

第二次作业报告

姓名 赵鸿博 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

- 首先调用加噪音函数为图片分别加高斯噪声和椒盐噪声

```
1  import numpy as np
2  import cv2
3  from numpy import shape
4  import random
5  from skimage.util import random_noise
6  from skimage import io
7  from tkinter import *
8
9  img = io.imread('1.JPG')
10 #高斯噪声
11 gauss_img = random_noise(img,mode='gaussian',seed=5000)
12 io.imsave('gauss_1.JPG',gauss_img)
13 #椒盐噪声
14 impulse_img = random_noise(img,mode='salt',seed=5000)
15 io.imsave('impulse_1.JPG',impulse_img)
```

- 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

```
1  def deal_image(path):
2      image = io.imread(path, as_gray= True)
3      med_img = io.imread(path, as_gray= True) #中值滤波
4      geometry_img = io.imread(path, as_gray= True) #几何均值滤波
5      mean_img = io.imread(path, as_gray= True) #算数均值滤波
6      max_img = io.imread(path, as_gray= True) #最大值滤波
7      min_img = io.imread(path, as_gray= True) #最小值滤波
8      mid_pot_img = io.imread(path, as_gray= True) #中点滤波
9      arf_img = io.imread(path, as_gray= True) #修正后的阿尔法滤波
10     xb_img = io.imread(path, as_gray= True) #谐波滤波
11     back_xb_img = io.imread(path, as_gray=True) #反谐波滤波
12
```

```

13     for i in range(image.shape[0]):
14         for j in range(image.shape[1]):
15             med_img[i][j] = image[i][j]
16             geometry_img[i][j] = image[i][j]
17             mean_img[i][j] = image[i][j]
18             xb_img[i][j] = image[i][j]
19             back_xb_img[i][j] = image[i][j]
20             max_img[i][j] = image[i][j]
21             min_img[i][j] = image[i][j]
22             mid_pot_img[i][j] = image[i][j]
23             arf_img[i][j] = image[i][j]
24     return image, med_img, mean_img, geometry_img, xb_img, back_xb_img,
max_img, min_img, mid_pot_img, arf_img
25 #定义滤波器
26 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

```

1 #算数均值滤波器
2 def mean_filter(x, y, step):
3     sum_s = 0
4     for k in range(-int(step / 2), int(step / 2) + 1):
5         for m in range(-int(step / 2), int(step / 2) + 1):
6             sum_s += image[x + k][y + m] / (step * step)
7     return sum_s

```

- 实现几何均值滤波器

```

1 #几何均值滤波器
2 def geometry_filter(x, y, step):
3     sum_s = 0
4     for k in range(-int(step / 2), int(step / 2) + 1):
5         for m in range(-int(step / 2), int(step / 2) + 1):
6             sum_s *= image[x + k][y + m]
7     sum_r = sum_s ** (1/(step * step))
8     return sum_r

```

- 实现谐波均值滤波器

```

1  #谐波均值滤波器
2  def xb_filter(x, y, step):
3      sum_s = 0
4      for k in range(-int(step / 2), int(step / 2) + 1):
5          for m in range(-int(step / 2), int(step / 2) + 1):
6              sum_s += 1.0/image[x + k][y + m]
7      sum_r = (step*step) / sum_s
8      return sum_r

```

- 实现逆谐波均值滤波器

```

1  #逆谐波均值滤波器
2  def back_xb_filter(x, y, step):
3      sum_s = 0
4      q = 1.5
5      for k in range(-int(step / 2), int(step / 2) + 1):
6          for m in range(-int(step / 2), int(step / 2) + 1):
7              sum_s += image[x + k][y + m] / (step * step)
8      sum_r = (sum_s ** (q+1)) / (sum_s ** q)
9      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

```

1  #中值滤波器
2  def med_filter(x, y, step):
3      sum_s = []
4      for k in range(-int(step / 2), int(step / 2) + 1):
5          for m in range(-int(step / 2), int(step / 2) + 1):
6              sum_s.append(image[x + k][y + m])
7      sum_s.sort()
8      return sum_s[(int(step * step / 2) + 1)]

```

- 最大值滤波器

```

1  #最大值滤波器
2  def max_filter(x, y, step):
3      sum_s = []
4      for k in range(-int(step / 2), int(step / 2) + 1):
5          for m in range(-int(step / 2), int(step / 2) + 1):
6              sum_s.append(image[x + k][y + m])
7      sum_s.sort()
8      return max(sum_s)

```

- 最小值滤波器

```

1 #最小值滤波器
2 def min_filter(x, y, step):
3     sum_s = []
4     for k in range(-int(step / 2), int(step / 2) + 1):
5         for m in range(-int(step / 2), int(step / 2) + 1):
6             sum_s.append(image[x + k][y + m])
7     sum_s.sort()
8     return min(sum_s)

```

- 中点滤波器

```

1 #中点滤波器
2 def mid_pot_filter(x, y, step):
3     return 0.5 * (max_filter(x, y, step) + min_filter(x, y, step))

```

- 修正后的阿尔法均值滤波器

```

1 #修正后的阿尔法均值滤波器
2 def arf_filter(x, y, step):
3     sum_s = 0
4     d = 5
5     for k in range(-int(step / 2), int(step / 2) + 1):
6         for m in range(-int(step / 2), int(step / 2) + 1):
7             sum_s += image[x + k][y + m] / (step * step - d)
8     return sum_s

```

- 设置滤波器大小，并调用各种滤波器处理图片，保存图片结果

```

1 # Step为滤波器的大小 3*3
2 def test(Step):
3     for i in range(int(Step / 2), image.shape[0] - int(Step / 2)):
4         for j in range(int(Step / 2), image.shape[1] - int(Step / 2)):
5             med_img[i][j] = med_filter(i, j, Step)
6             mean_img[i][j] = mean_filter(i, j, Step)
7             geometry_img[i][j] = mean_filter(i, j, Step)
8             xb_img[i][j] = xb_filter(i, j, Step)
9             back_xb_img[i][j] = back_xb_filter(i, j, Step)
10            max_img[i][j] = max_filter(i, j, Step)
11            min_img[i][j] = min_filter(i, j, Step)
12            mid_pot_img[i][j] = mid_pot_filter(i, j, Step)
13            arf_img[i][j] = arf_filter(i, j, Step)
14            io.imsave(str(Step) + 'impulse_med.jpg', med_img)
15            io.imsave(str(Step) + 'impulse_mean.jpg', mean_img)
16            io.imsave(str(Step) + 'impulse_geometry.jpg', geometry_img)
17            io.imsave(str(Step) + 'impulse_xb.jpg', xb_img)
18            io.imsave(str(Step) + 'impulse_back_xb.jpg', back_xb_img)

```

```
19 io.imsave(str(Step) + 'impulse_max.jpg', max_img)
20 io.imsave(str(Step) + 'impulse_min.jpg', min_img)
21 io.imsave(str(Step) + 'impulse_midpoint.jpg', mid_pot_img)
22 io.imsave(str(Step) + 'impulse_arf.jpg', arf_img)
23
24 #io.imsave(str(Step) + 'gauss_med.jpg', med_img)
25 #io.imsave(str(Step) + 'gauss_mean.jpg', mean_img)
26 #io.imsave(str(Step) + 'gauss_geometry.jpg', geometry_img)
27 #io.imsave(str(Step) + 'gauss_xb.jpg', xb_img)
28 #io.imsave(str(Step) + 'gauss_back_xb.jpg', back_xb_img)
29 #io.imsave(str(Step) + 'gauss_max.jpg', max_img)
30 #io.imsave(str(Step) + 'gauss_min.jpg', min_img)
31 #io.imsave(str(Step) + 'gauss_midpoint.jpg', mid_pot_img)
32 #io.imsave(str(Step) + 'gauss_arf.jpg', arf_img)
33
34 test(3)
```

实验结果对比

原图 Gauss加噪声 椒盐噪声



Gauss噪声还原

算数均值滤波处理



几何均值滤波处理



谐波均值滤波处理



逆谐波均值滤波处理



中值滤波处理



最大值滤波处理



最小值滤波处理



中点滤波处理



修正后的阿尔法滤波处理



椒盐噪声还原

算数均值滤波处理



几何均值滤波处理



谐波均值滤波处理



逆谐波均值滤波处理



中值滤波处理



最大值滤波处理



最小值滤波处理



中点滤波处理



修正后的阿尔法滤波处理

