**数字图像处理实验四实验报告**

**学号：20182131055 姓名: 侯佳耀**

1. 实验目的：

熟悉入门图像编程操作

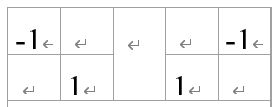
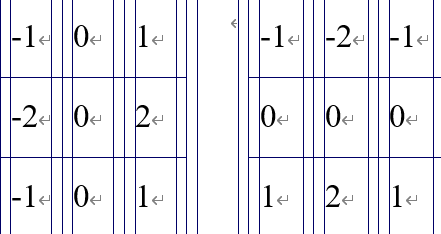
**二．实验内容：**

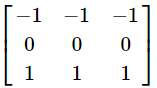
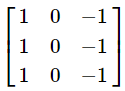
内容1： 图像锐化：分别用梯度锐化算子和拉普拉斯算子进行处理，要求 ：读取原图像；分别用两种方法进行锐化；比较锐化结果。实验后提交：原图像；两种方法锐化后的图像。

内容2：形态学图像处理：对图像进行二值化处理；读取二值化图像，形成结构元素矩阵，执行相关膨胀、腐蚀等运算并显示结果。

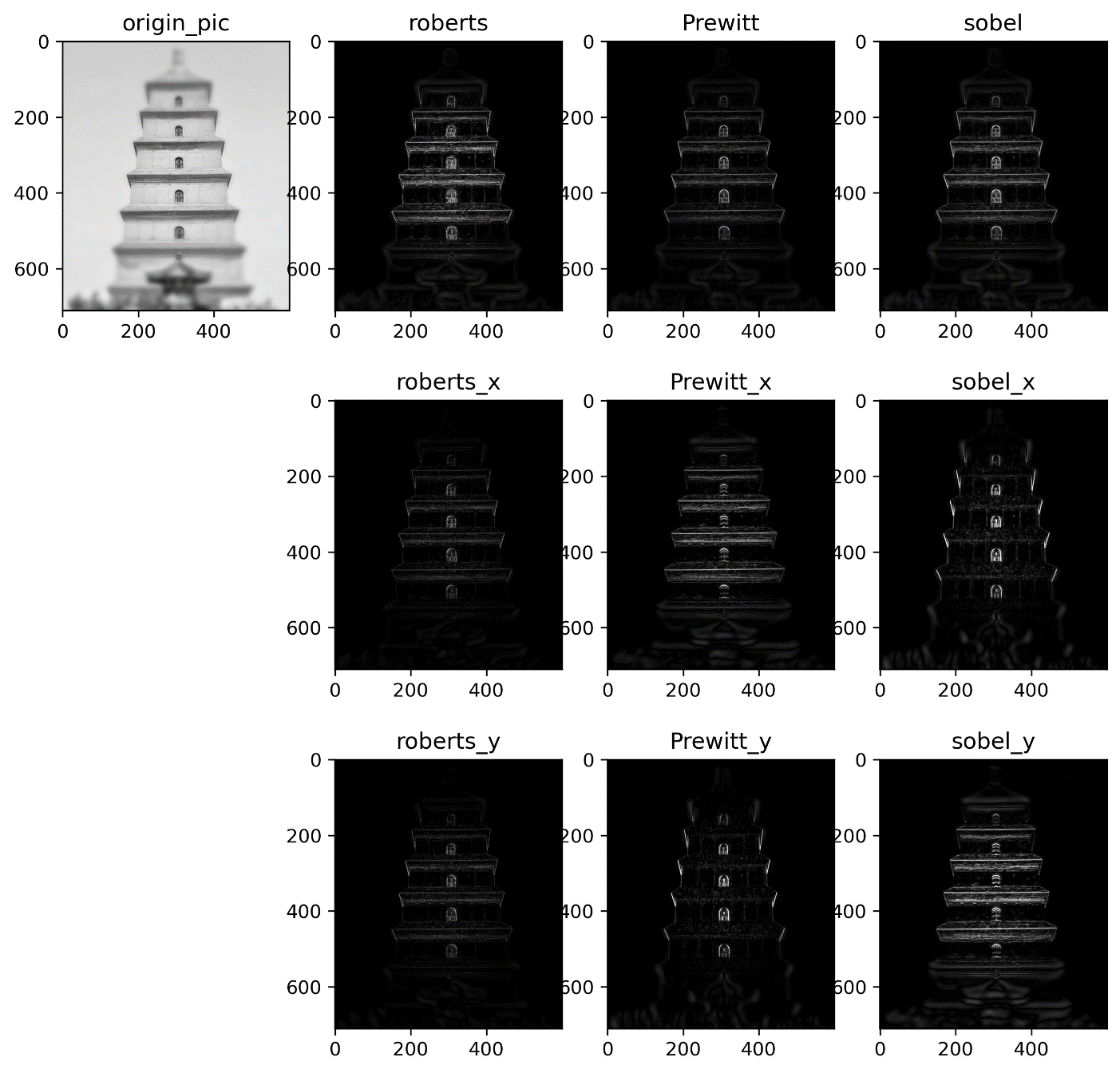
**三．实验过程：**

1、从sharpen文件夹中选取一幅图像。针对给出的图像，加入高斯和椒盐噪声。对原图像和加入噪声后的图像分别利用Roberts、Sobel、Prewit锐化算子对图像进行锐化，显示原图、算子对图像处理得到的梯度结果图像（x方向、y方向及最终梯度结果，最终梯度可以用x和y方向梯度和来简单计算）。实验结果进行分析：比较不同算子的锐化效果。其中：Roberts、Sobel、Prewitt算子如下：

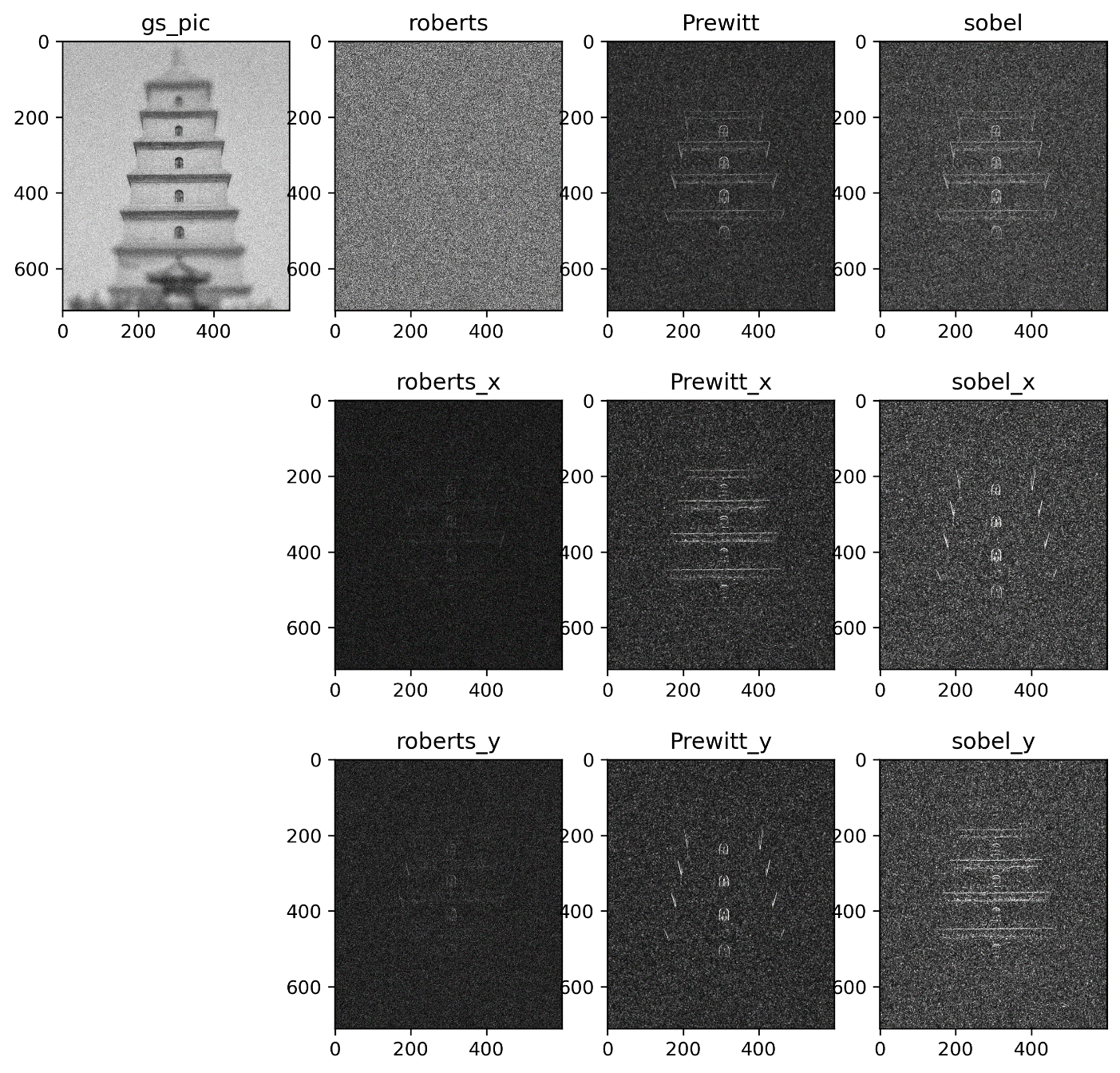
 



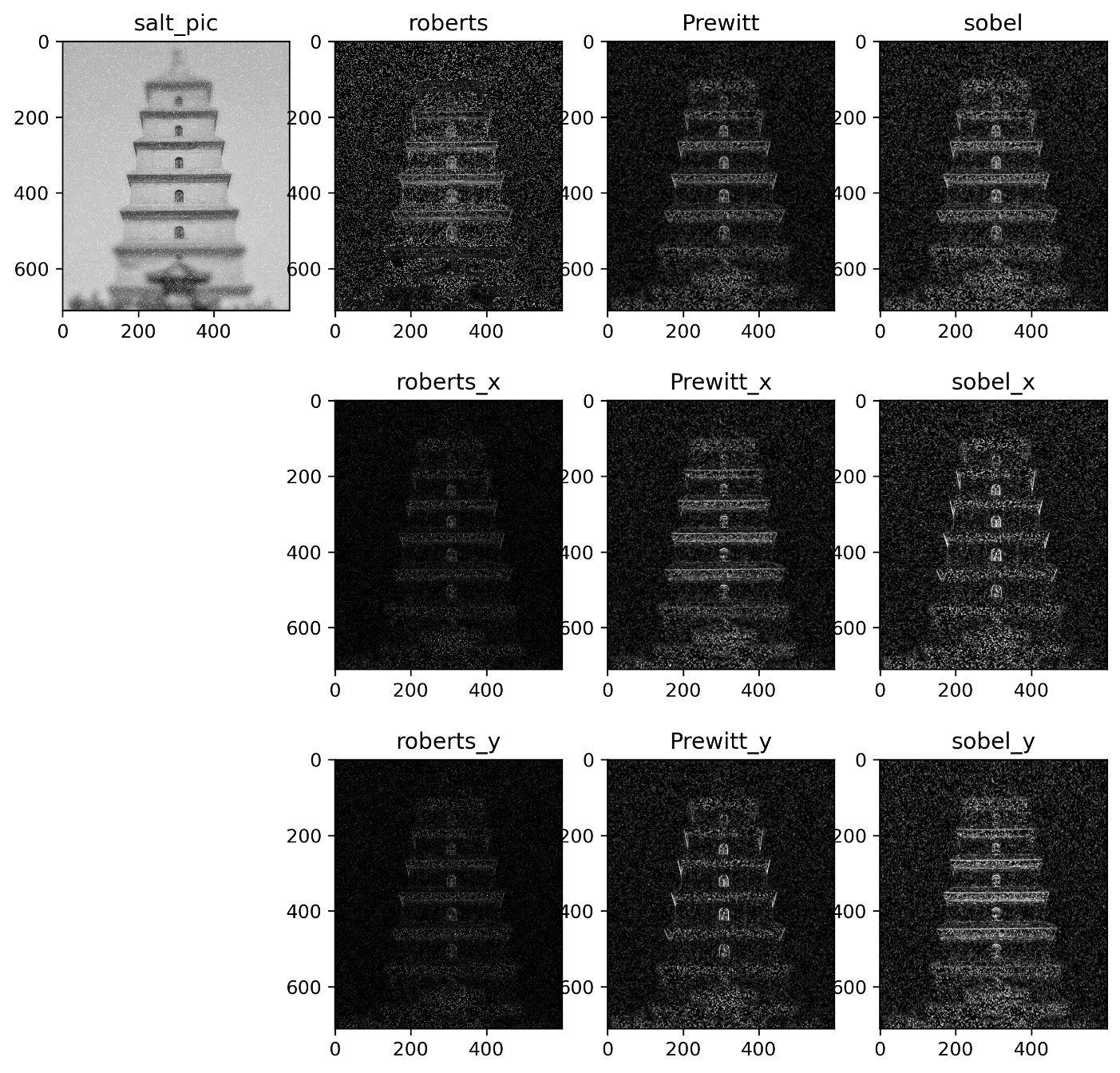
（1）原图的操作结果：



（2）高斯噪声下的处理结果：



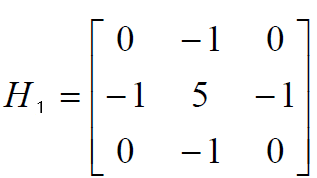
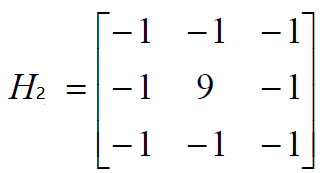
（3）椒盐噪声下的处理结果：

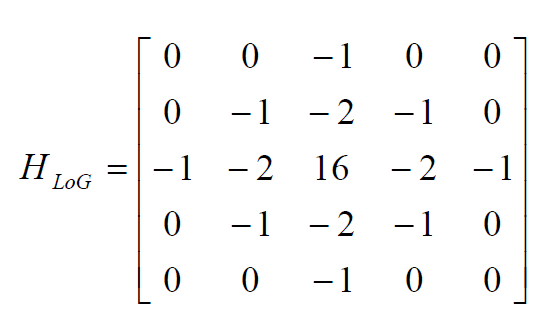


（a）Robert算子定位比较精确，但由于不包括平滑，所以对于噪声比较敏感；Prewitt算子和Sobel算子都是一阶的微分算子，而前者是平均滤波，后者是加权平均滤波且检测的图像边缘可能大于2个像素。这两者对灰度渐变低噪声的图像有较好的检测效果，但是对于混合多复杂噪声的图像，处理效果就不理想了。；

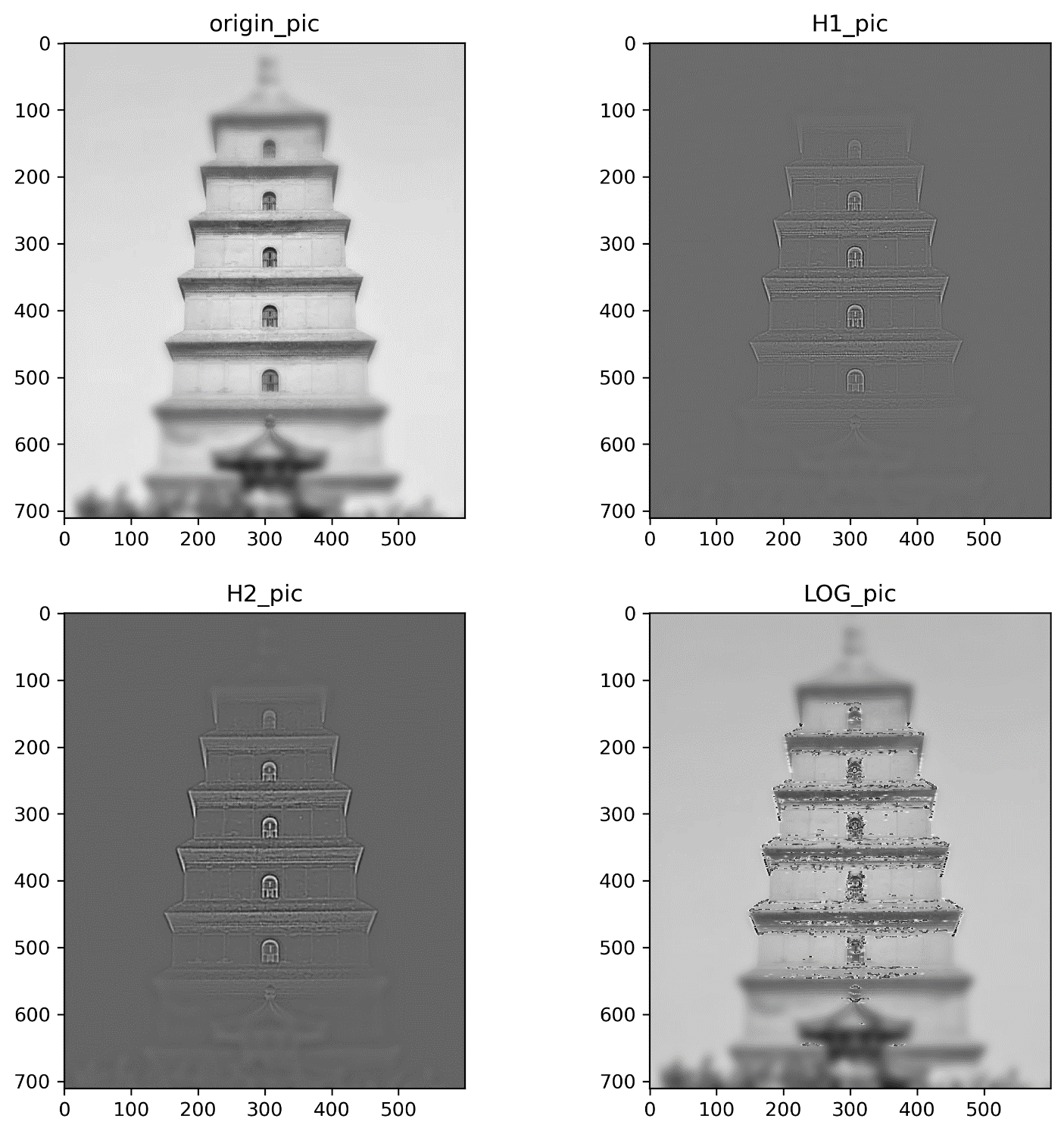
（b）梯度算子计算简单,但精度不高,只能检测出图像大致的轮廓,而对于比较细的边缘可能会忽略。Prewitt 和Sobel 算子比Roberts 效果要好一些。不同的系统,针对不同的环境条件和要求,选择合适的算子来对图像进行边缘检测。

2、从sharpen文件夹中选取一幅图像。针对给出的图像，加入高斯和椒盐噪声。对原图像和加入噪声后的图像分别利用拉普拉斯算子对图像进行锐化，显示原图、算子对图像处理得到的锐化结果。实验结果进行分析：1）比较拉普拉斯掩模 H1 和含有对角线分量的拉普拉斯掩模H2 的锐化效果；2）尝试用高斯型拉普拉斯掩模（LoG 算子）锐化图像，并观察与其他拉普拉斯掩模在抑制噪声上的差异。

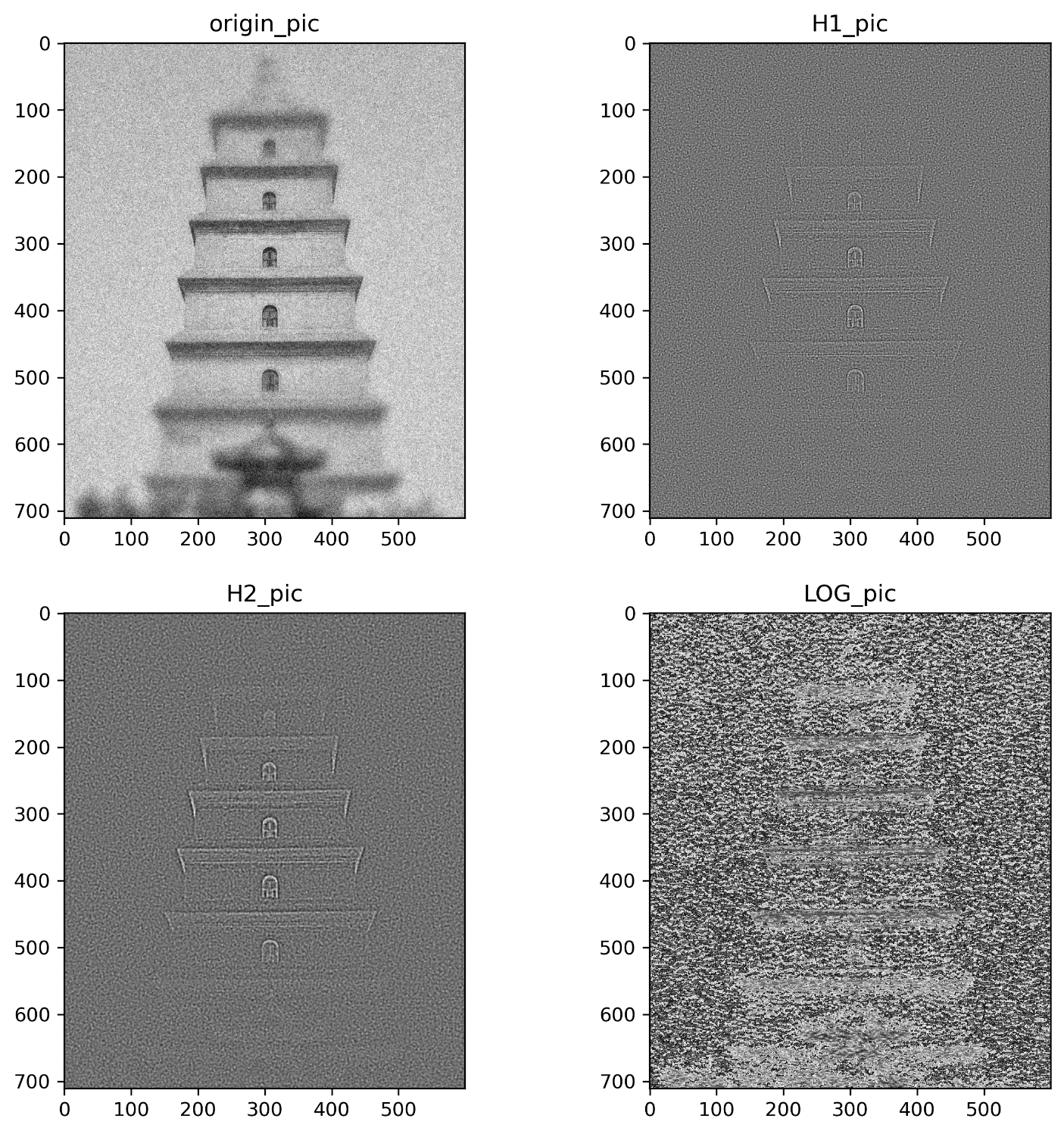
其中：拉普拉斯掩模H1如下：对角线方向也可以加入到拉普拉斯运算中，从而得到包含有对角线分量的拉普拉斯掩模H2如下：。LOG算子如下：



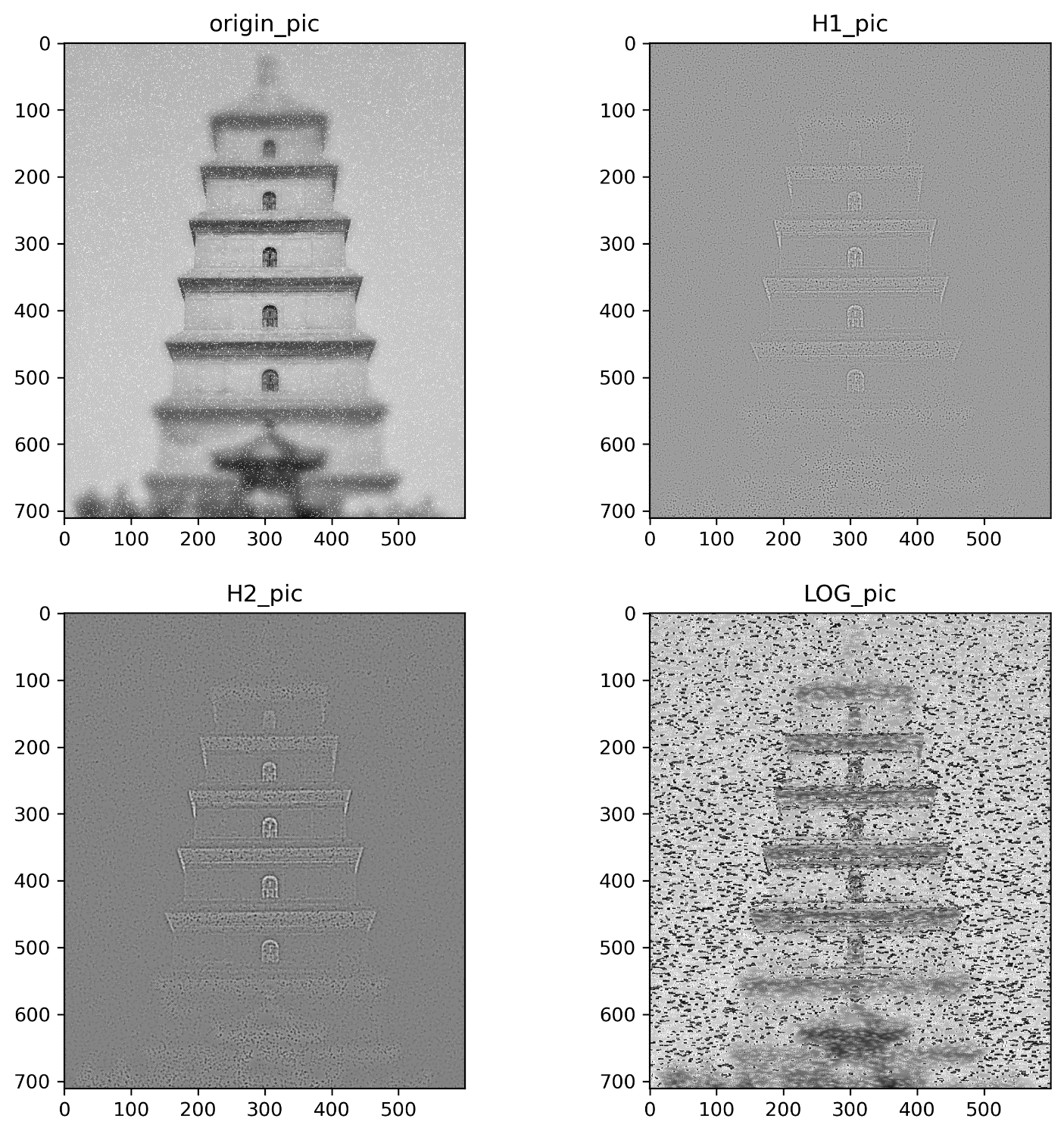
1. 原图拉普拉斯



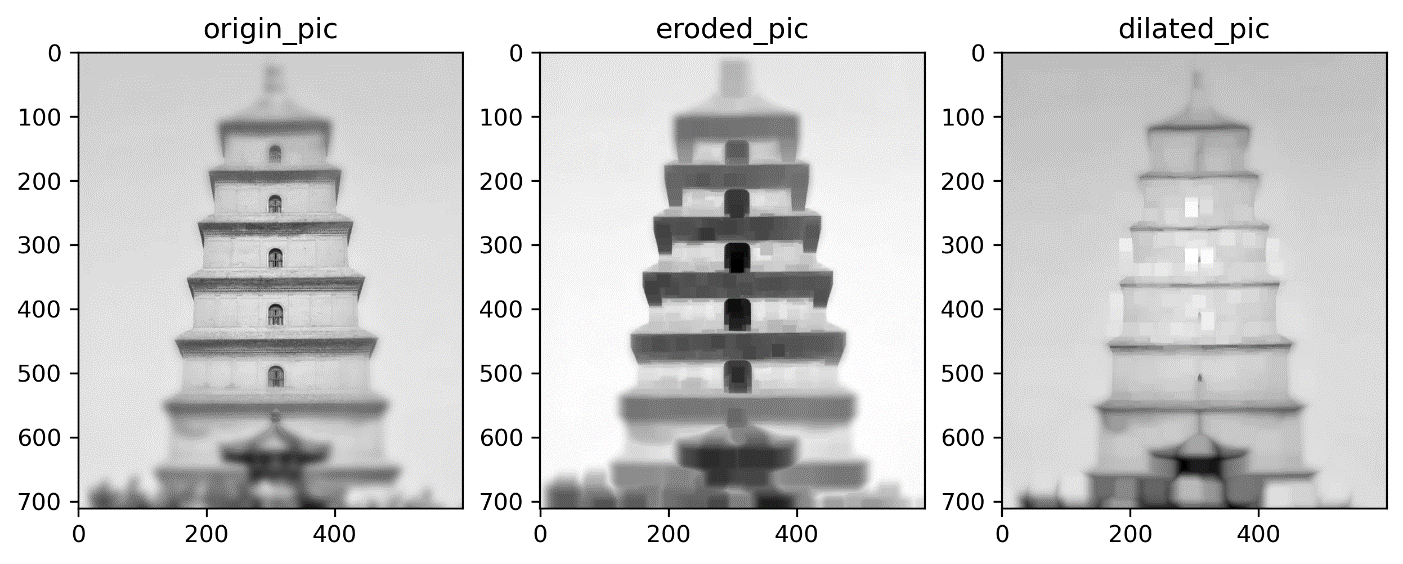
1. 高斯拉普拉斯



1. 椒盐拉普拉斯



3. 可任意选取一幅图像，运用Opencv库函数（可选），对图像进行膨胀和腐蚀操作，获得图像中物体的边缘。



**四．实验代码（实验四.py）**

# -\*- coding: utf-8 -\*-

"""

Created on Tue Dec 22 14:25:07 2020

数字图像处理实验四

@author: ASUS

"""

import cv2

#import pandas as pd

from skimage import util

from PIL import Image

import numpy as np

from matplotlib import pyplot as plt

import scipy.ndimage as nd

img=np.array(Image.open('dayanta\_mohu.bmp').convert('L'),dtype='uint8')

#b, g, r = img.split()

#img = np.array(Image.merge("RGB", (r, g, b)))

noise\_gs\_img = np.array(util.random\_noise(img,mode='gaussian')\*255,dtype='uint8')

noise\_salt\_img = np.array(util.random\_noise(img,mode='salt')\*255,dtype='uint8')

#img=noise\_salt\_img

def roberts(im):

kernelx = np.array([[-1, 0], [0, 1]],dtype=int)

kernely = np.array([[0, -1], [1, 0]],dtype=int)

x = cv2.filter2D(im, cv2.CV\_16S, kernelx)

y = cv2.filter2D(im, cv2.CV\_16S, kernely)

# 转转成uint8

absX = cv2.convertScaleAbs(x)

absY = cv2.convertScaleAbs(y)

Roberts = cv2.addWeighted(absX, 0.5, absY, 0.5, 0)

Roberts=Roberts\*5

#images = [img, Roberts]

imglist=[]

imglist.append(Roberts)

imglist.append(absX)

imglist.append(absY)

return imglist

def Prewitt(im):

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(im, cv2.CV\_16S, kernelx)

y = cv2.filter2D(im, cv2.CV\_16S, kernely)

# 转转成uint8

absX = cv2.convertScaleAbs(x)

absY = cv2.convertScaleAbs(y)

im1 = cv2.addWeighted(absX, 0.5, absY, 0.5, 0)

imglist=[]

imglist.append(im1)

imglist.append(absX)

imglist.append(absY)

return imglist

def sobel(im):

x = cv2.Sobel(im, cv2.CV\_16S, 1, 0)

y = cv2.Sobel(im, cv2.CV\_16S, 0, 1)

absX = cv2.convertScaleAbs(x)

absY = cv2.convertScaleAbs(y)

im1 = cv2.addWeighted(absX, 0.5, absY, 0.5, 0)

imglist=[]

imglist.append(im1)

imglist.append(absX)

imglist.append(absY)

return imglist

def lpls(im):

im1=cv2.Laplacian(im,cv2.CV\_64F, ksize=5)

im2=cv2.Laplacian(im,cv2.CV\_64F, ksize=9)

im3=nd.filters.gaussian\_laplace(im,sigma=2)

im3=im3+im

imglist=[im1,im2,im3]

return imglist

def cv2fspz(im):

kernel = cv2.getStructuringElement(cv2.MORPH\_RECT,(20, 20))

eroded = cv2.erode(im,kernel) #腐蚀图像

dilated = cv2.dilate(im,kernel)#膨胀图像

imglist=[eroded,dilated]

return imglist

plt.rcParams['figure.dpi'] = 300

plt.figure(figsize=(10,10))

'''

plt.subplot(341)

plt.imshow(img,'gray')

plt.title('gs\_pic')

plt.subplot(342)

plt.imshow(roberts(img)[0],'gray')

plt.title('roberts')

plt.subplot(346)

plt.imshow(roberts(img)[1],'gray')

plt.title('roberts\_x')

plt.subplot(3,4,10)

plt.imshow(roberts(img)[2],'gray')

plt.title('roberts\_y')

plt.subplot(343)

plt.imshow(Prewitt(img)[0],'gray')

plt.title('Prewitt')

plt.subplot(347)

plt.imshow(Prewitt(img)[1],'gray')

plt.title('Prewitt\_x')

plt.subplot(3,4,11)

plt.imshow(Prewitt(img)[2],'gray')

plt.title('Prewitt\_y')

plt.subplot(344)

plt.imshow(sobel(img)[0],'gray')

plt.title('sobel')

plt.subplot(348)

plt.imshow(sobel(img)[1],'gray')

plt.title('sobel\_x')

plt.subplot(3,4,12)

plt.imshow(sobel(img)[2],'gray')

plt.title('sobel\_y')

'''

'''

plt.subplot(221)

plt.imshow(img,'gray')

plt.title('origin\_pic')

plt.subplot(222)

plt.imshow(lpls(img)[0],'gray')

plt.title('H1\_pic')

plt.subplot(223)

plt.imshow(lpls(img)[1],'gray')

plt.title('H2\_pic')

plt.subplot(224)

plt.imshow(lpls(img)[2],'gray')

plt.title('LOG\_pic')

'''

plt.subplot(131)

plt.imshow(img,'gray')

plt.title('origin\_pic')

plt.subplot(132)

plt.imshow(cv2fspz(img)[0],'gray')

plt.title('eroded\_pic')

plt.subplot(133)

plt.imshow(cv2fspz(img)[1],'gray')

plt.title('dilated\_pic')

plt.show()