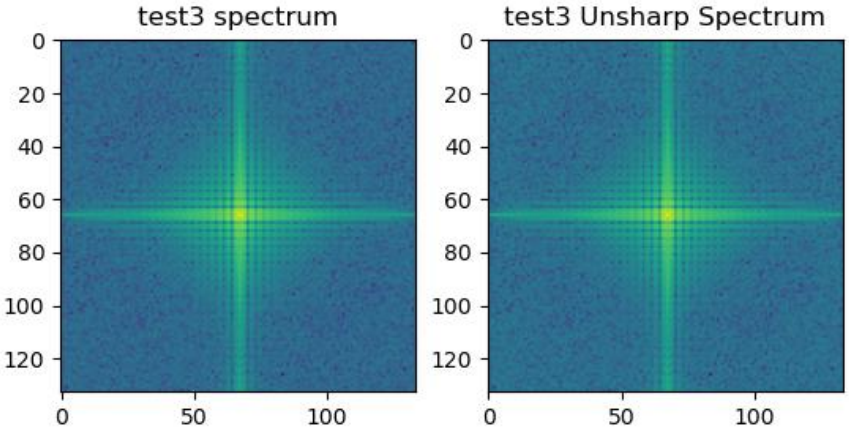

图22　test3 Unmask 滤波前后频谱

针对 test4：

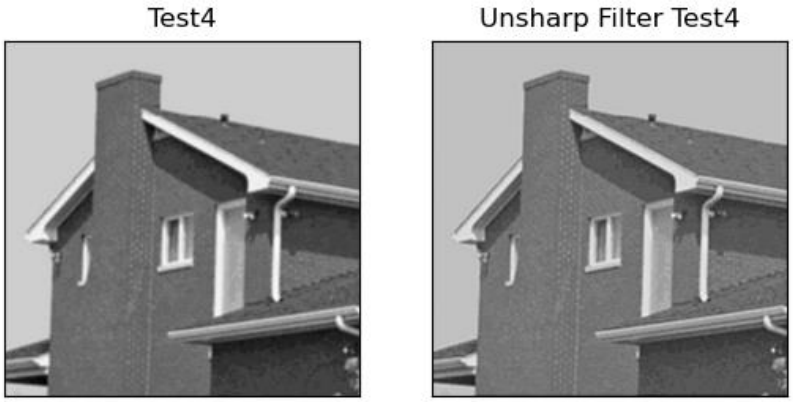

图23　test4 Unmask 滤波

test4 滤波前后的频谱图像如下图所示。

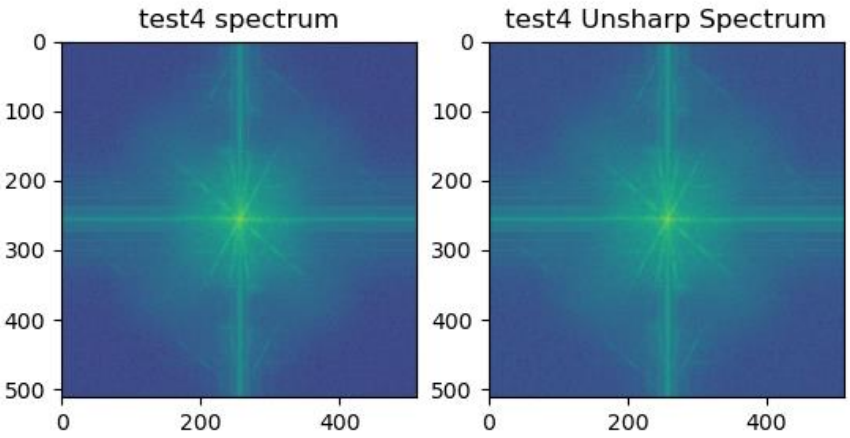

图24　test4 Unmask 滤波前后频谱

可以看出，通过 Unmask 高通滤波器分别对两幅图像进行高通滤波，图像的边缘产

生了一定的增强效果，使得图像的整体对比度有略微的提高，其效果不同于拉普拉斯滤波器，Unmask 良好地保留了图像的外形。

四 小结

空域滤波本质上是滤波函数与输入图像进行卷积，而频域滤波则是滤波函数与输入图像的傅里叶变换进行相乘。两者的理论基础分别是卷积定理和离散傅里叶变换。

频率域增强技术和空域增强技术有着密切的联系：空域增强技术可借助频域概念来分析和帮助设计，而诸多的空域增强技术也可以通过频域来变换和实现。

图像的空域滤波是用各种模板直接与图像进行卷积运算，实现对图像的处理，这种方法直接对图像在空间操作，操作上较为简单。而图像处理若在频率域进行，则要把空间域图像变换的掩膜卷积的形式，变换得到频率域的矩阵点乘形式，以得到想要的结果。图像频域滤波。亦即，频域要先把图像转换到频域空间，然后对不同的频率点进行滤波。例如，对于图像边缘的提取，在空间域滤波中我们使用拉普拉斯核进行卷积以进行提取，在频域内，则应该使用一个拉普拉斯传递函数与图像频域相乘完成任务。

如果将空域上的模板进行 DFT 得到频域上的模板，则用空域模板进行空域滤波和用得到的频域模板进行频域滤波最后结果是一样的，两种方法在一些情境下可以互换。但需要考虑的是，将原始图像与空域模板进行卷积运算，得到卷积结果的长度要比原来的图像长，即使对图像和模板进行填充，得到卷积结果的第一位，也不是模板在原始图像第一个像素处的卷积。因而要想得到和空域滤波器相同的结果，在填充和频域滤波之后提取图像时，需要将得到的处理结果的边缘去掉。

由此可见，空域滤波是局部滤波，频域滤波是全局滤波。因为空域滤波是只和局部信息有关，但是频域滤波是和整体图像信息有关，选用空域还是频域，哪种方法效果好，是和图片信息本身有关的，需要具体问题具体分析。

五 附录：所有代码

5.1 导入有关库

```
import cv2

from skimage import io
import numpy as np
from numpy.linalg import norm  # norm 计算二范数
from scipy.signal.windows import gaussian
from scipy.fftpack import fft2, ifft2, fftshift, ifftshift
from math import exp, dist  # dist 返回两点间欧几里得距离
import matplotlib.pyplot as plt
```

5.2 相关函数定义

5.2.1 高斯滤波

```
# 高斯滤波

def gaussian_frequency_filter(img, d0, is_lowpass=True):
    base_space = np.zeros(img.shape[:2])
    rows, columns = img.shape[:2]
    center_point = (rows / 2, columns / 2)

    for x in range(columns):
        for y in range(rows):
            if is_lowpass is True:
                base_space[y, x] = exp(((-dist((y, x), center_point) ** 2) / (2 *
(d0 ** 2))))
            else:
                base_space[y, x] = 1 - exp(((-dist((y, x), center_point) ** 2) /
(2 * (d0 ** 2))))

    centered = fftshift(fft2(img))
    pass_center = centered * base_space
    passed = ifft2(ifftshift(pass_center)).real
    ratio = calc_spectral_ratio(centered, pass_center)
    return passed, ratio
```

5.2.2 巴特沃斯滤波

```
# 巴特沃斯滤波
def butterworth_frequency_filter(img, d0, order, is_lowpass=True):
    base_space = np.zeros(img.shape[:2])
    rows, columns = img.shape[:2]
    center_point = (rows / 2, columns / 2)
    for x in range(columns):
        for y in range(rows):
            if is_lowpass is True:
                base_space[y, x] = 1 / (1 + (dist((y, x), center_point) / d0) **
(2 * order))
            else:
                base_space[y, x] = 1 - (1 / (1 + (dist((y, x), center_point) /
d0) ** (2 * order)))

    centered = fftshift(fft2(img))
    pass_center = centered * base_space
    passed = ifft2(ifftshift(pass_center)).real
    ratio = calc_spectral_ratio(centered, pass_center)
    return passed, ratio
```

5.2.3 拉普拉斯滤波

```
# 拉普拉斯高通滤波
def laplacian_frequency_filter(img):
    base_space = np.zeros(img.shape[:2])
    rows, columns = img.shape[:2]
    center_point = (rows / 2, columns / 2)
    for x in range(columns):
        for y in range(rows):
            base_space[y, x] = -4 * (np.pi ** 2) * (dist((y, x), center_point) **
2)

    centered = fftshift(fft2(img))
    pass_center = centered * base_space
```

```
    passed = ifft2(ifftshift(pass_center)).real
    return passed
```

5.2.4 Unsharp Masking 滤波

```
# unsharp 滤波（使用 Gaussian 高通滤波器）
def unsharp_frequency_filter(img, d0, k1, k2):
    base_space = np.zeros(img.shape[:2])
    rows, columns = img.shape[:2]
    center_point = (rows / 2, columns / 2)
    for x in range(columns):
        for y in range(rows):
            base_space[y, x] = k1 + k2 * (1 - exp(((-dist((y, x), center_point)
** 2) / (2 * (d0 ** 2)))))

    centered = fftshift(fft2(img))
    pass_center = centered * base_space
    passed = ifft2(ifftshift(pass_center)).real
    return passed
```

5.2.5 计算功率谱比

```
# 计算功率谱比
def calc_spectral_ratio(fft_shifted, fft_filtered):
    return (norm(fft_filtered) ** 2) / (norm(fft_shifted) ** 2)
```

5.2.6 显示两幅图像

```
# 显示两幅图像的函数
def display_image(image_list, title_list):
    for i in range(2):
        plt.subplot(1, 2, i + 1)
        plt.imshow(image_list[i], 'gray')
        plt.title(title_list[i])
```

```
        plt.xticks([]), plt.yticks([])
    plt.show()
```

5.2.7 绘制频谱

```
# 显示频谱的函数
def magnitude_spectrum(img):
    img_dft = cv2.dft(np.float32(img), flags=cv2.DFT_COMPLEX_OUTPUT)
    dft_shift = np.fft.fftshift(img_dft)
    spectrum = 20 * np.log10(cv2.magnitude(dft_shift[:, :, 0], dft_shift[:, :,
1]) + 1)  # +1 是为了显示更为清晰
    return spectrum
```

5.3 题目 1

```
# %% test1 Gaussian Lowpass Filter
test1 = io.imread("Resource/test1.pgm", as_gray=True)
test1GaussianLowpass, ratioGaussianLowpassTest1 =
gaussian_frequency_filter(test1, d0=50, is_lowpass=True)
print(ratioGaussianLowpassTest1)
images = [test1, test1GaussianLowpass]
titles = ['Test1', 'Gaussian Lowpass Test1']
display_image(images, titles)

# %% plot spectrum of test1 and Gaussian Lowpass
plt.subplot(121), plt.imshow(magnitude_spectrum(test1)), plt.title('test1
spectrum')
plt.subplot(122), plt.imshow(magnitude_spectrum(test1GaussianLowpass)),
plt.title('test1 Gaussian Lowpass spectrum')
plt.show()

# %% test1 Butterworth Lowpass Filter
test1ButterworthLowpass, ratioButterLowpassTest1 =
butterworth_frequency_filter(test1, d0=50, order=6, is_lowpass=True)
print(ratioButterLowpassTest1)
images = [test1, test1ButterworthLowpass]
titles = ['Test1', 'Butterworth Lowpass Test1']
```

```
display_image(images, titles)

# %% plot spectrum of test1 and Butterworth Lowpass
plt.subplot(121), plt.imshow(magnitude_spectrum(test1)), plt.title('test1
spectrum')
plt.subplot(122), plt.imshow(magnitude_spectrum(test1ButterworthLowpass)),
plt.title('test1 Butterworth Lowpass spectrum')
plt.show()

# %% test2 Gaussian Lowpass Filter
test2 = io.imread("Resource/test2.tif", as_gray=True)
test2GaussianLowpass, ratioGaussianLowpassTest2 =
gaussian_frequency_filter(test2, d0=50, is_lowpass=True)
print(ratioGaussianLowpassTest2)
images = [test2, test2GaussianLowpass]
titles = ['Test2', 'Gaussian Lowpass Test2']
display_image(images, titles)

# %% plot spectrum of test2 and Gaussian Lowpass
plt.subplot(121), plt.imshow(magnitude_spectrum(test2)), plt.title('test2
spectrum')
plt.subplot(122), plt.imshow(magnitude_spectrum(test2GaussianLowpass)),
plt.title('test2 Gaussian Lowpass spectrum')
plt.show()

# %% test2 Butterworth Lowpass Filter
test2ButterworthLowpass, ratioButterLowpassTest2 =
butterworth_frequency_filter(test2, d0=40, order=6, is_lowpass=True)
print(ratioButterLowpassTest2)
images = [test2, test2ButterworthLowpass]
titles = ['Test2', 'Butterworth Lowpass Test2']
display_image(images, titles)

# %% plot spectrum of test2 and Butterworth Lowpass
plt.subplot(121), plt.imshow(magnitude_spectrum(test2)), plt.title('test2
spectrum')
plt.subplot(122), plt.imshow(magnitude_spectrum(test2ButterworthLowpass)),
plt.title('test2 Butterworth Lowpass spectrum')
plt.show()
```

5.4 题目 2

```
# %% test3 Gaussian Highpass Filter

test3 = io.imread("Resource/test3_corrupt.pgm", as_gray=True)
test3GaussianHighpass, ratioGaussianHighPassTest3 =
gaussian_frequency_filter(test3, d0=50, is_lowpass=False)
print(ratioGaussianHighPassTest3)
images = [test3, test3GaussianHighpass]
titles = ['Test3', 'Gaussian Highpass Test3']
display_image(images, titles)

# %% plot spectrum of test3 and Gaussian Highpass
plt.subplot(121), plt.imshow(magnitude_spectrum(test3)), plt.title('test3
spectrum')
plt.subplot(122), plt.imshow(magnitude_spectrum(test3GaussianHighpass)),
plt.title('test3 Gaussian Highpass Spectrum')
plt.show()

# %% test3 Butterworth Highpass Filter
test3ButterworthHighpass, ratioButterHighpassTest3 =
butterworth_frequency_filter(test3, d0=50, order=5, is_lowpass=False)
print(ratioButterHighpassTest3)
images = [test3, test3ButterworthHighpass]
titles = ['Test3', 'Butterworth Highpass Test3']
display_image(images, titles)

# %% plot spectrum of test3 and Butterworth Highpass
plt.subplot(121), plt.imshow(magnitude_spectrum(test3)), plt.title('test3
spectrum')
plt.subplot(122), plt.imshow(magnitude_spectrum(test3ButterworthHighpass)),
plt.title('test3 Butterworth Highpass Spectrum')
plt.show()

# %% test4 Gaussian Highpass Filter
test4 = io.imread("Resource/test4 copy.bmp", as_gray=True)
test4GaussianHighpass, ratioGaussianHighPassTest4 =
gaussian_frequency_filter(test4, d0=50, is_lowpass=False)
print(ratioGaussianHighPassTest4)
images = [test4, test4GaussianHighpass]
titles = ['Test4', 'Gaussian Highpass Test4']
display_image(images, titles)
```

```
# %% plot spectrum of test4 and Gaussian Highpass
plt.subplot(121), plt.imshow(magnitude_spectrum(test4)), plt.title('test4
spectrum')
plt.subplot(122), plt.imshow(magnitude_spectrum(test4GaussianHighpass)),
plt.title('test4 Gaussian Highpass Spectrum')
plt.show()

# %% test4 Butterworth Highpass Filter
test4ButterworthHighpass, ratioButterHighpassTest4 =
butterworth_frequency_filter(test4, d0=50, order=5, is_lowpass=False)
print(ratioButterHighpassTest4)
images = [test4, test4ButterworthHighpass]
titles = ['Test4', 'Butterworth Highpass Test4']
display_image(images, titles)

# %% plot spectrum of test4 and Butterworth Highpass
plt.subplot(121), plt.imshow(magnitude_spectrum(test4)), plt.title('test4
spectrum')
plt.subplot(122), plt.imshow(magnitude_spectrum(test4ButterworthHighpass)),
plt.title('test4 Butterworth Highpass Spectrum')
plt.show()
```

5.5 题目 3

```
# %% test3 Laplacian Highpass Filter

test3Laplacian = laplacian_frequency_filter(test3)
images = [test3, test3Laplacian]
titles = ['Test3', 'Laplacian Filter Test3']
display_image(images, titles)

# %% plot spectrum of test3 and Laplacian
plt.subplot(121), plt.imshow(magnitude_spectrum(test3)), plt.title('test3
spectrum')
plt.subplot(122), plt.imshow(magnitude_spectrum(test3Laplacian)),
plt.title('test3 Laplacian Spectrum')
plt.show()

# %% test4 Laplacian Highpass Filter
test4Laplacian = laplacian_frequency_filter(test4)
```

```
images = [test4, test4Laplacian]
titles = ['Test4', 'Laplacian Filter Test4']
display_image(images, titles)

# %% plot spectrum of test4 and Laplacian
plt.subplot(121), plt.imshow(magnitude_spectrum(test4)), plt.title('test4
spectrum')
plt.subplot(122), plt.imshow(magnitude_spectrum(test4Laplacian)),
plt.title('test4 Laplacian Spectrum')
plt.show()


# %% test3 Unsharp Filter
test3Unsharp = unsharp_frequency_filter(test3, d0=50, k1=1, k2=1)
images = [test3, test3Unsharp]
titles = ['Test3', 'Unsharp Filter Test3']
display_image(images, titles)

# %% plot spectrum of test3 and Unsharp
plt.subplot(121), plt.imshow(magnitude_spectrum(test3)), plt.title('test3
spectrum')
plt.subplot(122), plt.imshow(magnitude_spectrum(test3Unsharp)), plt.title('test3
Unsharp Spectrum')
plt.show()

# %% test4 Unsharp Filter
test4Unsharp = unsharp_frequency_filter(test4, d0=50, k1=1, k2=1)
images = [test4, test4Unsharp]
titles = ['Test4', 'Unsharp Filter Test4']
display_image(images, titles)

# %% plot spectrum of test4 and Unsharp
plt.subplot(121), plt.imshow(magnitude_spectrum(test4)), plt.title('test4
spectrum')
plt.subplot(122), plt.imshow(magnitude_spectrum(test4Unsharp)), plt.title('test4
Unsharp Spectrum')
plt.show()
```