数字图像处理上机作业

19200300004 黄铭瑞

1.说出最感兴趣的图像处理应用及涉及的课程章节和知识点

答：

图像修复现已是计算机图形学和计算机视觉中的一个研究热点，在文物保护、影视特技制作、虚拟现实、多余物体剔除(如视频图像中删除部分人物、文字、小标题等)等方面有着重大的应用价值。识别修复区域涉及了：

**图像边缘检测**。检出边缘信息，识别到修复区域并根据相邻像素值进行扩张达到弥补像素值修复图片的效果。

**图像分割**。分割出图像里的污点，裂痕。

**图像纹理分析**。有时为了除去大块污斑，需要使用纹理合成的方法。

**图像增强**。对褪色的老照片进行图像增强。

2. 产生右图所示亮块图像 f1(x,y)（128×128大小，暗处=0，亮处=255），对其进行FFT：

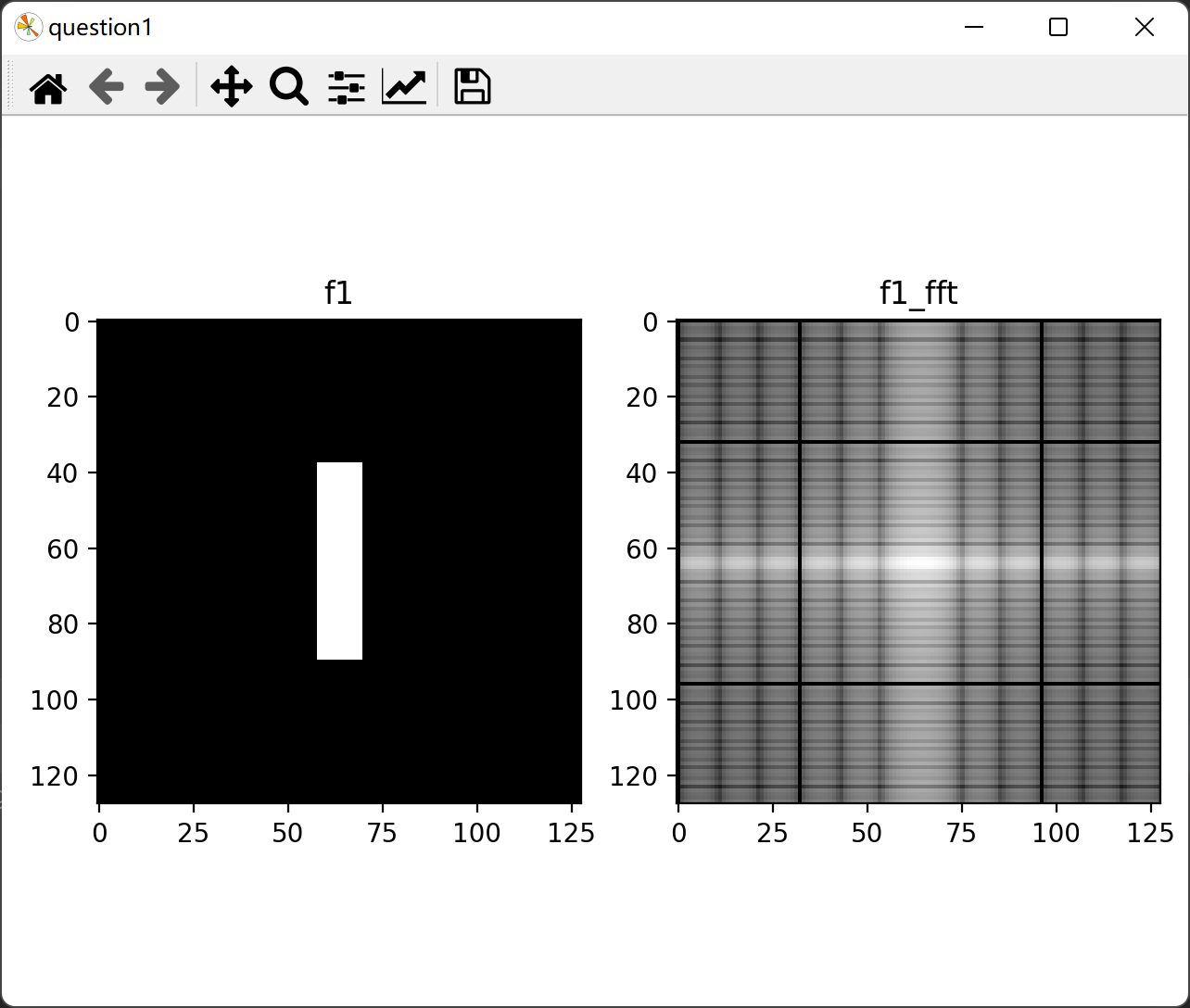
（1）同屏显示原图f1和FFT(f1)的幅度谱图；

（2）若令f2(x,y)=（-1）x+y f1(x,y)，重复以上过程，比较二者幅度谱的异同，简述理由；

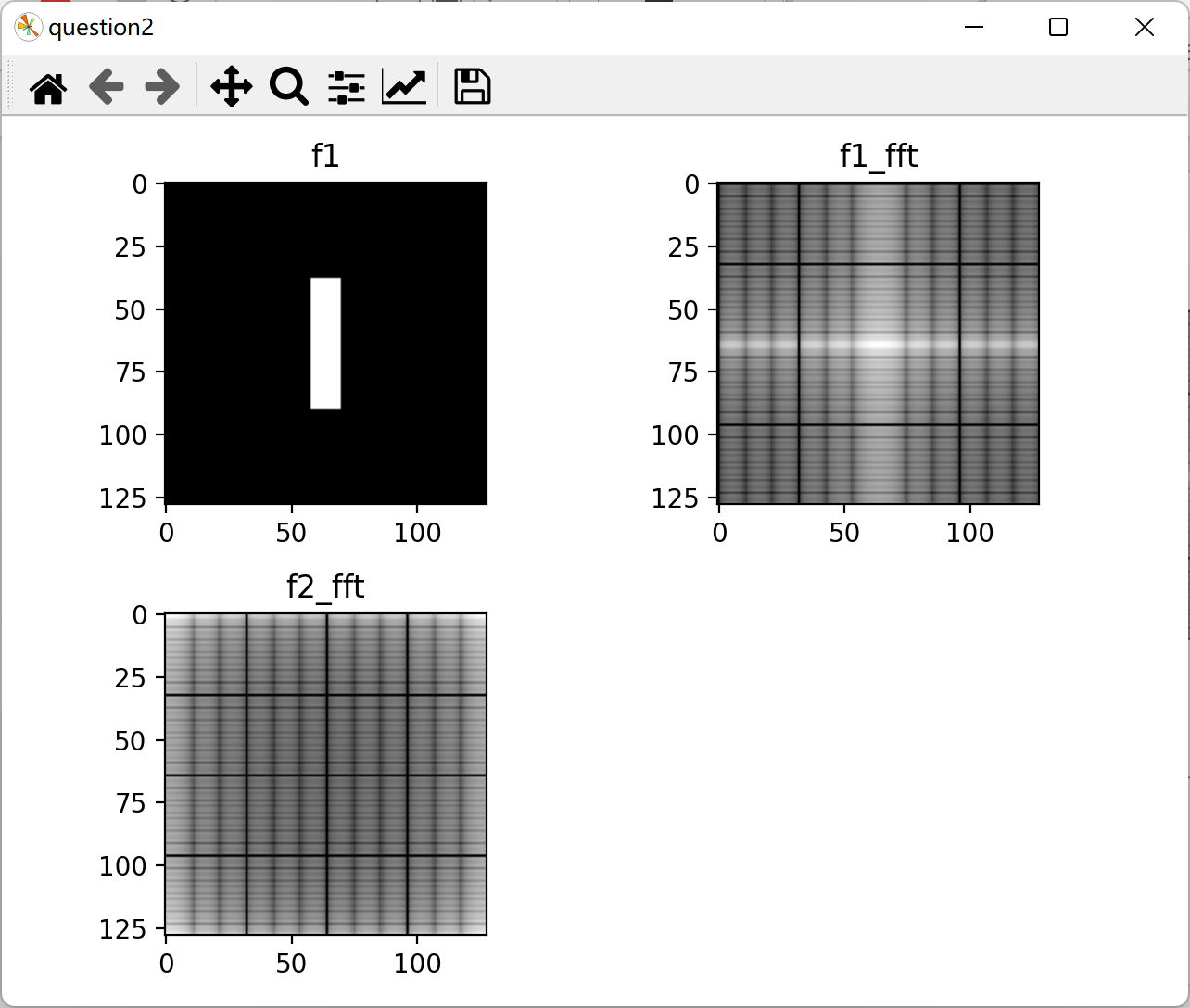
（3）若将f2(x,y)顺时针旋转45度得到f3(x,y)，试显示FFT(f3)的幅度谱，并与FFT(f2)的幅度谱进行比较。

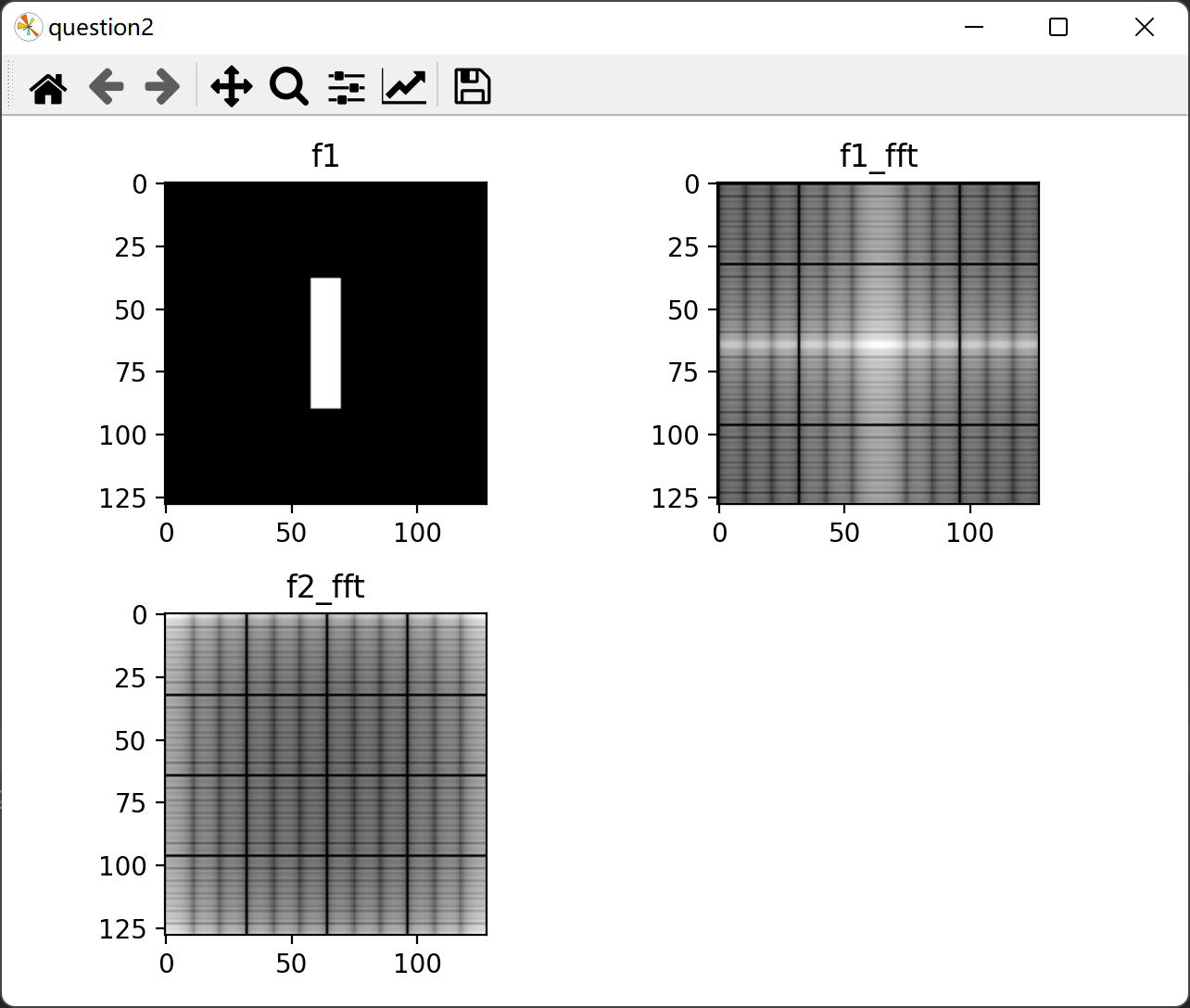
答：使用numpy.fft.fft2()对矩阵进行fft，之后原点移动，最后对数变换，绘出图像。

（1）



（2）

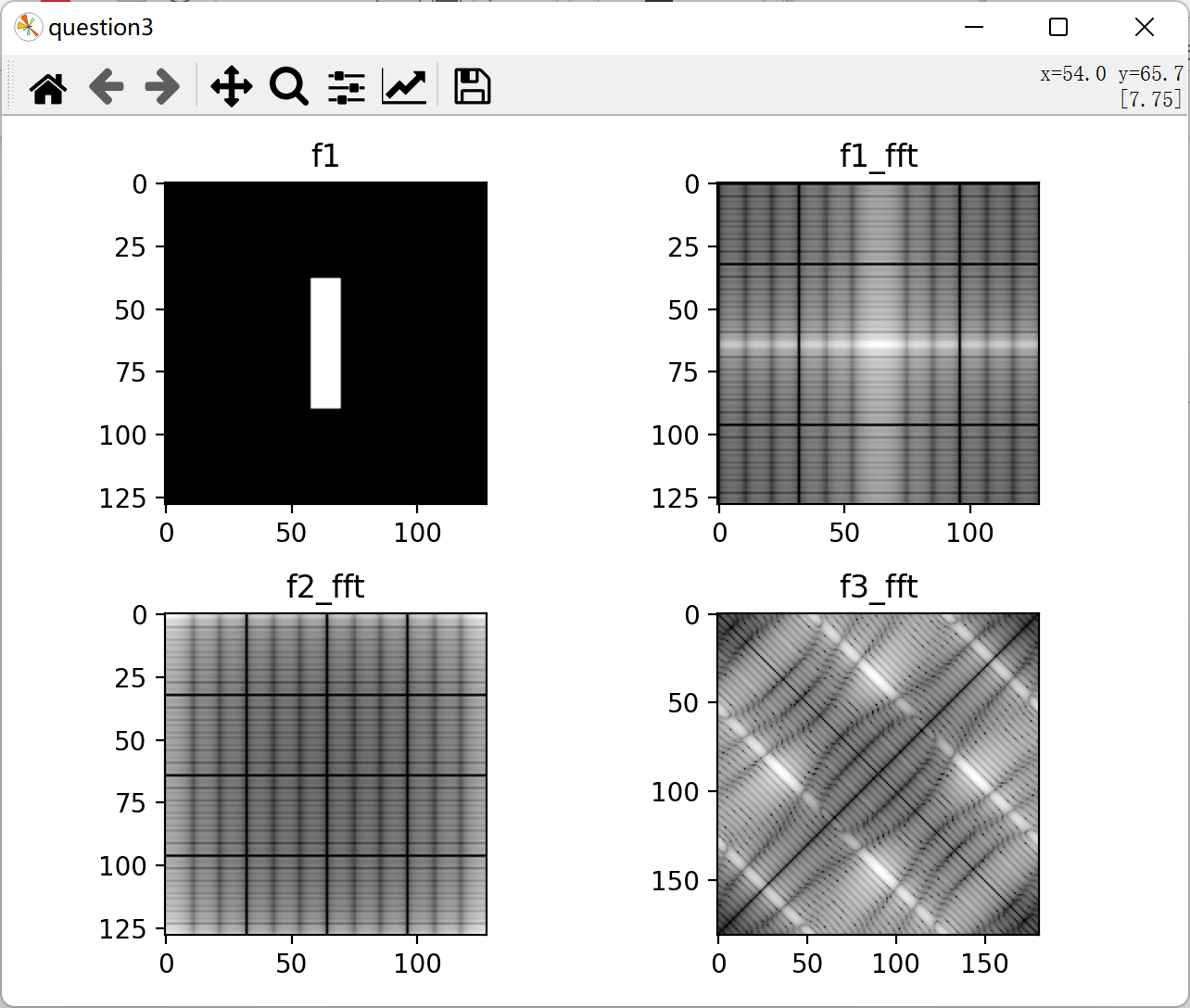






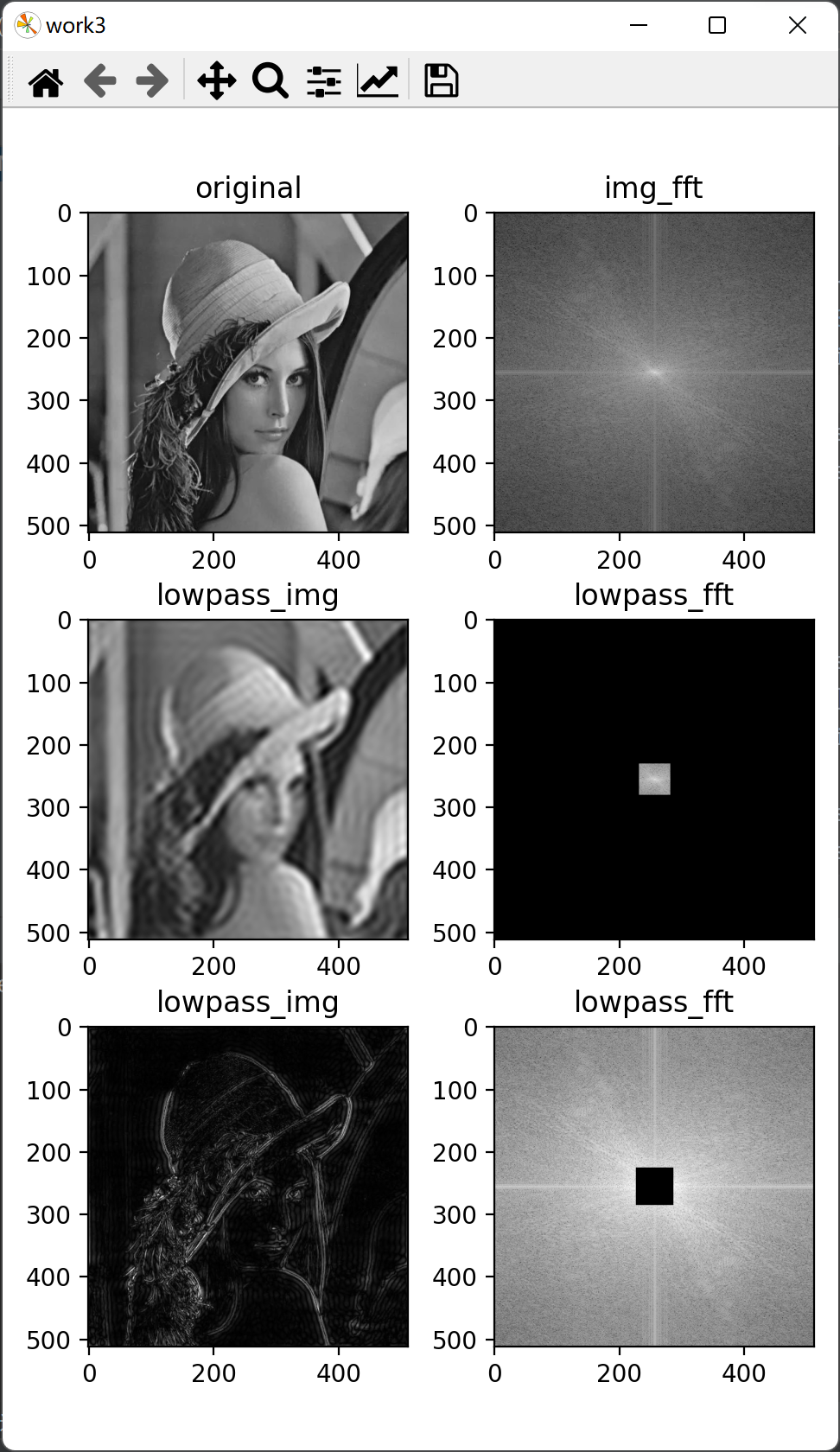
异：FFT(f2)的频谱图相当于f1经过点四等分成4块后，重排的。移动到,移动到,移动到, 移动到位置。

同：频谱的幅度没变化

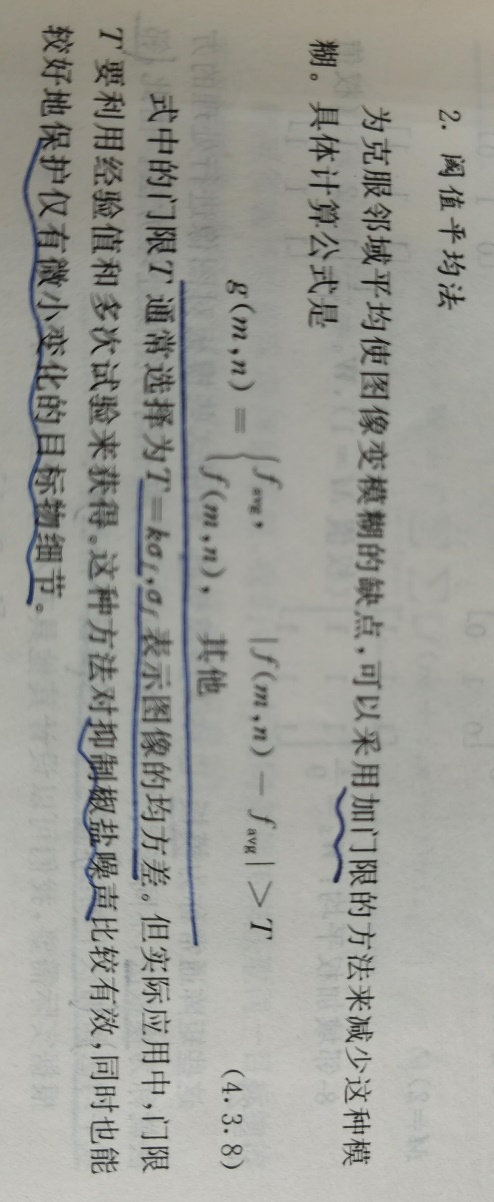


3.对512×512大小、256级灰度的数字图像lena进行频域的理想低通、高通滤波，同屏显示原图、幅度谱图和低通、高通滤波的结果图。

答：生成低通掩膜和高通掩膜，使用掩膜过滤掉对应的频率分量。再经过傅里叶逆变换恢复图片。



4.对图像pollen进行如下处理：

（1）使用直方图均衡化及直方图规定化，展示对比度拉伸的效果，要求先转化为灰度级L=8，再按要求转化，写出详细的过程，其中规定直方图为0,0,0,0.15,0.20,0.30,0.20,0.15

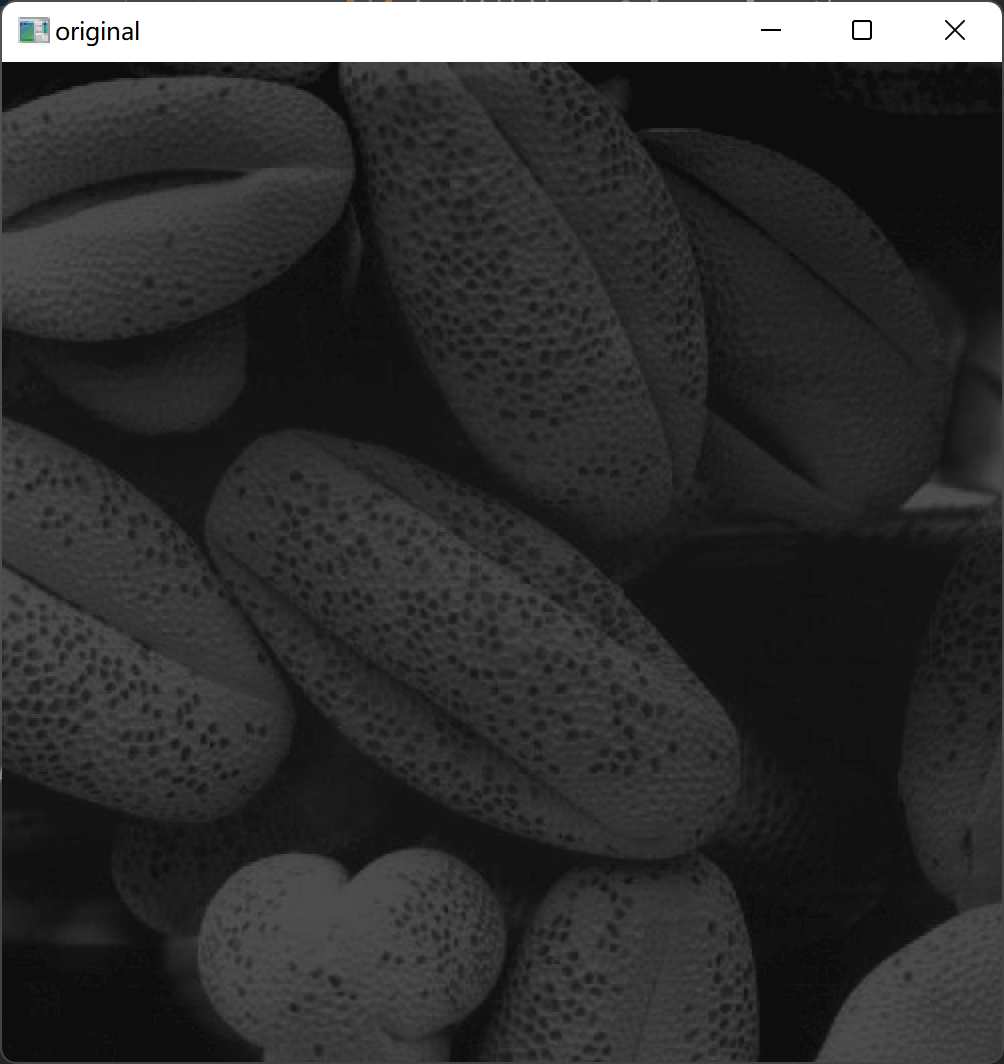
（2）对直方图均衡化后的图像加入高斯噪声，用4-邻域平均法和中值滤波平滑加噪声图像（图像四周边界不处理，下同），同屏显示原图像、加噪图像和处理后的图像。

① 不加门限；

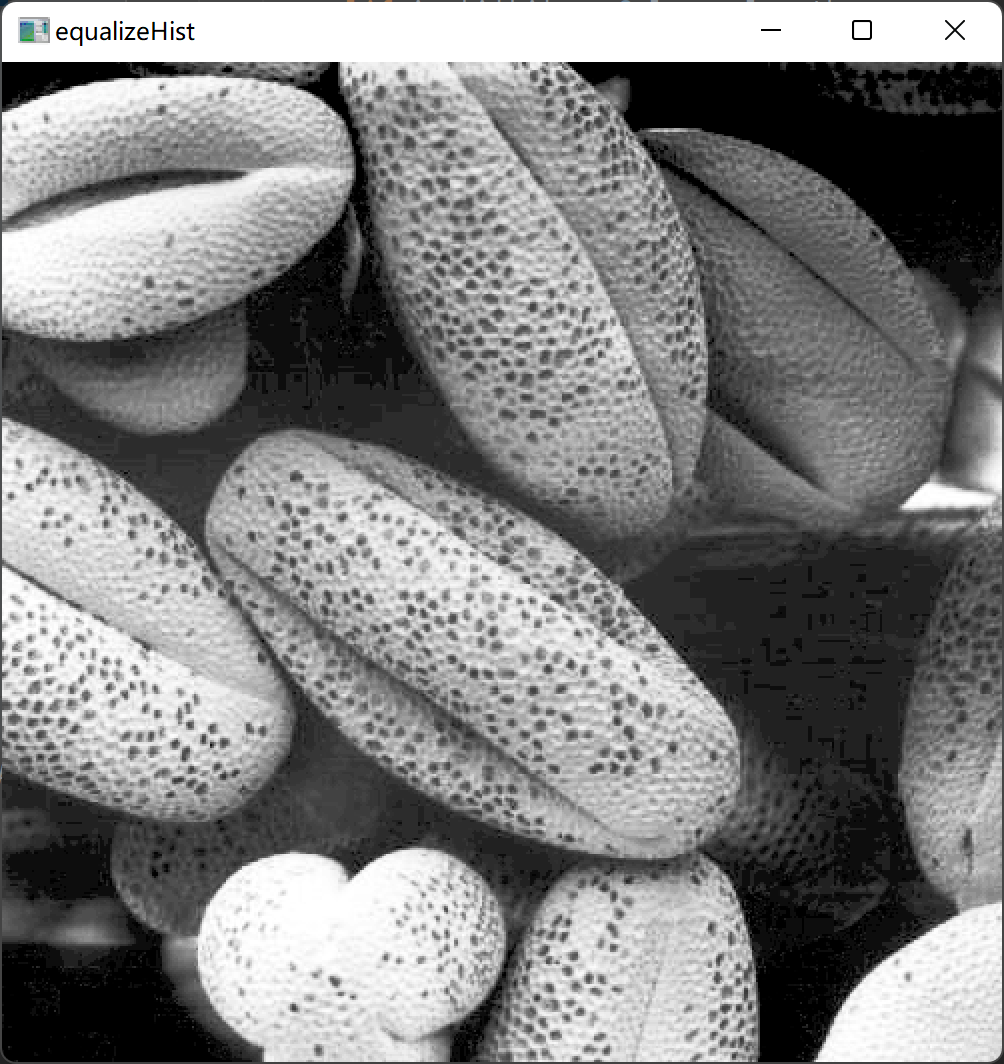
② 加门限 ，（其中）

答：（1）cv2.imread()读取pollen.tif图片，转为灰度图，使用cv2.equalizeHist对他进行直方图均衡化，然后由各像素级的累计直方图乘以255得到该i像素级的值。

原始图片：



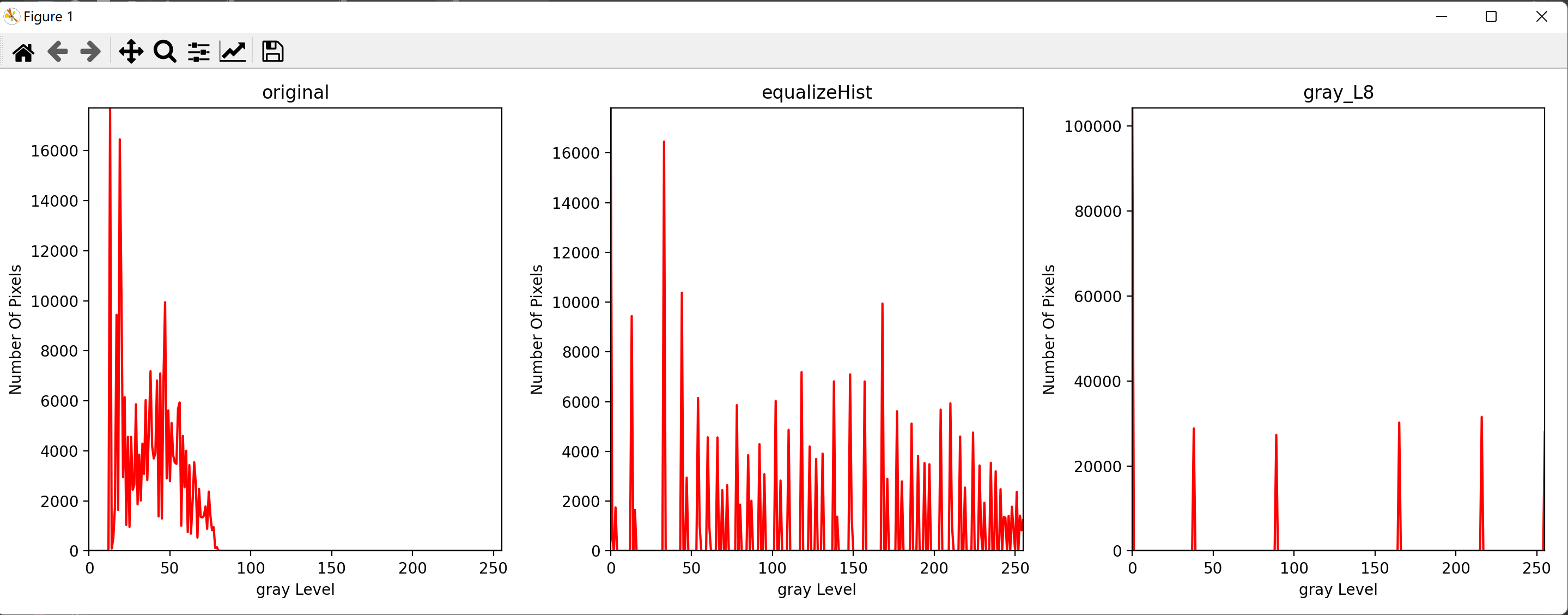
直方图均衡化后的图片：



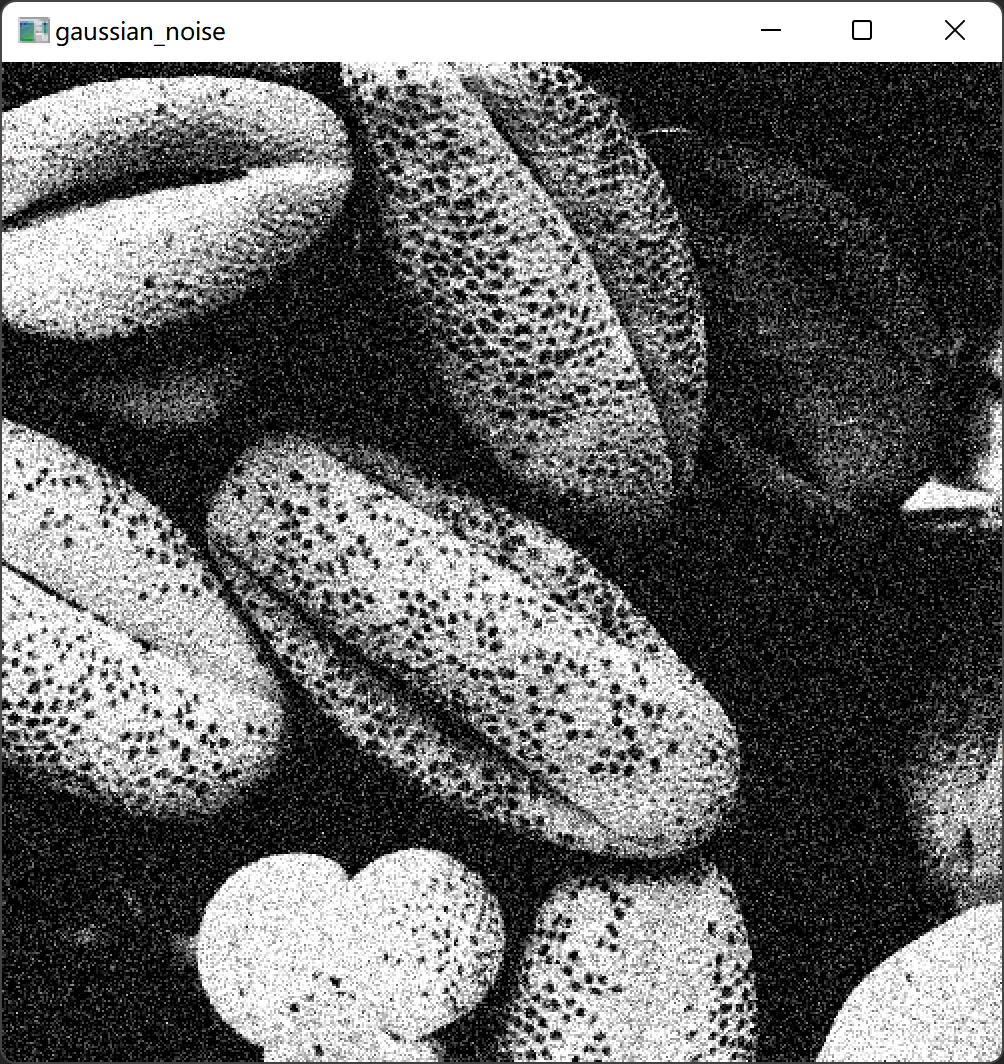
转化成L8灰度级：



对应的直方图：



（2）添加高斯噪声：



4-邻域平均法平滑处理：

在求4邻域平均值时，需要把像素值转为int类型在计算，防止溢出。



中值滤波：



5、 （1） 用Laplacian锐化算子（分和两种情况）对512×512大小、256级灰度的数字图像lena进行锐化处理，显示处理前、后图像。

（2） 若令

，





则回答如下问题：

① 、和之间有何关系？

② 代表图像中的哪些信息？

③ 由此得出图像锐化的实质是什么？

答：（1）



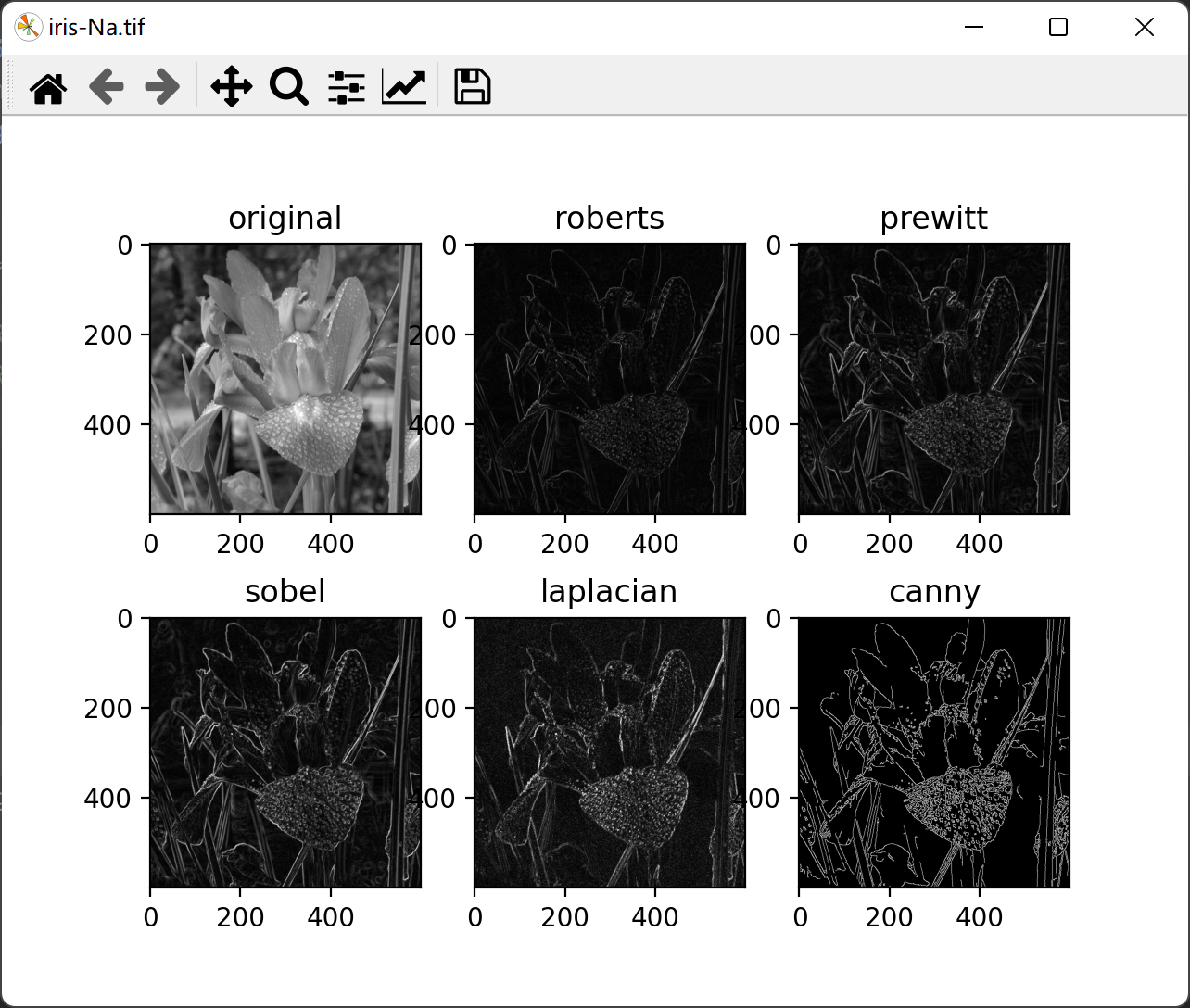
（2）①,锐化后的图像等于原始图像信息叠加边缘图像信息。

②代表图像边缘信息。

③对边缘信息的增强。

6、分别利用Roberts、Prewitt、Sobel、Laplacian、Canny边缘检测算子，对256级灰度的自然图像iris-Na、光学遥感图像bridge-RS、SAR图像road-SAR进行边缘检测，显示处理前、后图像，并阐述不同算法和数据源对结果的影响。

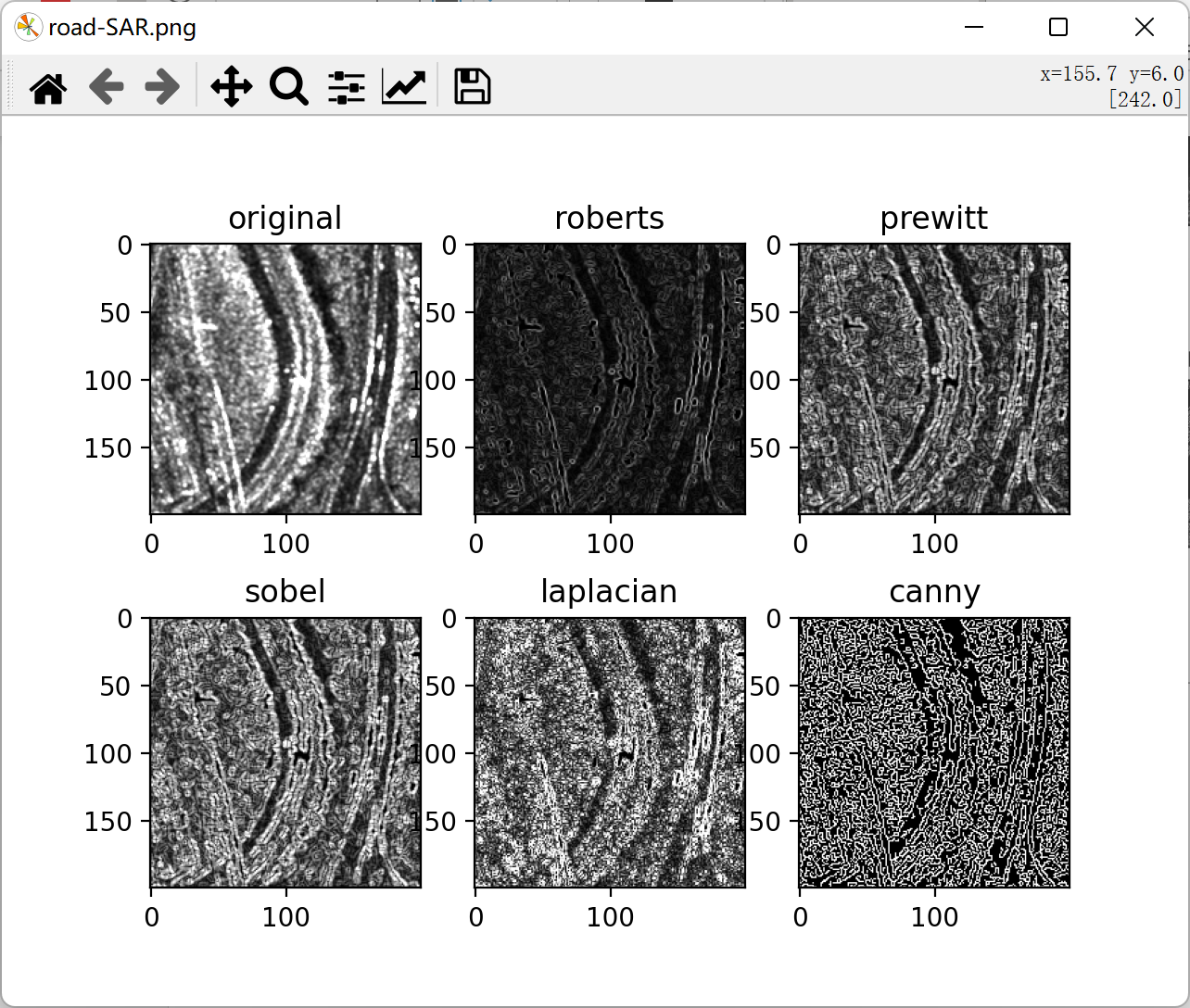
自然图像iris-Na：



光学遥感图像bridge-RS：



SAR图像road-SAR：



（2）Robert算子对这三个个原始图像的边缘信息的提取不是很好，并不是很清晰。

Prewitt算子对灰度渐变的图像边缘提取效果较好，而没有考虑相邻点的距离远近对当前像素点的影响。

Sobel算子对噪声较多的图像处理效果比较好，比如SAR图像中采用sobel算子相比其他算子是最清晰的。

Laplacian 算子易丢失边缘信息，如iris图像使用laplacian算子，上半部分边缘信息变淡。

Canny算子边缘检测是最强的，甚至一些人眼难以观察的边缘也能检测出来，比如iris花的花瓣上，bridge-RS图像的岸上纹理。

7、学习数字图像处理课程的心得体会，该课程在哪些方面需要改进，对该课程或者任课老师有哪些意见或建议。

答：学习了数字图像处理这门课，我学会了图像增强，边缘检测和图像分割的方法，这对之后的图像数据处理大有作用，同时也建议老师在课堂上多举一一些具体的例子，或者现场演示，因为有的内容真的很抽象。

附录

第二题.py

import numpy as np  
np.set\_printoptions(threshold=np.inf)  
from scipy.ndimage import rotate  
  
  
import matplotlib.pyplot as plt  
  
# 生成初始图像  
f1 = np.zeros((128, 128), dtype=int)  
for i in range(38, 90):  
 for j in range(58, 70):  
 f1[i, j] = 255  
  
f2 = np.zeros((len(f1[0]), len(f1[1])), dtype=int)  
for m in range(len(f1[0])):  
 for n in range(len(f1[1])):  
 f2[m, n] = pow(-1, (m + n)) \* f1[m, n]  
  
f3 = rotate(f2, angle=-45)  
  
# f1进行fft  
f1\_fft = np.fft.fft2(f1)  
shiftcenter1 = np.fft.fftshift(f1\_fft)  
log\_fft1 = np.log(1 + np.abs(shiftcenter1))  
  
# f2进行fft  
f2\_fft = np.fft.fft2(f2)  
shiftcenter2 = np.fft.fftshift(f2\_fft)  
log\_fft2 = np.log(1 + np.abs(shiftcenter2))  
  
# f3进行fft  
f3\_fft = np.fft.fft2(f3)  
shiftcenter3 = np.fft.fftshift(f3\_fft)  
log\_fft3 = np.log(1 + np.abs(shiftcenter3))  
  
  
  
def first\_question():  
  
 plt.figure('question1')  
 plt.subplot(121)  
 plt.title('f1')  
 plt.imshow(f1, cmap='gray')  
  
 plt.subplot(122)  
 plt.title('f1\_fft')  
 plt.imshow(log\_fft1, cmap='gray')  
  
 plt.tight\_layout()  
 plt.show()  
  
def second\_question():  
  
 plt.figure('question2')  
 plt.subplot(221)  
 plt.title('f1')  
 plt.imshow(f1, cmap='gray')  
  
 plt.subplot(222)  
 plt.title('f1\_fft')  
 plt.imshow(log\_fft1, cmap='gray')  
  
 plt.subplot(223)  
 plt.title('f2\_fft')  
 plt.imshow(log\_fft2, cmap='gray')  
  
 plt.tight\_layout()  
 plt.show()  
  
def third\_question():  
  
 plt.figure('question3')  
 plt.subplot(221)  
 plt.title('f1')  
 plt.imshow(f1, cmap='gray')  
  
 plt.subplot(222)  
 plt.title('f1\_fft')  
 plt.imshow(log\_fft1, cmap='gray')  
  
 plt.subplot(223)  
 plt.title('f2\_fft')  
 plt.imshow(log\_fft2, cmap='gray')  
  
 plt.subplot(224)  
 plt.title('f3\_fft')  
 plt.imshow(log\_fft3, cmap='gray')  
  
 plt.tight\_layout()  
 plt.show()  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 first\_question()  
 second\_question()  
 third\_question()

第三题.py

import cv2  
import numpy as np  
import matplotlib.pyplot as plt  
  
  
# 制作低通掩膜  
def make\_lowpass\_mask(image, mask\_cols=40, mask\_rows=40):  
 mask = np.zeros((len(image[1]), len(image[0])), dtype=int)  
 center = (int(len(image[1]) / 2), int(len(image[0]) / 2))  
 mask[  
 center[0] - mask\_cols // 2: center[0] + mask\_cols // 2,  
 center[1] - mask\_rows // 2: center[1] + mask\_rows // 2] = 1  
 return mask  
  
  
# 制作高通掩膜  
def make\_highpass\_mask(image, mask\_cols=40, mask\_rows=40):  
 mask = np.ones((len(image[1]), len(image[0])), dtype=int)  
 center = (int(len(image[1]) / 2), int(len(image[0]) / 2))  
 mask[  
 center[0] - mask\_cols // 2: center[0] + mask\_cols // 2,  
 center[1] - mask\_rows // 2: center[1] + mask\_rows // 2] = 0  
 return mask  
  
  
# fft，中心化  
def fshift\_fft(image):  
 f = np.fft.fft2(image)  
 fshift = np.fft.fftshift(f)  
 return fshift  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 # 读取lena图像，fft  
 img = cv2.imread('lena.jpg', 0)  
 img\_fshift = fshift\_fft(img)  
 img\_fshift\_fft = np.log(np.abs(img\_fshift))  
  
 # 低通处理  
 lowpass\_mask = make\_lowpass\_mask(img, 50, 50)  
 lowpass\_fshift = lowpass\_mask \* img\_fshift  
 lowpass\_fshift\_fft = np.log(1 + np.abs(lowpass\_fshift))  
 img\_lowpass = np.fft.ifftshift(lowpass\_fshift)  
 img\_lowpass = np.fft.ifft2(img\_lowpass)  
 img\_lowpass = np.abs(img\_lowpass)  
  
 # 高通处理  
 highpass\_mask = make\_highpass\_mask(img, 60, 60)  
 highpass\_fshift = highpass\_mask \* img\_fshift  
 highpass\_fshift\_fft = np.log(1 + np.abs(highpass\_fshift))  
 img\_highpass = np.fft.ifftshift(highpass\_fshift)  
 img\_highpass = np.fft.ifft2(img\_highpass)  
 img\_highpass = np.abs(img\_highpass)  
  
 # 绘制  
 plt.figure('work3', figsize=(5, 15))  
 plt.subplot(321), plt.imshow(img, 'gray'), plt.title('original')  
 plt.subplot(322), plt.imshow(img\_fshift\_fft, 'gray'), plt.title('img\_fft')  
 plt.subplot(323), plt.imshow(img\_lowpass, 'gray'), plt.title('lowpass\_img')  
 plt.subplot(324), plt.imshow(lowpass\_fshift\_fft, 'gray'), plt.title('lowpass\_fft')  
 plt.subplot(325), plt.imshow(img\_highpass, 'gray'), plt.title('lowpass\_img')  
 plt.subplot(326), plt.imshow(highpass\_fshift\_fft, 'gray'), plt.title('lowpass\_fft')  
  
 plt.tight\_layout()  
 plt.show()

第四题.py

import cv2  
import numpy as np  
import matplotlib.pyplot as plt  
  
  
def plot\_grayHist(grayHist,title):  
 plt.plot(range(256), grayHist, 'r', linewidth=1.5, c='red')  
 y\_maxValue = np.max(grayHist)  
 plt.axis([0, 255, 0, y\_maxValue])  
 plt.title(title)  
 plt.xlabel("gray Level")  
 plt.ylabel("Number Of Pixels")  
  
  
def gray\_l8(img):  
 gray\_list = [0, 0, 0, 0.15, 0.35, 0.65, 0.85, 1]  
 layer\_length = 256 // 8  
 h, w = img.shape[0], img.shape[1]  
 for i in range(h):  
 for j in range(w):  
 if img[i][j] <= layer\_length:  
 img[i][j] = 255 \* gray\_list[0]  
 elif img[i][j] <= 2\*layer\_length:  
 img[i][j] = 255 \* gray\_list[1]  
 elif img[i][j] <= 3\*layer\_length:  
 img[i][j] = 255 \* gray\_list[2]  
 elif img[i][j] <= 4\*layer\_length:  
 img[i][j] = 255 \* gray\_list[3]  
 elif img[i][j] <= 5\*layer\_length:  
 img[i][j] = 255 \* gray\_list[4]  
 elif img[i][j] <= 6\*layer\_length:  
 img[i][j] = 255 \* gray\_list[5]  
 elif img[i][j] <= 7\*layer\_length:  
 img[i][j] = 255 \* gray\_list[6]  
 elif img[i][j] <= 8\*layer\_length:  
 img[i][j] = 255 \* gray\_list[7]  
 return img  
  
  
def gaussian\_noise(img, mean, sigma):  
 # 将图片灰度标准化  
 img = img / 255  
 # 产生高斯 noise  
 noise = np.random.normal(mean, sigma, img.shape)  
 # 将噪声和图片叠加  
 gaussian\_out = img + noise  
 # 将超过 1 的置 1，低于 0 的置 0  
 gaussian\_out = np.clip(gaussian\_out, 0, 1)  
 # 将图片灰度范围的恢复为 0-255  
 gaussian\_out = np.uint8(gaussian\_out\*255)  
 # 将噪声范围搞为 0-255  
 # noise = np.uint8(noise\*255)  
 return gaussian\_out# 这里也会返回噪声，注意返回值  
  
def blur(img):  
 h, w = img.shape[0], img.shape[1]  
 for i in range(1, h-1, 1):  
 for j in range(1, w-1, 1):  
 img[i][j] = (int(img[i][j-1]) + int(img[i][j+1]) + int(img[i-1][j]) + int(img[i+1][j])) // 4  
 return img  
  
  
img1 = cv2.imread('./pollen.tif')  
img1 = cv2.cvtColor(img1, cv2.COLOR\_BGR2GRAY)  
cv2.imshow('original', img1)  
cv2.waitKey()  
grayHist1 = cv2.calcHist([img1], [0], None, [256], [0, 256])  
  
# 直方图均衡化  
img2 = cv2.equalizeHist(img1)  
cv2.imshow('equalizeHist', img2)  
cv2.waitKey()  
grayHist2 = cv2.calcHist([img2], [0], None, [256], [0, 256])  
  
# L8灰度级  
img3 = gray\_l8(img2)  
cv2.imshow('gray\_L8', img3)  
cv2.waitKey()  
grayHist3 = cv2.calcHist([img3], [0], None, [256], [0, 256])  
  
# 添加高斯噪声  
img\_noise = gaussian\_noise(img2, 0, 0.2)  
cv2.imshow('gaussian\_noise', img\_noise)  
cv2.waitKey()  
  
# 4邻域平均法  
img\_4blur = blur(img\_noise)  
cv2.imshow('4-blur', img\_4blur)  
cv2.waitKey()  
  
# 中值滤波  
img\_medianblur = cv2.medianBlur(img\_noise, 3)  
cv2.imshow('medianblur', img\_medianblur)  
cv2.waitKey()  
  
plt.figure(figsize=(15, 5))  
plt.subplot(131)  
plot\_grayHist(grayHist1, 'original')  
plt.subplot(132)  
plot\_grayHist(grayHist2, 'equalizeHist')  
plt.subplot(133)  
plot\_grayHist(grayHist3, 'gray\_L8')  
plt.tight\_layout()  
plt.show()

第五题.py

import cv2  
import numpy as np  
import matplotlib.pyplot as plt  
  
  
def laplacian\_sharpe(img, kernel\_size=3):  
 h, w = img.shape[0], img.shape[1]  
 pad = kernel\_size // 2  
 out = np.zeros((h + 2 \* pad, w + 2 \* pad), dtype=np.float)  
 out[pad: h + pad, pad: w + pad] = img.copy().astype(np.float)  
 tmp = out.copy()  
  
 K = np.array([  
 [0., 1., 0.],  
 [1., -4., 1.],  
 [0., 1., 0.]])  
  
 for i in range(h):  
 for j in range(w):  
 out[pad + i, pad + j] = (-1) \* np.sum(K \* (tmp[i:i + kernel\_size, j:j + kernel\_size])) + tmp[  
 pad + i, pad + j]  
 out = np.clip(out, 0, 255)  
 out = out[pad: pad + h, pad: pad + w].astype(np.uint8)  
 return out  
  
  
img = cv2.imread('./lena.jpg')  
img = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)  
img\_sharpe = laplacian\_sharpe(img, 3)  
  
images = np.hstack([img, img\_sharpe])  
cv2.imshow('original and laplacian\_sharpe', images)  
cv2.waitKey()

第六题.py

import cv2  
import numpy as np  
import matplotlib.pyplot as plt  
  
  
def roberts(img):  
 kernelx = np.array([[-1, 0], [0, 1]], dtype=int)  
 kernely = np.array([[0, -1], [1, 0]], dtype=int)  
 x = cv2.filter2D(img, cv2.CV\_16S, kernelx)  
 y = cv2.filter2D(img, cv2.CV\_16S, kernely)  
 # 转uint8  
 absX = cv2.convertScaleAbs(x)  
 absY = cv2.convertScaleAbs(y)  
 Roberts = cv2.addWeighted(absX, 0.5, absY, 0.5, 0)  
 return Roberts  
  
def prewitt(img):  
 kernelx = np.array([[1, 1, 1], [0, 0, 0], [-1, -1, -1]], dtype=int)  
 kernely = np.array([[-1, 0, 1], [-1, 0, 1], [-1, 0, 1]], dtype=int)  
 x = cv2.filter2D(img, cv2.CV\_16S, kernelx)  
 y = cv2.filter2D(img, cv2.CV\_16S, kernely)  
 # 转uint8  
 absX = cv2.convertScaleAbs(x)  
 absY = cv2.convertScaleAbs(y)  
 Prewitt = cv2.addWeighted(absX, 0.5, absY, 0.5, 0)  
 return Prewitt  
  
  
def sobel(img):  
 x = cv2.Sobel(img, cv2.CV\_16S, 1, 0) # 对x求一阶导  
 y = cv2.Sobel(img, cv2.CV\_16S, 0, 1) # 对y求一阶导  
 absX = cv2.convertScaleAbs(x)  
 absY = cv2.convertScaleAbs(y)  
 Sobel = cv2.addWeighted(absX, 0.5, absY, 0.5, 0)  
 return Sobel  
  
  
def laplacian(img):  
 dst = cv2.Laplacian(img, cv2.CV\_16S, ksize=3)  
 Laplacian = cv2.convertScaleAbs(dst)  
 return Laplacian  
  
  
def canny(img):  
 Canny = cv2.Canny(img, 50, 150)  
 return Canny  
  
def show(fname):  
 if fname != 'road-SAR.png':  
 img = cv2.imread(fname, cv2.COLOR\_BGR2RGB)  
 img = cv2.cvtColor(img, cv2.COLOR\_RGB2GRAY)  
 else:  
 img = cv2.imread(fname, cv2.COLOR\_BGR2GRAY)  
  
 img\_roberts = roberts(img)  
 img\_prewitt = prewitt(img)  
 img\_sobel = sobel(img)  
 img\_laplacian = laplacian(img)  
 img\_canny = canny(img)  
  
 plt.figure(num= fname)  
 plt.subplot(231)  
 plt.title('original')  
 plt.imshow(img, 'gray')  
  
 plt.subplot(232)  
 plt.title('roberts')  
 plt.imshow(img\_roberts, 'gray')  
  
 plt.subplot(233)  
 plt.title('prewitt')  
 plt.imshow(img\_prewitt, 'gray')  
  
 plt.subplot(234)  
 plt.title('sobel')  
 plt.imshow(img\_sobel, 'gray')  
  
 plt.subplot(235)  
 plt.title('laplacian')  
 plt.imshow(img\_laplacian, 'gray')  
  
 plt.subplot(236)  
 plt.title('canny')  
 plt.imshow(img\_canny, 'gray')  
 plt.show()  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 show('iris-Na.tif')  
 show('bridge-RS.jpg')  
 show('road-SAR.png')