数字图像处理第一次实验报告（文末附代码）

1 问题

动手实验测试均值滤波的效果

2 问题分析

这一实验首先我们要明确对什么噪声进行测试，因为同样的滤波器对不同的噪声会有不同的结果。本实验我对高斯噪声和椒盐噪声分别进行测试，这两种噪声的差异较大，比较能有实验价值。高斯噪声采用均值为0，方差为1的高斯噪声，而椒盐噪声的阈值设为0.02（即噪声像素比例为0.02）

除此之外我还测试了三种不同的加权滤波器。首先是

1滤波器1均值滤波器，权重为 :[ 1/9,1/9,1/9

1/9,1/9,1/9

1/9,1/9,1/9]

2 滤波器2 ，权重为:[ 1/16,1/8,1/16

1/8,1/4,1/9

1/16,1/8,1/16]

2 滤波器3 ，权重为:[ 1/25,3/25,1/25

3/25,9/25,3/25

1/25,3/25,1/25]

可以看到从滤波器1 到3 我逐渐增大了中间像素的贡献，距离中心像素越远的像素贡献越小。我们将三种不同滤波器分别作用在2中不同的噪声图像上。

3 实验结果：

（原图 lena 未加噪声）



高斯噪声不同滤波器的降噪效果：

 (加高斯噪声后的图)

  （从左到右从上到下分别是 filter 0 –filter 3）

椒盐噪声不同的滤波效果:

(加了椒盐噪声后的图)







（从左到右从上到下分别是 filter 0 –filter 3）

4 实验分析

我们可以看到不论对哪种噪声，如果我们越是突出中心像素值的贡献，那么在降噪的时候对图像的细节影响的越小，但同时对噪声的抑制就相对减少了，从图像中看出从filter1 – 3细节越来月清晰，但同时噪声也越来越明显。同时不论是哪种滤波器都会模糊图像的细节，同时抑制噪声起到一定的降噪效果。

但对比不同的噪声类型，我们发现均值滤波对与高斯噪声的降噪效果要好于椒盐噪声，越是突出中心像素贡献的滤波器对椒盐噪声的降噪效果越差，其原因可能是椒盐噪声自身噪声分布比较没有规律,噪声像素值高，用邻域像素的加权平均值比较难对椒盐噪声起到抑制作用。椒盐噪声可以使用中值滤波进行降噪。

均值滤波这种简单的降噪方式计算简单，但是会对图像造成模糊，不是十分理想的降噪方法。

5 代码：

**import** numpy **as** np  
**import** cv2 **as** cv  
  
  
**def** add\_gaussian\_noise (image,weight,mean=0,sdv=1):  
 *'''  
 给原图像添加一个 正太分布的噪声* **:param** *image: 原图像 灰度图* **:param** *weight: 原图像和噪声做加权和的权值* **:param** *mean: 高斯分布均值 默认为 0* **:param** *sdv: 高斯分布的方差，默认为1* **:return***:  
 '''* image = np.array(image,dtype=np.int32)  
 noise = np.random.normal(mean,sdv,image.shape)  
 *#将图像归一化 为 [-1,1]* image = np.array((image-128)/128,dtype=np.float32)  
 *#添加噪声* image = image + weight\* noise  
 *#反映射* image = np.array(image\*128+128,dtype=np.int32);  
 image = np.clip(image,0,255)  
 **return** image  
  
**def** add\_salt\_noise(image,prob):  
 *'''  
 #添加校验噪声，噪声比例为 prob， 高于prob时设置像素值为 255* **:param** *image:* **:param** *prob:* **:return***:  
 '''* image = np.array(image,dtype=np.int32)  
 shape = image.shape  
 **for** i **in** range(shape[0]):  
 **for** j **in** range(shape[1]):  
 th = np.random.uniform(0,1)  
 **if** th<prob:  
 image[i][j]=255  
 **return** image  
  
  
**def** cov2D (image,kernal,stride=1):  
 *'''  
 # 卷积函数 same padding ,stride 默认为 1,不加 bias* **:param** *image:* **:param** *kernal: 滤波器，其size为奇数 nd 数组* **:return***:  
 '''* image = np.array(image,dtype=np.int32)  
 new\_image = image.copy()  
 image\_shape = image.shape  
 k\_shape = kernal.shape  
 padding =[  
 int (((stride-1)\*image\_shape[i]+ k\_shape[i] - stride)/2) **for** i **in** range(2)  
 ]  
 image =np.pad(image,((padding[0],padding[0]),(padding[1],padding[1])),**'constant'**)  
 *# 卷积操作* **for** i **in** range(image.shape[0]-k\_shape[0]+1):  
 **for** j **in** range(image.shape[1]-k\_shape[1]+1):  
 new\_image[i,j] = np.sum(image[i:i+k\_shape[0],j:j+k\_shape[1]] \* kernal) *#* **return** new\_image  
  
**if** \_\_name\_\_ ==**"\_\_main\_\_"**:  
 noise\_function={**"gaussian"**:add\_gaussian\_noise,**"salt"**:add\_salt\_noise}  
 kernal1 = np.ones([3,3])/9  
 kernal2 = np.array([[1/16,1/8,1/16],[1/8,1/4,1/8],[1/16,1/8,1/16]],dtype=np.float32)  
 kernal3 = np.array([[1/25,3/25,1/25],[3/25,9/25,3/35],[1/25,3/25,1/25]],dtype=np.float32)  
 kernals =[kernal1,kernal2,kernal3]  
 **for** kernal **in** kernals:  
 print(kernal)  
 **import** os  
 BASE\_DIR = os.path.dirname(os.path.abspath(\_\_file\_\_))  
 IMAGE\_PATH = os.path.join(BASE\_DIR,**"src"**)  
 RESULT\_PATH = os.path.join(BASE\_DIR,**"result"**);  
  
 images = os.listdir(IMAGE\_PATH)  
 **for** image **in** images:  
 ext = os.path.splitext(image)[1]  
 imgname = os.path.splitext(image)[0]  
 result\_sub\_path = os.path.join(RESULT\_PATH,imgname)  
 **if not** os.path.exists(result\_sub\_path):  
 os.mkdir(result\_sub\_path)  
 **for** noise\_key **in** noise\_function.keys():  
 result\_sub\_sub\_path = os.path.join(result\_sub\_path,noise\_key)  
 **if not** os.path.exists(result\_sub\_sub\_path):  
 os.mkdir(result\_sub\_sub\_path)  
 *# 读取图片 灰度图* image = cv.imread(os.path.join(IMAGE\_PATH, imgname+ext),flags=cv.IMREAD\_GRAYSCALE)  
 *#b 保存原图方便对比* cv.imwrite( os.path.join(result\_sub\_sub\_path,imgname+**'\_original'**)+ext,image)  
 *# 加噪声 对于均值噪声和高斯噪声来说 默认产生 均值为 0 方差为1 的噪声* **if** noise\_key == **"salt"**:  
 noise\_image = noise\_function[noise\_key](image, 0.02)  
 **else**:  
 noise\_image = noise\_function[noise\_key](image, 0.1)  
 *#保存噪声图片* noise\_image\_name = os.path.join(result\_sub\_sub\_path,noise\_key+**"\_"**+imgname+ext)  
 cv.imwrite( noise\_image\_name,noise\_image )  
 **for** i,kernal **in** enumerate(kernals):  
 *#滤波* result\_image = cov2D(noise\_image,kernal)  
 *# 保存* result\_name = os.path.join(result\_sub\_sub\_path,noise\_key+**"\_kernal"**+str(i)+**"\_"**+imgname+ext)  
 cv.imwrite(result\_name, result\_image)