

第 9 课 图像处理——平滑

1.图像噪声

在采集、处理和传输过程中，数字图像可能会受到不同噪声的干扰，从而导致图像质量降低、图像变得模糊、图像特征被淹没，而图像平滑处理就是通过除去噪声来达到图像增强的目的。常见的图像噪声有椒盐噪声、高斯噪声等。

1.1 椒盐噪声

椒盐噪声（Salt-and-pepper Noise）也称为脉冲噪声，是一种随机出现的白点或黑点，具体表现为亮的区域有黑色像素，或是暗的区域有白色像素，又或是两者皆有。

下面左侧为图像原图，右侧为添加椒盐噪声的效果图：



1.2 高斯噪声

高斯噪声（Gauss Noise）是指概率密度函数服从高斯分布（正态分布）的一类噪声。除了常用抑制噪声的方法外，常采用数理统计方法对高斯噪声进行抑制。

下面左侧为图像原图，右侧为添加高斯噪声的效果图：



2.图像平滑处理

从信号处理的角度分析，图像平滑就是去除其中的高频信息，保留低频信息，即可以通过低通滤波来去除图像中的噪声，实现对图像的平滑处理。

根据滤波器的不同，滤波方式可分为均值滤波、高斯滤波和中值滤波。

2.1 均值滤波

均值滤波（Mean Filtering）就是对图像的所有像素点进行取均值，即以一个方形区域为单位，将该区域的中心像素点赋值为区域内所有像素点的平均值。

以下图为例，设定 3×3 领域为核，图（a）的中心像素点值为“226”。图（a）中的 9 个像素点值之和除以像素点个数，所得值“122”就是中心像素点的新值，计算式如下：

$$(40 + 107 + 5 + 198 + 226 + 223 + 37 + 68 + 193) \div 9 = 122$$

将中心像素点的值替换为计算得到的新值，如图（b）：

40	107	5
198	226	223
37	68	193

(a)

40	107	5
198	122	223
37	68	193

(b)

均值滤波的算法简单、计算速度较快，但在图像去噪的同时也破坏了图像的细节部分，从而导致图像清晰度降低。

2.2 高斯滤波

将各数值乘以相应的权数后加总求和，并将得到的总体值除以总的单位数，最后求得的值就是加权平均值。

高斯滤波（Gauss Filtering）就是对图像的所有像素点进行加权平均，即以一个方形区域为单位，将其中中心像素点赋值为该区域的加权平均值。

以下图为例，图（a）的中心像素点值为“226”，根据图（b）的权重分配计算加权平均值，所得值“164”就是中心像素点的新值，计算式如下：

$$40 \times 0.05 + 107 \times 0.1 + 5 \times 0.05 + 198 \times 0.1 + 226 \times 0.4 + 223 \times 0.1 + 37 \times 0.05 + 68 \times 0.1 + 193 \times 0.05 = 164$$

将中心像素点的值替换为计算得到的新值，如图（c）：

40	107	5
198	226	223
37	68	193

(a)

×

0.05	0.1	0.05
0.1	0.4	0.1
0.05	0.1	0.05

(b)

=

40	107	5
198	164	223
37	68	193

(c)

2.3 中值滤波

将统计总体中的各个变量按值的大小顺序进行排列，形成一个数列，处于数列中间位置的变量值就称为中值。

中值滤波（Median Filtering）就是对图像的所有像素点进行取中值，即以一个方形区域为单位，将其中心像素点赋值为该区域的中值。

以下图为例，设定 3×3 领域为核，图（a）的中心像素点值为“226”，将图（a）中9个像素点值按照从小到大或从大到小的顺序进行排列，即 5、37、40、68、107、193、198、223、226，中间值“107”就是中心像素点的新值。

将中心像素点的值替换为新值，如图（b）：

40	107	5
198	226	223
37	68	193

(a)

40	107	5
198	107	223
37	68	193

(b)



3.实验步骤

本节例程会对图像分别进行均值滤波、高斯滤波和中值滤波处理。

开始操作前，需要先将目录“第4章 OpenCV 计算机视觉学习->图像处理进阶篇->第10课 图像处理——平滑->例程源码”下的例程“filtering.py”和示例图片“noise.jpg”复制到共享文件夹。

关于共享文件夹的配置方法，可参考目录“第2章 Linux 系统简介及使用入门->Linux 基础课程->第3课 Linux 系统安装及换源方法”下的文档。

注意：输入指令时需严格区分大小写，且可以使用“Tab”键补齐关键字。

1) 打开虚拟机，启动系统。点击系统任务栏的图标，并点击图标，或使用快捷键“**Ctrl+Alt+T**”，打开命令行终端。

2) 输入指令“**cd /mnt/hgfs/Share/**”，并按下回车，进入共享文件夹。

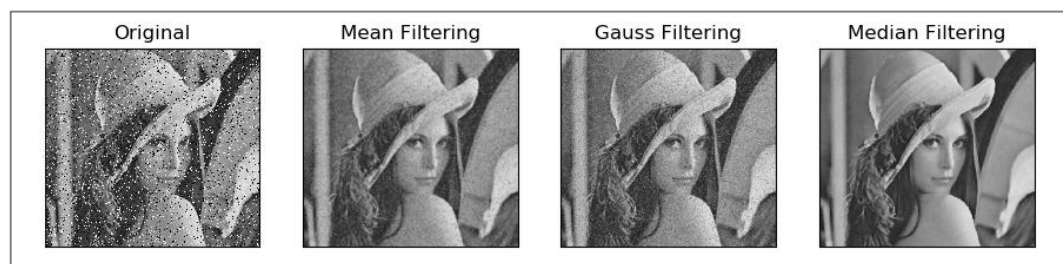
```
hiwonder@ubuntu:~$ cd /mnt/hgfs/Share/
```

3) 输入指令“**python3 filtering.py**”，并按下回车，运行例程。

```
hiwonder@ubuntu:/mnt/hgfs/Share$ python3 filtering.py
```

4.实验效果

执行程序后，图像处理结果如下图所示：



- 1) Original: 原始图像;
- 2) Mean Filtering: 对原图进行均值滤波处理的结果图像;
- 3) Gauss Filtering: 对原图进行高斯滤波处理的结果图像;
- 4) Median Filtering: 对原图进行中值滤波处理的结果图像。

5.程序分析

可以在目录“第4章 OpenCV 计算机视觉学习->图像处理进阶篇->第10课 图像处理——平滑->例程源码”下查看例程“filtering.py”。

```
1 import cv2
2 import numpy as np
3 import matplotlib.pyplot as plt
4
5 # 图像读取
6 img = cv2.imread('noise.jpg')
7
8 # 图像平滑
9 blur1 = cv2.blur(img, (5, 5)) # 均值滤波
10 blur2 = cv2.GaussianBlur(img, (5, 5), 1) # 高斯滤波
11 blur3 = cv2.medianBlur(img, 5) # 中值滤波
12
13 # 图像显示
14 plt.figure(figsize=(10, 5), dpi=100)
15 plt.rcParams['axes.unicode_minus'] = False
16 plt.subplot(141), plt.imshow(img), plt.title("Original")
17 plt.xticks([], plt.yticks([]))
18 plt.subplot(142), plt.imshow(blur1), plt.title("Mean Filtering")
19 plt.xticks([], plt.yticks([]))
20 plt.subplot(143), plt.imshow(blur2), plt.title("Gauss Filtering")
21 plt.xticks([], plt.yticks([]))
22 plt.subplot(144), plt.imshow(blur3), plt.title("Median Filtering")
23 plt.xticks([], plt.yticks([]))
24 plt.show()
```

5.1 图像处理

◆ 导入模块

首先，通过 import 语句导入所需模块。

```
1 import cv2
2 import numpy as np
3 import matplotlib.pyplot as plt
```

◆ 读取图像

通过调用 cv2 模块中的 imread() 函数，读取用于滤波操作的图像。

```
6 img = cv2.imread('noise.jpg')
```

函数括号内的参数是图像名称。

◆ 均值滤波

通过调用 cv2 模块中的 blur() 函数，对指定图像进行均值滤波处理。

```
9 blur1 = cv2.blur(img, (5, 5)) # 均值滤波
```

blur() 函数的语法格式如下：

```
cv2.blur(src, ksize)
```

第一个参数 “**src**” 是输入图像；

第二个参数 “**ksize**” 是卷积核的大小。

◆ 高斯滤波

通过调用 cv2 模块中的 GaussianBlur() 函数，对指定图像进行高斯滤波处理。

```
10 blur2 = cv2.GaussianBlur(img, (5, 5), 1) # 高斯滤波
```

GaussianBlur() 函数的语法格式如下：

```
cv2.GaussianBlur(src, ksize, sigmaX, sigmaY, borderType)
```

第一个参数 “**src**” 是输入图像；

第二个参数 “**ksize**” 是高斯卷积核的大小，卷积核的高度和宽度都必须为正数和奇数；

第三个参数 “**sigmaX**” 是水平方向的高斯核标准偏差；

第四个参数 “**sigmaY**” 是垂直方向的高斯核标准偏差，默认值为 “0”，表示与参数 “**sigmaX**” 相同；

第五个参数 “**borderType**” 是边界的填充类型。

◆ 中值滤波

通过调用 cv2 模块中的 medianBlur() 函数，对指定图像进行中值滤波处理。

```
11 blur3 = cv2.medianBlur(img, 5) # 中值滤波
```

medianBlur()函数的语法格式如下：

```
cv2.medianBlur(src, ksize)
```

第一个参数 “**src**” 是输入图像；

第二个参数 “**ksize**” 是卷积核的大小。

5.2 图像显示

◆ 创建自定义图像

通过调用 matplotlib.pyplot 模块中的 figure()函数，创建一个自定义画像（Figure），用于显示滤波处理的结果图像。

```
14 plt.figure(figsize=(10, 5), dpi=100)
```

figure()函数的语法格式如下：

```
matplotlib.pyplot.figure(num=None,      figsize=None,      dpi=None,      facecolor=None,  
edgecolor=None, frameon=True, FigureClass=<class 'matplotlib.figure.Figure'>, clear=False,  
**kwargs)
```

第一个参数 “**num**” 是画像的唯一标识符，即画像编号（数字）或名称（字符串）；

第二个参数 “**figsize**” 是画像的宽和高，单位为英寸；

第三个参数 “**dpi**” 是画像的分辨率，即每英寸的像素数；

第四个参数 “**facecolor**” 是画像的背景颜色；

第五个参数 “**edgecolor**” 是画像的边框颜色；

第六个参数 “**frameon**” 用于决定是否绘制画像，默认值为 “**True**” ；

第七个参数 “**FigureClass**” 是生成画像时选用的自定义 Figure；

第八个参数 “**clear**” 用于决定当画像存在时是否清除原有画像；

第九个参数 “****kwargs**” 是画像的其他属性。

◆ 修改 matplotlib 配置

matplotlib 是 Python 的绘图库,用户可以通过参数字典“rcParams”访问与修改 matplotlib 的配置项。

```
15 plt.rcParams[axes.unicode_minus] = False
```

上图所示代码用于控制正常字符显示。

◆ 设置图像显示参数

通过调用 matplotlib.pyplot 模块中的 subplot()、imshow()和 title()函数,指定画像 (Figure) 内的子图位置分布、子图颜色类型和子图标题。

```
16 plt.subplot(141), plt.imshow(img), plt.title("Original")
```

1) subplot()函数用于设置子图的位置分布,其语法格式如下:

```
matplotlib.pyplot.subplot(nrows, ncols, index, **kwargs)
```

第一个参数“nrows”和第二个参数“ncols”分别是 subplot 的行、列数;

第三个参数“index”是索引位置,从“1”开始累计(左上角为“1”),依次向右递增。

当行、列数都小于“10”,可以将两个数值缩写为一个整数,例如,代码“subplot(1, 4, 1)”和“subplot(141)”的含义相同,都表示将画像分为 1 行 4 列,当前子图排在第 1 位,即第 1 行的第 1 列。

2) imshow()函数用于设置子图颜色类型,其语法格式如下:

```
matplotlib.pyplot.imshow(X, cmap=None)
```

第一个参数“X”是图像数据;

第二个参数“cmap”是颜色图谱(colormap),默认为 RGB(A) 颜色空间。

3) title()函数用于设置子图标题,函数括号内的参数是子图名称,其语法格式如下。

```
matplotlib.pyplot.title(label, fontdict=None, loc=None, pad=None, *, y=None, **kwargs)
```

第一个参数 “**label**” 是标题文本，类型为字符串；

第二个参数 “**fontdict**” 是标题文本的字体属性，类型为字典；

第三个参数 “**loc**” 是标题位置，可取值 “**left**”、“**center**” 或 “**right**”，默认值为 “**center**”；

第四个参数 “**pad**” 是标题与子图的填充距离（内边距），默认值为 “**6.0**”；

第五个参数 “**y**” 是标题在子图中的垂直距离，单位为子图高度的百分比，默认值为 “**None**”，即自动确定标题位置，避免与其他元素重叠。值为 “**1.0**” 表示在子图顶端；

第六个参数 “****kwargs**” 是文本对象关键字属性，用于控制文本的外观属性，如字体、文本颜色等。

◆ 设置坐标轴刻度

通过调用 `matplotlib.pyplot` 模块中的 `xticks()` 和 `yticks()` 函数，设置 X、Y 轴的刻度及标签，由于此例程在显示图像时无需使用坐标轴信息，此处将列表设为空，即不显示坐标轴。

```
17 plt.xticks([], plt.yticks([]))
```

`xticks()` 函数的语法格式如下：

```
xticks(ticks=None, labels=None, **kwargs)
```

当参数为空，函数会返回当前 X 轴的刻度及标签；否则，函数用于设置当前 X 轴的刻度及标签。

第一个参数 “**ticks**” 是 X 轴刻度的位置列表，若列表为空，X 轴刻度将清空；

第二个参数 “**labels**” 是 X 轴刻度的标签，当参数 “**ticks**” 不为空，此参数才会传递；

第三个参数 “****kwargs**” 用于控制刻度标签的外观。

`yticks()` 函数的语法格式和 `xticks()` 函数的相同，区别在于 `yticks()` 函数的作用对象是 Y 轴。

◆ 显示图像

通过调用 matplotlib.pyplot 模块中的 show() 函数，在窗口显示图像。

```
24 plt.show()
```

图像显示部分的完整代码如下：

```
14 plt.figure(figsize=(10, 5), dpi=100)
15 plt.rcParams['axes.unicode_minus'] = False
16 plt.subplot(141), plt.imshow(img), plt.title("Original")
17 plt.xticks(), plt.yticks()
18 plt.subplot(142), plt.imshow(blur1), plt.title("Mean Filtering")
19 plt.xticks(), plt.yticks()
20 plt.subplot(143), plt.imshow(blur2), plt.title("Gauss Filtering")
21 plt.xticks(), plt.yticks()
22 plt.subplot(144), plt.imshow(blur3), plt.title("Median Filtering")
23 plt.xticks(), plt.yticks()
24 plt.show()
```