第⼆次作业报告

姓名：艾再提艾力·米吉提 学号：3017218180

实验⽬的

为图像加上⾼斯噪声和椒盐噪声，再分别实现多种均值滤波器、统计排序滤波器和⾃适应滤波器对 加了噪⾳后的图⽚进⾏还原处理，分析结果

****实验过程

原图

⾸先调⽤加噪⾳函数为图⽚分别加⾼斯噪声和椒盐噪声

import numpy as np

import cv2

from numpy import shape

import random

from skimage.util import random\_noise

from skimage import io

from tkinter import \*

img = io.imread('1.JPG')

#⾼斯噪声

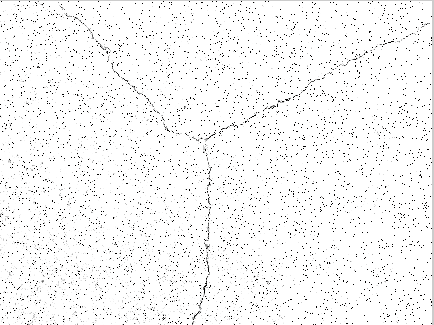
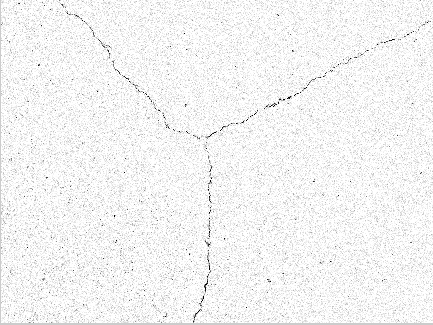
gauss\_img = random\_noise(img,mode='gaussian',seed=5000)

io.imsave('gauss\_1.JPG',gauss\_img)

#椒盐噪声

impulse\_img = random\_noise(img,mode='salt',seed=5000)

io.imsave('impulse\_1.JPG',impulse\_img)



加入高斯噪声后的图像 加入椒盐噪声后的图像

处理图⽚，⽣成定义滤波器，分别对gauss加噪声和椒盐加噪声后图⽚进⾏处理

def deal\_image(path):

image = io.imread(path, as\_gray= True)

med\_img = io.imread(path, as\_gray= True) #中值滤波

geometry\_img = io.imread(path, as\_gray= True) #⼏何均值滤波

mean\_img = io.imread(path, as\_gray= True) #算数均值滤波

max\_img = io.imread(path, as\_gray= True) #最⼤值滤波

min\_img = io.imread(path, as\_gray= True) #最⼩值滤波

mid\_pot\_img = io.imread(path, as\_gray= True) #中点滤波

arf\_img = io.imread(path, as\_gray= True) #修正后的阿尔法滤波

xb\_img = io.imread(path, as\_gray= True) #谐波滤波

back\_xb\_img = io.imread(path, as\_gray=True) #反谐波滤波

for i in range(image.shape[0]):

for j in range(image.shape[1]):

med\_img[i][j] = image[i][j]

geometry\_img[i][j] = image[i][j]

mean\_img[i][j] = image[i][j]

xb\_img[i][j] = image[i][j]

back\_xb\_img[i][j] = image[i][j]

max\_img[i][j] = image[i][j]

min\_img[i][j] = image[i][j]

mid\_pot\_img[i][j] = image[i][j]

arf\_img[i][j] = image[i][j]

return image, med\_img, mean\_img, geometry\_img, xb\_img, back\_xb\_img,

max\_img, min\_img, mid\_pot\_img, arf\_img

#定义滤波器

image, med\_img, mean\_img, geometry\_img, xb\_img, back\_xb\_img, max\_img,

min\_img, mid\_pot\_img, arf\_img = deal\_image('impulse\_1.JPG')#impulse\_1.JPG

均值滤波器 均值滤波器都是使⽤3\*3⼤⼩的滤波器⽤9个像素的均值代替中间的像素，⽤系数为1/mn(step)的 卷积模板来实现 实现算数均值滤波器

#算数均值滤波器

def mean\_filter(x, y, step):

sum\_s = 0

for k in range(-int(step / 2), int(step / 2) + 1):

for m in range(-int(step / 2), int(step / 2) + 1):

sum\_s += image[x + k][y + m] / (step \* step)

return sum\_s

#⼏何均值滤波器

def geometry\_filter(x, y, step):

sum\_s = 0

for k in range(-int(step / 2), int(step / 2) + 1):

for m in range(-int(step / 2), int(step / 2) + 1):

sum\_s \*= image[x + k][y + m]

sum\_r = sum\_s \*\* (1/(step \* step))

return sum\_r

#谐波均值滤波器

def xb\_filter(x, y, step):

sum\_s = 0

for k in range(-int(step / 2), int(step / 2) + 1):

for m in range(-int(step / 2), int(step / 2) + 1):

sum\_s += 1.0/image[x + k][y + m]

sum\_r = (step\*step) / sum\_s

return sum\_r

#逆谐波均值滤波器

def back\_xb\_filter(x, y, step):

sum\_s = 0

q = 1.5

for k in range(-int(step / 2), int(step / 2) + 1):

for m in range(-int(step / 2), int(step / 2) + 1):

sum\_s += image[x + k][y + m] / (step \* step)

sum\_r = (sum\_s \*\* (q+1)) / (sum\_s \*\* q)

return sum\_r

统计排序滤波器

中值滤波器：最著名的顺序统计滤波器是中值滤波器，⽤该像素的相邻像素的灰度中值来替代该像素的值

#中值滤波器

def med\_filter(x, y, step):

sum\_s = []

for k in range(-int(step / 2), int(step / 2) + 1):

for m in range(-int(step / 2), int(step / 2) + 1):

sum\_s.append(image[x + k][y + m])

sum\_s.sort()

return sum\_s[(int(step \* step / 2) + 1)]

#最⼤值滤波器

def max\_filter(x, y, step):

sum\_s = []

for k in range(-int(step / 2), int(step / 2) + 1):

for m in range(-int(step / 2), int(step / 2) + 1):

sum\_s.append(image[x + k][y + m])

sum\_s.sort()

return max(sum\_s)

#最⼩值滤波器

def min\_filter(x, y, step):

sum\_s = []

for k in range(-int(step / 2), int(step / 2) + 1):

for m in range(-int(step / 2), int(step / 2) + 1):

sum\_s.append(image[x + k][y + m])

sum\_s.sort()

return min(sum\_s)

#中点滤波器

def mid\_pot\_filter(x, y, step):

return 0.5 \* (max\_filter(x, y, step) + min\_filter(x, y, step))

# Step为滤波器的⼤⼩ 3\*3

def test(Step):

for i in range(int(Step / 2), image.shape[0] - int(Step / 2)):

for j in range(int(Step / 2), image.shape[1] - int(Step / 2)):

med\_img[i][j] = med\_filter(i, j, Step)

mean\_img[i][j] = mean\_filter(i, j, Step)

geometry\_img[i][j] = mean\_filter(i,j,Step)

xb\_img[i][j] = xb\_filter(i, j, Step)

back\_xb\_img[i][j] = back\_xb\_filter(i, j, Step)

max\_img[i][j] = max\_filter(i, j, Step)

min\_img[i][j] = min\_filter(i, j, Step)

mid\_pot\_img[i][j] = mid\_pot\_filter(i,j,Step)

arf\_img[i][j] = arf\_filter(i, j, Step)

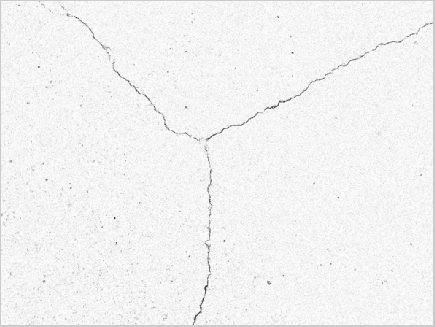
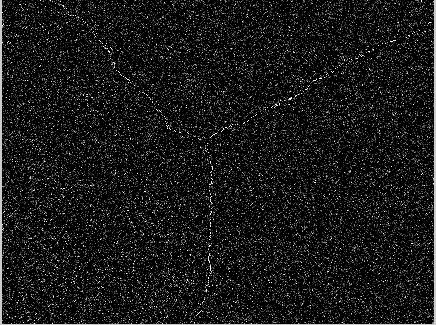
io.imsave(str(Step) + 'impulse\_med.jpg', med\_img)

io.imsave(str(Step) + 'impulse\_mean.jpg', mean\_img)

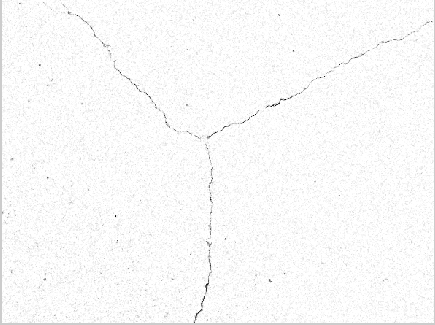
io.imsave(str(Step) + 'impulse\_geometry.jpg', geometry\_img)

io.imsave(str(Step) + 'impulse\_xb.jpg', xb\_img)

io.imsave(str(Step) + 'impulse\_back\_xb.jpg', back\_xb\_img)

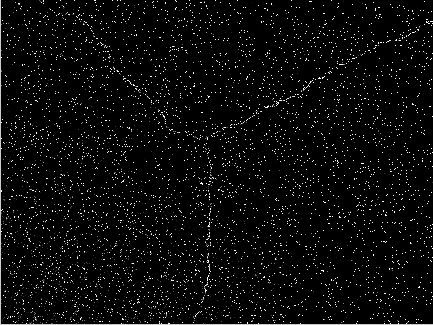
****

均值滤波图像 拉普拉斯滤波图像

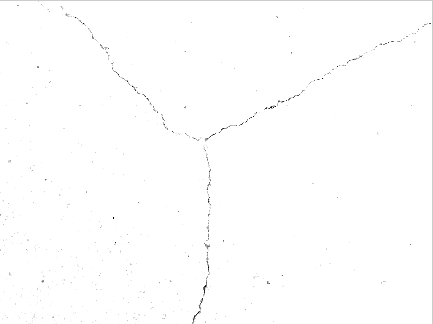
****

中值滤波图像

将加入椒盐噪声后的图像分别进行均值滤波、拉普拉斯滤波和中值滤波。，程序和上述程序类似，滤波后的结果如图

****

均值滤波图像 拉普拉斯滤波图像

****

中值滤波图像