华北水利水电大学

North China University of Water Resources and Electric Power

《数字图像处理》
实验报告(3)

院　　系：　信息工程学院

专　　业：　人工智能

班　　级：　2020185

姓　　名：　高树林

学　　号：　202018526

指 导 教 师：　韩光辉

2022-2023 学年 第一学期

实验三 频域图像增强

一、实验目的

(1) 理解傅里叶变换的思想；

(2) 掌握傅里叶变换的 Python 实现方法；

(3) 理解并掌握频域滤波方法。

二、实验内容和要求

实验报告内容应格式统一(注意同一个大纲级别下的字体、字号、行间距等应统一，英文用 Times New Roman 字体)，整体美观整洁。实验代码可读性强，包含适量注释，说明求解思路和过程。

1. 编程实现傅里叶变换

分别使用 OpenCV 和 Numpy 工具包中的函数进行傅里叶变换，要求显示图像移中后的幅值谱和相位谱，以及重构后的图像。结果展示图中应包含原图、幅值谱、相位谱、重构图像。

提示：

（1）OpenCV 工具包相关函数：

① dst = cv2.dft(src,flags)

函数功能：实现傅里叶变换，输入图像需要转换为浮点格式，其输出结果是双通道的，第一个通道 dst[:,:,0]为傅里叶变换的实部，第二个通道 dst[:,:,1]为傅里叶变换的虚部。

具体函数说明和使用方法请查阅官方文档

② dst = cv2.magnitude(x,y)

函数功能：获得傅里叶变换的幅值谱。x 和 y 分别是傅里叶变换的实部和虚部。

③ dst = cv2.phase(x,y)

函数功能：获得傅里叶的相位谱。

④ dst = cv2.idft(src)

函数功能：进行傅里叶反变换。

（2）NumPy 工具包相关函数：

① dst = np.fit.fft2(src) 傅里叶变换

② dst = np.fit.fftshift(src) 低频移中

③ dst = np.fit.ifftshift(src) 低频移中的逆运算

④ dst = np.fit.abs(src) 幅值谱

⑤ dst = np.fit.angle() 相位谱

⑥ dst = np.fit.ifft2(src) 傅里叶逆变换

2. 编程给图像加噪后进行频域滤波

先给图像添加高斯噪声（0 均值，0.1 方差），再使用理想低通滤波器对图像进行低通平滑滤波，分析不同截止频率的滤波性能。结果展示部分应使程序输出的一组图像，包括原图、噪声图、噪声图幅值谱、滤波后幅值谱、重构图像等。

将完善后的程序代码放置于实验过程的代码部分。

提示：

添加噪声部分可参考前次实验内容。

核心代码参考(自行补全代码，仅供参考，可更改优化)：

```
# 导入需要的工具包
# 添加画图的中文支持
img = cv2.imread(_________________)  # 使用图像 iris.jpg
img0 = addGaussianNoise(_______ __)
f = np.fft.fft2(_______)
fshift = np.fft.fftshift(__________)
magnitude_spectrum0 = 20*np.log(1+np.abs(__________))

#进行理想低通滤波
r = __________        #截止频率的设置
[m,n] = fshift.shape
H = np.zeros((m,n), dtype=complex)  #滤波核
for i in range(m):
    for j in range(n):
        d=sqrt((i-m/2)**2+(j-n/2)**2)
        if d<r:
            H[i,j] = _______________
G = _______________ #理想低通滤波
magnitude_spectrum1 =20*np.log(1+np.abs(___))  #理想低通滤波后的幅值谱
f1 = np.fft.ifftshift(________)
img1 = _________________  #重构图像
# 自行添加可视化输出代码，幅值谱的显示需要取对数特殊处理
```

三、实验环境(学生填写)

（较详细地说明实验运行环境，包括操作系统、编程 IDE、编程语言及版本号、依赖的第三方类库及版本号，及其它可能影响重复实验的因素）

本实验环境分为硬件和软件两大方面叙述。

硬件方面：本实验室基于处理器为 Intel(R) Core(TM) i5-10210U CPU @

1.60GHz 2.11 GHz，基带 RAM 为 16.0 GB (15.8 GB 可用)，系统类型为 64 位操作系统，基于 x64 的处理器的 MagicBook 商务本上运行并得出相应结果和数据的。

软件方面：本实验基于 windows10 家庭中文版，版本号为 21H2，操作系统内部版本为 19044.2130 的 MagicBook 商务本。编程语言选择 python 语言，使用运行时版本号为 11.0.14.1+1-b2043.45 amd64 的 PyCharm 2022.1.1 (Professional Edition)集成开发环境编写代码。其中代码的 python 解释器版本为 Python 3.8.13 (default, Mar 28 2022, 06:59:08) [MSC v.1916 64 bit (AMD64)] :: Anaconda, Inc. on win32。引入 1.0.0.14 版本的 OpenCV-python 库，其中集成的 OpenCV 版本为 4.6.0。引入 Matplotlib3.5.3 版本的库。

四、实验过程(学生填写)

1. 编程实现傅里叶变换

(1)实验方案（设计思路说明）

本实验分两块进行，分别是利用 OpenCV 进行傅里叶变换和处理以及利用 Numpy 进行傅里叶变换和处理两个模块。

OpenCV 模块首先需要读取图像，经过傅里叶变换后调用 np.fft.fftshift()方法将图像中的低频部分移动到图像的中心。再读取变换后数据的的实部和虚部，分别对其使用 cv2.magnitude()方法和 cv2.phase()方法进行提取幅值谱和相位谱的操作。最后，对读取的图像经过傅里叶反变换进而得到最后的重构图像。

Numpy 模块首先需要读取图像，这一步需要用到 OpenCV.read()方法，经过傅里叶变换后调用 np.fft.fftshift()方法将图像中的低频部分移动到图像的中心。再读取变换后数据的的实部和虚部，分别对其使用 np.log(np.abs())方法和 np.log(np.angle())方法进行提取幅值谱和相位谱的操作。最后，对读取的图像经过傅里叶反变换进而得到最后的重构图像。

上述两个模块完成后，均利用 Matplotlib 工具包对其变化过程中生成的图像谱可视化出来，主要用到的方法为 matplotlib.imshow()方法。最后一定要利用 matplotlib.show()展示，否则不能正常显示。

(2)代码编写

```
1.  # 在下面编写傅里叶变换的实验代码
2.  import cv2
3.  import numpy as np
4.  import matplotlib.pyplot as plt
5.  from matplotlib.font_manager import FontProperties
6.  font = FontProperties(fname="work/img/simhei.ttf")
7.
8.  # 基于 numpy
9.  def dft_np(path):
10.     plt.subplots_adjust(left=None,bottom=None,right=None,top=None,wspace=0.5) # 设置子图左右和上下间隔，一般为 0.5 为佳
11.     original = cv2.imread(path, 0) # 转为灰度图
```

```
    dft = np.fft.fft2(original)
    dftShift = np.fft.fftshift(dft)    # 将图像中的低频部分移动到图像的中心
    result = np.log(np.abs(dftShift))  # 幅值谱
    phase = np.log(np.angle(dftShift)) # 相位谱
    ishift = np.fft.ifftshift(dftShift)    # 低频部分从图像中心移开
    iImg = np.fft.ifft2(ishift)              # 傅里叶反变换
    iImg = np.abs(iImg)

    plt.subplot(141), plt.imshow(original, cmap='gray'),plt.title('原图', fontproperties=font)
    plt.subplot(142), plt.imshow(result, cmap='gray'),plt.title('幅值谱', fontproperties=font)
    plt.subplot(143), plt.imshow(phase, cmap='gray'), plt.title('相位谱', fontproperties=font)
    plt.subplot(144), plt.imshow(iImg, cmap='gray'),plt.title('重构图', fontproperties=font)
    plt.show()


#基于 OpenCV
def dft_CV(path):
    plt.subplots_adjust(left=None,bottom=None,right=None,top=None,wspace=0.5) # 设置子图左右和上下间隔，一般为 0.5 为佳
    original = cv2.imread(path, 0) # 转为灰度图
    dft = cv2.dft(np.float32(original), flags=cv2.DFT_COMPLEX_OUTPUT)
    dftShift = np.fft.fftshift(dft) # 将图像中的低频部分移动到图像的中心
    amplitude = np.log(cv2.magnitude(dftShift[:, :, 0], dftShift[:, :, 1]))
    phase = np.log(cv2.phase(dftShift[:, :, 0], dftShift[:, :, 1]))

    ishift = np.fft.ifftshift(dftShift) # 低频部分从图像中心移开
    iImg = cv2.idft(ishift) # 傅里叶反变换
    iImg = cv2.magnitude(iImg[:, :, 0], iImg[:, :, 1]) # 转化为空间域

    plt.subplot(245), plt.imshow(original, cmap='gray'), plt.title('原图', fontproperties=font)
    plt.subplot(246), plt.imshow(amplitude, cmap='gray'), plt.title('幅值谱', fontproperties=font) # 幅值谱
    plt.subplot(247), plt.imshow(phase, cmap='gray'), plt.title('相位谱', fontproperties=font)
    plt.subplot(248), plt.imshow(iImg, cmap='gray'), plt.title('重构图', fontproperties=font)
    plt.show()

if __name__ == '__main__':

    print(111)
    path = 'work/img/Lenna.png'
    print('基于 numpy')
    dft_np(path)
    print('基于 OpenCV')
    dft_CV(path)
```

(3)调试（编码实现过程中遇到的错误，及调试解决等内容）

问题 1：在代码调试的过程中，幅值图输出为下图 1 所示。

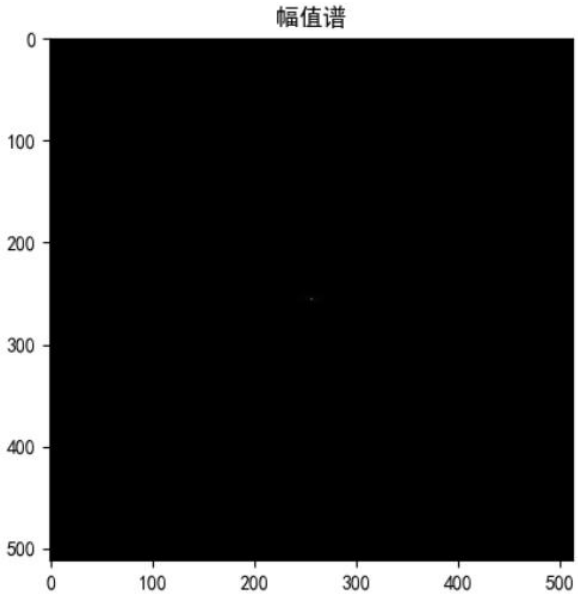

图 1 问题 1 所指的幅值图出错

解决 1：这是因为一般输出的幅值图为 log 类型的，在代码中应该对幅值图的数据做一个取对数的操作才能正常显示幅值图。解决方案如下图 2 所示。

图 2 问题 1 解决方案

问题 2：在代码调试的过程中，我按照给定的 dst = np.fit.angle()函数对傅里叶变换后的图像进行处理，但是结果确报错，报错结果如下图 3 所示。

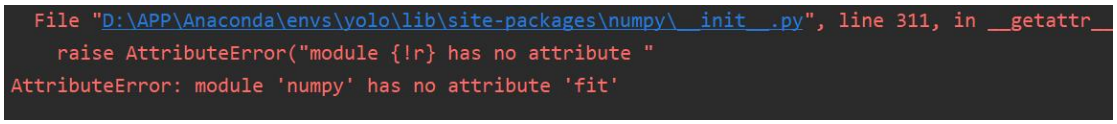

图 3 问题 2 报错内容

解决 2：这是因为在 numpy 库里面没有子 fit 库，如果需要在 numpy 中引用到 angle 方法，直接使用 numpy.angle()即可。于是我修改代码结果运行成功。修改部分的截图如下试图 4 所示。

图 4 问题 2 解决方案

(4)实验结果（实验运行结果截图等体现实验结果的内容）

结果 1：利用 OpenCV 库经过傅里叶变换处理后再由低频处理得到的幅值谱

如下图 5 所示

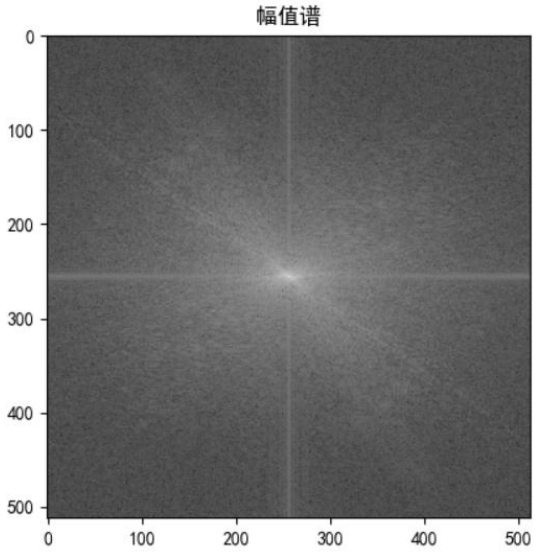

图 5 OpenCV 库处理得到的幅值谱

结果 2：利用 OpenCV 库经过傅里叶变换处理后再由低频处理得到的相位谱如下图 6 所示。

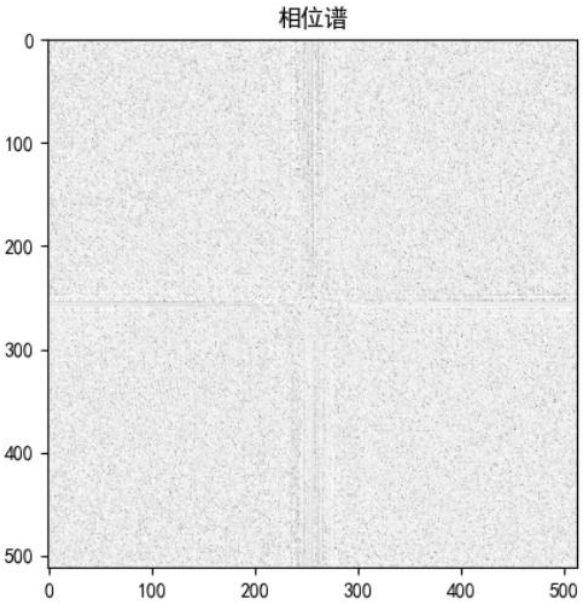

图 6 OpenCV 库处理得到的相位谱

结果 3：利用 OpenCV 库经过傅里叶变换后经过逆傅里变换处理得到的重构图如下图 7 所示。

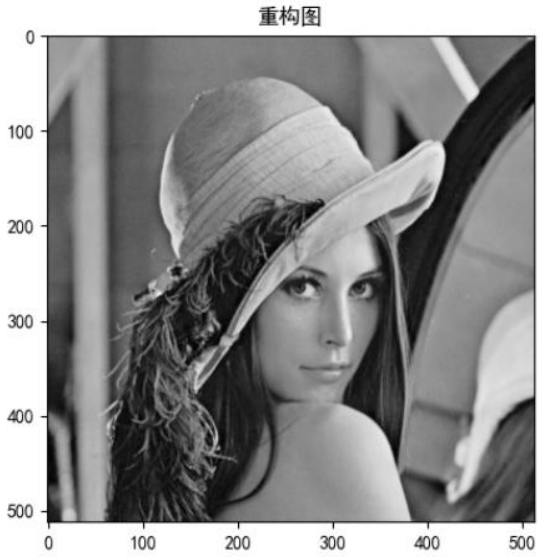

图 7 OpenCV 库处理得到的重构谱

结果 4：利用 OpenCV 工具包对原图像进行傅里叶变换处理后经逆傅里叶变换得到的幅值图、相位图、重构图展示如下图 8 所示。

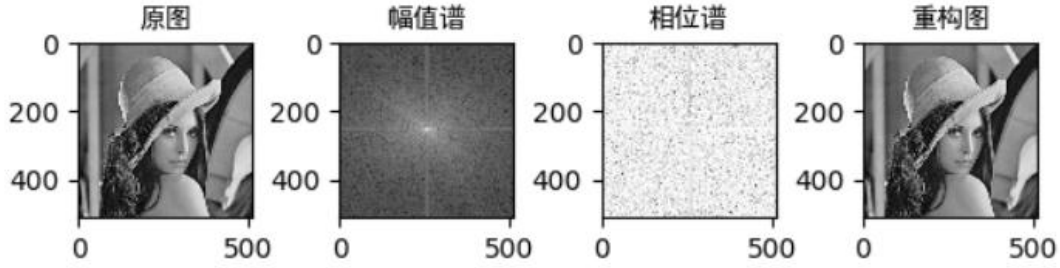

图 8 OpenCV 傅里叶变换得到的幅值图、相位图、重构图

结果 5：利用 numpy 库经过傅里叶变换处理后再由低频处理得到的幅值谱如下图 9 所示。

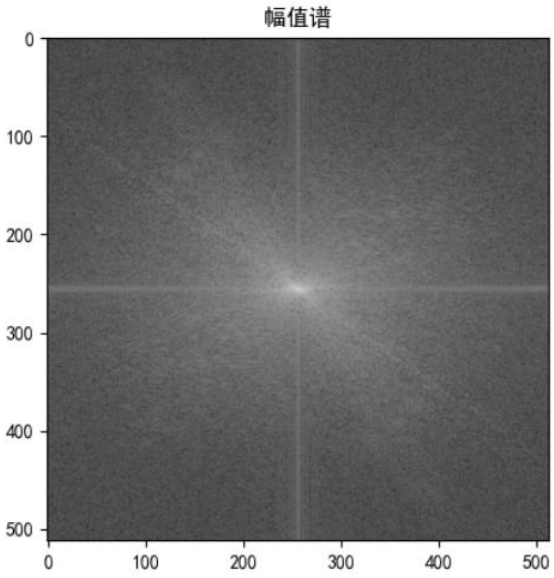

图 9 numpy 库处理得到的幅值谱

结果 6：利用 numpy 库经过傅里叶变换处理后再由低频处理得到的相位谱如下图 10 所示。

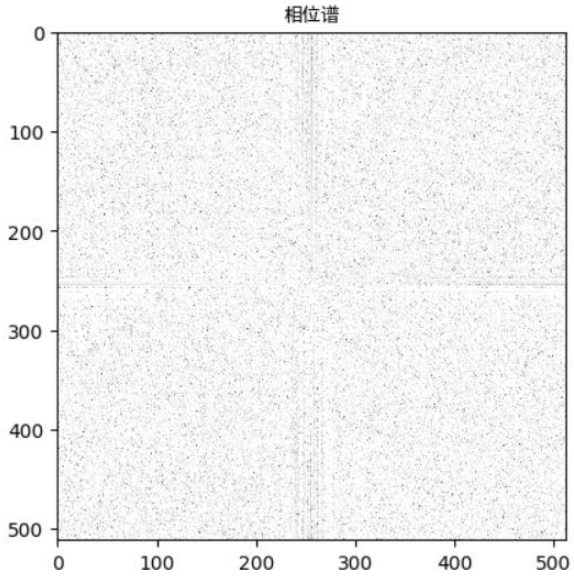

图 10 numpy 库处理得到的相位谱

结果 7：利用 numpy 库经过傅里叶变换处理后再由低频处理得到的重构图如下图 11 所示

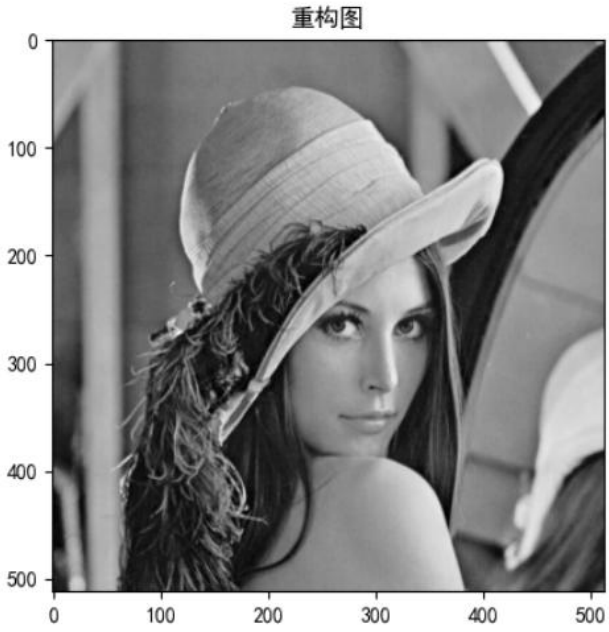

图 11 numpy 库处理得到的重构图

结果 8：利用 numpy 工具包对原图像进行傅里叶变换处理后经逆傅里叶变换得到的幅值图、相位图、重构图展示如下图 12 所示。

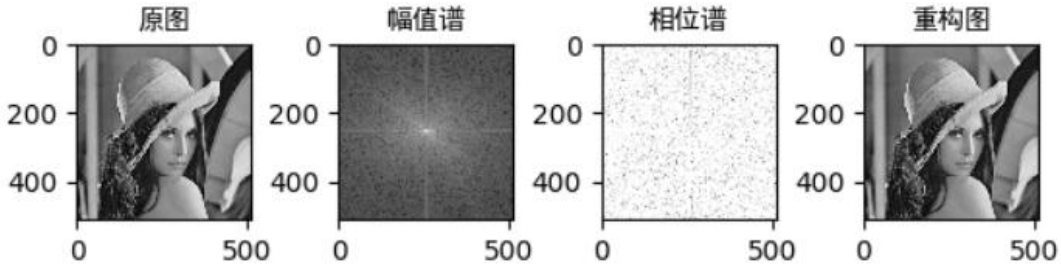

图 12 numpy 傅里叶变换得到的幅值图、相位图、重构图

(5)分析与总结（与本次实验相关的分析与总结）

通过本次实验，我加深了对傅里叶变换的认识，巩固了 matplotlib 工具包以及 numpy 工具包的操作技能。

傅里叶变换的作用是将图像从空间域转换到频率域，其实质是将图像的灰度分布函数变换为图像的频率分布函数。图像的频率是表征图像中灰度变化的剧烈程度的指标。换句话说，他就是灰度在平面空间上的梯度，反映在傅里叶频谱图上就是能够看到敏感不一的点。一般来说，梯度大则该点的亮度强，否则该点的亮度弱。由于傅里叶变换涉及图像处理的多个方面，因此在数字图像处理中，它的应用范围很广。比如离散余弦变换，gabor 变换，图像增强去噪，图像分割、压缩、提取等方面都涉及到傅里叶变换。

2. 编程给图像加噪后进行频域滤波

(1)实验方案（设计思路说明）

首先向图像中添加高斯噪声，添加噪声后将图像转为灰度图。读取灰度图，对灰度图进行傅里叶变换，之后就能够得到该图像的幅值图。对加过噪声的原灰度图进行理想低通滤波，在滤波的过程中首先要定义 mask 卷积核，mask 卷积核的大小要和原图像一样大，因为后面要做矩阵的乘法运算。在得到低通滤波后数据之后，通过傅里叶逆变换能得到重构图，重构图经傅里叶变换就能得到滤波后的幅值谱。

(2)代码编写

理想低通滤波：

```
1.  import cv2
2.  import numpy as np
3.  from pylab import mpl
4.  from matplotlib import pyplot as plt
5.  from matplotlib.font_manager import FontProperties
6.  mpl.rcParams["axes.unicode_minus"] = False
7.  # font = FontProperties(fname="work/font/simhei.ttf")
8.  font = FontProperties(fname="../figures/font/simhei.ttf")
9.  plt.subplots_adjust(left=None, bottom=None, right=None, top=None, wsp
    ace=0.5, hspace=0.05)  # 设置子图左右和上下间隔，一般为 0.5 为佳
10.
11. # 读取图像
```

```
12. # img = cv2.imread('..\\figures\\img\\iris.jpg', 0)
13. img = cv2.imread('lenna.jpg', 0)
14.
15.
16. def Low_Pass_Filter(img):
17.     # 傅里叶变换
18.     dft = cv2.dft(np.float32(img), flags=cv2.DFT_COMPLEX_OUTPUT)
19.     fshift = np.fft.fftshift(dft)
20.     # 设置低通滤波器
21.     rows, cols = img.shape
22.     crow, ccol = int(rows / 2), int(cols / 2)  # 中心位置
23.     mask = np.zeros((rows, cols, 2), np.uint8)
24.     mask[crow - 30:crow + 30, ccol - 30:ccol + 30] = 1
25.     print(mask.shape)
26.     # 掩膜图像和频谱图像乘积
27.     f = fshift * mask
28.     # 傅里叶逆变换
29.     ishift = np.fft.ifftshift(f)
30.     iimg = cv2.idft(ishift)
31.     res = cv2.magnitude(iimg[:, :, 0], iimg[:, :, 1])
32.     return res
33.
34.
35. def addGaussianNoise(src, means, sigma):
36.     image = np.array(src / 255, dtype=float)
37.     noise = np.random.normal(means, sigma, image.shape)
38.     gauss_noise = image + noise
39.     if gauss_noise.min() < 0:
40.         low_clip = -1.
41.     else:
42.         low_clip = 0.
43.     gauss_noise = np.clip(gauss_noise, low_clip, 1.0)
44.     gauss_noise = np.uint8(gauss_noise * 255)
45.     return gauss_noise
46.
47.
48. def fz(img):  # 求幅值
49.     return np.log(np.abs(np.fft.fftshift(np.fft.fft2(img))))
50.
51.
52. if __name__ == '__main__':
53.     path = '..\\figures\\img\\iris.jpg'
54.     img = cv2.imread(path)
55.     img0 = img[:, :, 0]
```

```
    img1 = addGaussianNoise(img, 0, 0.1)[:, :, 0]  # 添加高斯噪声后的灰度图
    img2 = fz(img1)  # 噪声幅值图
    img3 = Low_Pass_Filter(img1)  # 低通滤波图
    img4 = fz(img3)  # 低通滤波幅值图

    plt.subplot(151), plt.imshow(img0, 'gray'), plt.title('原图', fontsize=10, fontproperties=font),plt.axis('off')
    plt.subplot(152), plt.imshow(img1, 'gray'), plt.title('高斯噪声', fontsize=10, fontproperties=font),plt.axis('off')
    plt.subplot(153), plt.imshow(img2, 'gray'), plt.title('高斯噪声幅值谱', fontsize=10, fontproperties=font),plt.axis('off')
    plt.subplot(154), plt.imshow(img4, 'gray'), plt.title('滤波后幅值谱', fontsize=10, fontproperties=font),plt.axis('off')
    plt.subplot(155), plt.imshow(img3, 'gray'), plt.title('低通滤波', fontsize=10, fontproperties=font),plt.axis('off')
    plt.show()
```

高斯低通滤波：

```
import cv2
import numpy as np
from pylab import mpl
import matplotlib.pyplot as plt
from matplotlib.font_manager import FontProperties
mpl.rcParams["axes.unicode_minus"] = False
# font = FontProperties(fname="work/font/simhei.ttf")
font = FontProperties(fname="../figures/font/simhei.ttf")

def addGaussianNoise(src, means, sigma):
    image = np.array(src / 255, dtype=float)
    noise = np.random.normal(means, sigma, image.shape)
    gauss_noise = image + noise
    if gauss_noise.min() < 0:
        low_clip = -1.
    else:
        low_clip = 0.
    gauss_noise = np.clip(gauss_noise, low_clip, 1.0)
    gauss_noise = np.uint8(gauss_noise * 255)
    return gauss_noise


def fz(img):  # 求幅值
    return np.log(np.abs(np.fft.fftshift(np.fft.fft2(img))))


```

```
if __name__ == '__main__':
    path = '..\\figures\\img\\iris.jpg'
    img0 = cv2.imread(path)  # 原图
    img1 = addGaussianNoise(img0, 0, 0.1)[:, :, 0]  # 噪声图
    img2 = fz(img1)  # 噪声幅值图
    img3 = cv2.GaussianBlur(img0, (3, 3), 1.3)[:, :, 0]  # 高斯滤波
    img4 = fz(img3)  # 高斯滤波幅值谱

    plt.subplot(151),plt.imshow(img0[:, :, 0], 'gray'), plt.title('原图', fontsize=10, fontproperties=font), plt.axis('off')  # 原图
    plt.subplot(152),plt.imshow(img1, 'gray'), plt.title('噪声图', fontsize=10, fontproperties=font), plt.axis('off')  # 噪声图
    plt.subplot(153),plt.imshow(img2, 'gray'), plt.title('噪声幅值图', fontsize=10, fontproperties=font), plt.axis('off')  # 噪声幅值图
    plt.subplot(154),plt.imshow(img4, 'gray'), plt.title('滤波后幅值图', fontsize=10, fontproperties=font), plt.axis('off')  # 滤波后幅值图
    plt.subplot(155),plt.imshow(img3, 'gray'), plt.title('高斯滤波图', fontsize=10, fontproperties=font), plt.axis('off')  # 高斯滤波图
    plt.show()
```

(3)调试（编码实现过程中遇到的错误，及调试解决等内容）

问题 1：在调试代码的过程中，我遇到了如下图 13 所示的报错。

```
Traceback (most recent call last):
  File "D:/2020185 _and_10208/Kernel_lessons/数字图像处理/实验3/高斯滤波.py", line 36, in <module>
    plt.subplot(152),plt.imshow(img1[:,:,0], 'gray'), plt.title('噪声图', fontsize=10, fontpropertie
IndexError: too many indices for array: array is 2-dimensional, but 3 were indexed

Process finished with exit code 1
```

图 13 报错 1 截图

解决 1：报错显示我的数据是 2 维，但是我需要提取的是三维的，经过查验整个代码，我发现这个数据是已经是灰度图了，灰度图只有两个维度，三位提取的方法对他提取的时候会报图 13 的错误。因此我修改了代码，将[:,:,0]去掉后正常运行。修改的结果如下图 14 所示。

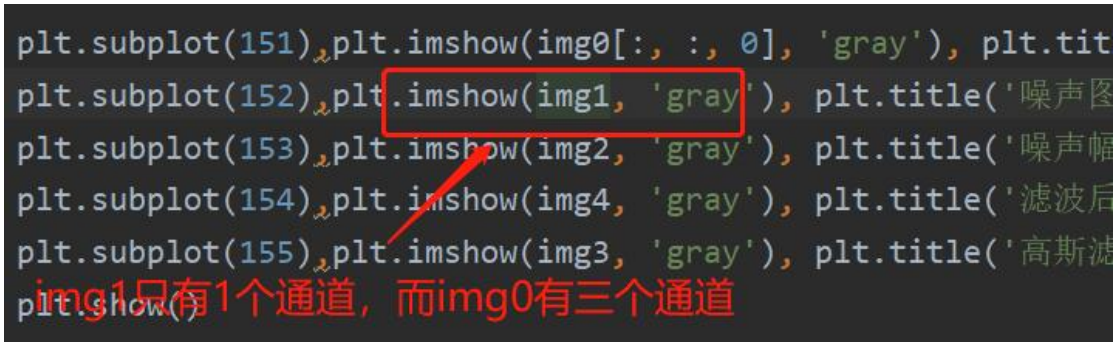

图 14 报错 1 解决方案

(4)实验结果（实验运行结果截图等体现实验结果的内容）

结果 1：给图像加上高斯噪声的结果如下图 15 所示。

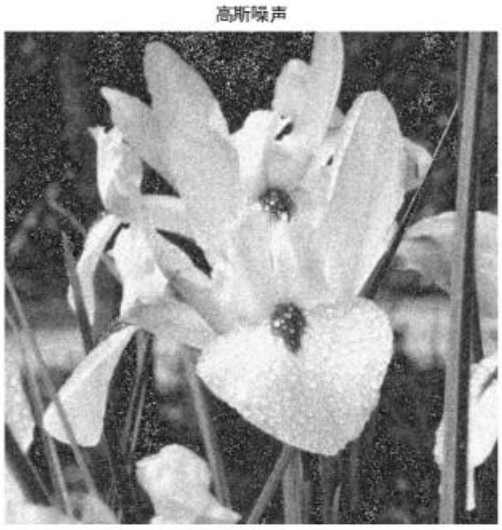

图 15 添加高斯噪声

结果 2：给加噪声后的图片做傅里叶变换后，提取加噪后图像的幅值谱如下图 16 所示。

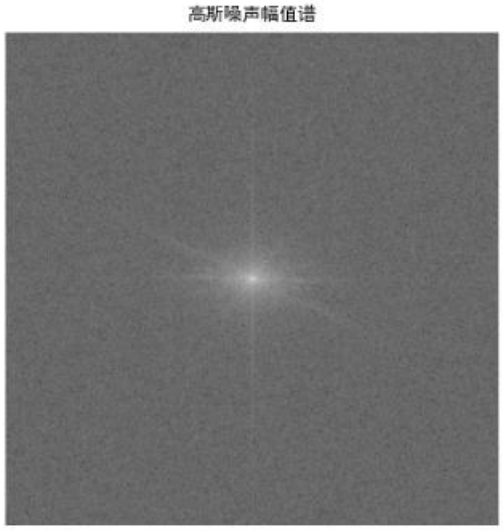

图 16 加高斯噪声图幅值谱

结果 3：对加高斯噪声后的图像进行低通滤波，滤波后得到的幅值谱图像如下图 17 所示。

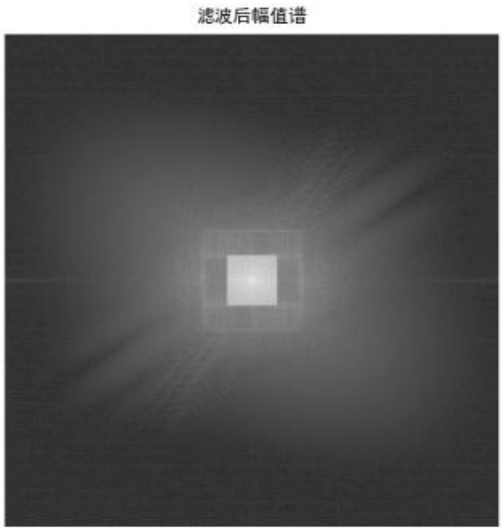

图 17 低通滤波得到的幅值谱

结果 4：由结果 3 得到的幅值谱经过逆傅里叶变换后得到的重构图像如下图 18 所示。

图 18 低通滤波得到的重构图像

结果 5：对加高斯噪声后的图像进行高斯滤波，滤波后得到的幅值谱图像如下图 19 所示。

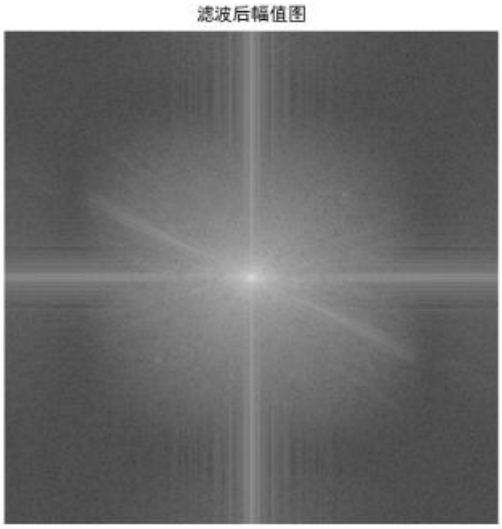

图 19 高斯滤波得到的幅值谱

结果 6：由结果 5 得到的幅值谱经过逆傅里叶变换后得到的重构图像如下图 20 所示。

图 20 高斯滤波得到的重构图像

(5)分析与总结（与本次实验相关的分析与总结）

本实验主要进行的是理想低通滤波对含噪图像的处理。所谓理想低通滤波就是滤掉高频部分，仅允许低频通过，以去掉噪声，使图像得到平滑。其使用的方法也很简单——按照模板设计滤波核，然后用滤波核和经傅里叶变换后的图像进行矩阵运算，得到的图像经过傅里叶逆变换得到重构图。其模板如下：

$$H(u,v)=\begin{cases}1, D(u,v)\le D_0\\ 0, D(u,v)\ge D_0\end{cases}$$

除要求的实验之外，我还设置了一组对比组：利用高斯滤波对含高斯噪声的图像进行低通滤波。得到的效果如图所示。从对比中可以很容易看出，经高斯滤波得到的图像要比理性低通滤波得到的图像更鲜艳，这是因为在高斯低通滤波中

不会产生“振铃”现象。所谓“振铃”现象，就是输出图像的灰度剧烈变化处产生的震荡，就好像钟被敲击后产生的空气震荡。回归实质就是高斯低通滤波的滤波核分布并不像理想低通滤波那样只有 0 和 1 这样的值，因此经它重构的图像更加真实。

对本次实验的总结

在数字图像处理中，有两个极重要处理方式：空域和频域。空域优点：直观，直接操作图像像素值。频域优势：频域在空域的基础上抽象了一层，采用“频率”来代表像素之间的变化和波动，这种方式更加全局。傅里叶变换的核心思想就是试图寻找一组正交基来表示原函数，以实现从空域到频域的转换。

在 python 中，若要实现傅里叶变换，一般有两种途径，分别是利用 numpy 工具包和 OpenCV 工具包实现。利用 numpy 实现傅里叶变换的步骤为：利用 numpy.fft.fft2()函数对图像进行傅里叶变换，返回一个复数数组；之后经过 numpy.fft.fftshift()方法对图像进行处理将零频率的分量移动到频谱中心；在这之后利用 20*np.log(np.abs(fshift))方法将附属映射到[0,255]之间，事实上这里得到的就是幅值谱。最后利用 np.fft.ifft2(ishift)方法进行逆傅里叶变换能够得到重构的图像。利用 OpenCV 实现傅里叶变换的步骤和上述类似，只是调用的类库不同，其步骤为：利用 cv2.dif()实现傅里叶变换，返回一个复数数组；之后经过 numpy.fft.fftshift()方法对图像进行处理将零频率的分量移动到频谱中心；再经过 cv2.magnitude()返回每个复数的大小，最后通过 cv2.idft()进行逆傅里叶变换得到重构图像。

频域滤波的核心思想是：可以通过选择合适的频率传递函数突出图像的某一方面的特征，从而得到需要的图像。在实际操作中的步骤为：

1. 大小为 M*N 的输入图像 f(x,y)，得到填充参数 P-2M,Q=2N
2. 形成大小为 P*N 的图像 f’(x,y)
3. 用(-1)^(x+y)乘 2 中结果得到变换的中心。
4. 计算 3 中的图像 DFT，得到 F(u,v)
5. 滤波函数 H(u,v)与 F(u,v)相乘
6. 对 5 得出的结果进行 IDFT，并选择其中的实部
7. 从 6 中得出的做上线提取 M*N 区域，得到最终处理的图像。