华北水利水电大学

North China University of Water Resources and Electric Power

《数字图像处理》
实验报告(2)

院　　系：　信息工程学院

专　　业：　人工智能

班　　级：　2020185

姓　　名：　高树林

学　　号：　202018526

指 导 教 师：　韩光辉

2022-2023 学年 第一学期

实验二 空域图像增强

一、实验目的

通过实验，巩固所学的理论内容，掌握图像空域滤波的工程实现方法。

(1) 掌握灰度变换的实现方法；

(2) 理解并掌握点运算的实现方法；

(3) 理解并掌握空域平滑和锐化滤波方法；

(4) 理解点运算与邻域运算的差别与联系。

二、实验内容和要求

实验报告内容应格式统一(注意同一个大纲级别下的字体、字号、行间距等应统一，英文用 Times New Roman 字体)，整体美观整洁。实验代码可读性强，包含适量注释，说明求解思路和过程。

1. 编程实现图像的灰度变换

包括图像变暗、图像变亮、降低对比度和直方图均衡化处理。

提示：

前三个处理可通过灰度值增加或减少一个常数值、全局线性变换等实现，直方图均衡化可通过调用 OpenCV 工具包中的 equalizeHist()函数实现。

全局线性变换：如果要将灰度范围从[a，b]更改为[c，d]，则：

$$s = (\frac{d-c}{b-a})(r-a)+c$$

如果扩展到最大灰度范围[0，255]，则：

$$s = (\frac{255}{b-a})(r-a)$$

r 是输入灰度，s 是输出灰度，斜率小于 1，则灰度范围压缩。

2. 编程实现图像的 gamma 变换

提示：将灰度值先归一化到[0,1]之间，然后执行 gamma 变换，最后再还原回[0,255]之间。

3. 验证实验：阅读以下程序，掌握图像空域滤波方法。

修改代码，使用以下滤波算子对图像进行滤波，查看滤波效果，并从平滑/锐化、轮廓模糊/轮廓清晰等方面分析滤波效果。

$$\frac{1}{9}\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix} \qquad \frac{1}{10}\begin{bmatrix}1&1&1\\1&2&1\\1&1&1\end{bmatrix} \qquad \begin{bmatrix}-1&-1&-1\\-1&9&-1\\-1&-1&-1\end{bmatrix}$$

代码（仅供参考，可修改优化）

```
import matplotlib.pyplot as plt
import cv2
import    numpy as np
img = cv2.imread(r"..\img\kennysmall.jpg",0) #使用此图像，需据实修改路径
cv2.imshow('original image',img)
#  设置滤波核
kernel = 1/16 * np.array([[1,2,1],
                          [2,4,2],
                          [1,2,1]])
#  使用 OpenCV 的卷积函数
ImgSmoothed =cv2.filter2D(img,-1,kernel, borderType=cv2.BORDER_DEFAULT)
cv2.imshow('Smoothed image',ImgSmoothed)   #显示滤波后的图像
cv2.waitKey(0)
```

4. 编程实现拉普拉斯锐化和拉普拉斯锐化增强。

提示：拉普拉斯锐化和拉普拉斯锐化增强对应的滤波核分别为

$$\begin{bmatrix}0 & 1 & 0\\ 1 & -4 & 1\\ 0 & 1 & 0\end{bmatrix} \quad 和 \quad \begin{bmatrix}0 & -1 & 0\\ -1 & 5 & -1\\ 0 & -1 & 0\end{bmatrix}$$

5. 添加和抑制噪声实验

编写程序，首先对图像添加椒盐噪声或高斯噪声，然后对加噪图像进行均值滤波、中值滤波和高斯滤波，查看并分析滤波效果(需附结果展示图像)。

提示：

（1）给图像添加椒盐噪声的 Python 参考代码：

```
def addSaltAndPepper(src, percentage):
    #NoiseImg = src     #使用此语句传递的是地址，程序会出错
    #  在此要使用 copy 函数，否则 src 和主程序中的 img 都会跟着改变
    NoiseImg = src.copy()
    NoiseNum = int(percentage * src.shape[0] * src.shape[1])
    for i in range(NoiseNum):
        #  注意需要引入 random 包
        #  产生[0, src.shape[0] - 1]之间随机整数
        randX = random.randint(0, src.shape[0] - 1)
        randY = random.randint(0, src.shape[1] - 1)
        if random.randint(0, 1) == 0:
            NoiseImg[randX, randY] = 0
        else:
            NoiseImg[randX, randY] = 255
    return NoiseImg
```

（2）给图像添加高斯噪声的 Python 参考代码：

```
def addGaussianNoise(src,means,sigma):
    NoiseImg=src.copy()
    NoiseImg=NoiseImg/NoiseImg.max()
    rows=NoiseImg.shape[0]
    cols=NoiseImg.shape[1]
    for i in range(rows):
        for j in range(cols):
            # Python 里使用 random.gauss  函数添加高斯噪声
            NoiseImg[i,j]=NoiseImg[i,j]+ random.gauss(means,sigma)
            if   NoiseImg[i,j]< 0:
                NoiseImg[i,j]=0
            elif   NoiseImg[i,j]>1:
                NoiseImg[i,j]=1
    NoiseImg=np.uint8(NoiseImg*255)
    return NoiseImg
```

（3）OpenCV 工具包中实现均值滤波、中值滤波和高斯滤波的函数：

```
cv2.blur(img,(3,3))
cv2.medianBlur(img,3)
cv2.GaussianBlur(img,(3,3),1)
```

三、实验环境(学生填写)

（较详细地说明实验运行环境，包括操作系统、编程 IDE、编程语言及版本号、依赖的第三方类库及版本号等）

本实验环境分为硬件和软件两大方面叙述。

硬件方面：本实验室基于处理器为 Intel(R) Core(TM) i5-10210U CPU @ 1.60GHz 2.11 GHz，基带 RAM 为 16.0 GB (15.8 GB 可用)，系统类型为 64 位操作系统，基于 x64 的处理器的 MagicBook 商务本上运行并得出相应结果和数据的。

软件方面：本实验基于 windows10 家庭中文版，版本号为 21H2，操作系统内部版本为 19044.2130 的 MagicBook 商务本。编程语言选择 python 语言，使用运行时版本号为 11.0.14.1+1-b2043.45 amd64 的 PyCharm 2022.1.1 (Professional Edition)集成开发环境编写代码。其中代码的 python 解释器版本为 Python 3.8.13 (default, Mar 28 2022, 06:59:08) [MSC v.1916 64 bit (AMD64)] :: Anaconda, Inc. on win32。引入 1.0.0.14 版本的 OpenCV-python 库，其中集成的 OpenCV 版本为 4.6.0。引入 Matplotlib3.5.3 版本的库。

四、实验过程(学生填写)

1. 编程实现图像的灰度变换、Gamma 变换

(1)实验方案（设计思路说明）

1.通过 OpenCV 读取图像数据，之后基于这个数据对图形进行多种变化处理。

2.利用 numpy 库中的 clip()函数对 1 中读取的矩阵进行加、减、乘处理来实现图像的变亮、变暗、降低对比度的操作，之后利用 OpenCV 工具包中的 equalizeHish()函数对其操作得到均衡化的图像。

3.利用 plt.show()函数来显示各个图像。

(2)代码编写（包含适量注释）

```
# -*- coding: utf-8 -*-
from pylab import mpl
import cv2 as cv
import numpy as np
from matplotlib import pyplot as plt

# 设置显示中文字体
mpl.rcParams["font.sans-serif"] = ["SimHei"]
# 设置正常显示符号
mpl.rcParams["axes.unicode_minus"] = False
plt.subplots_adjust(left=None, bottom=None, right=None, top=None, wspace=0.5, hspace=0.5)  # 设置子图左右和上下间隔，一般为 0.5 为佳


fig = plt.figure(figsize=(11,4))

def gammaTransform(imgDir):
    imgGray = cv.imread(imgDir, cv.IMREAD_GRAYSCALE)  # OpenCV 读取灰度图
    imgGrayNorm = imgGray / 255  # 像素值归一化
    gamma = 2  # gamma 系数
    dst = np.power(imgGrayNorm, gamma)  # gamma 变换
    plt.subplot(1, 6, 6)
    plt.imshow(dst,'gray')
    plt.xlabel('gamma 变换',fontsize='10')
    plt.show()


plt.subplot(1, 6, 1)
img = cv.imread("..\\figures\\img\\lenna.png", 0)
plt.imshow(img,'gray')
plt.xlabel('原图',fontsize=10)
# 增加图像亮度
# 注意需要使用 cv.add(),不能直接 x+y
plt.subplot(1, 6, 2)
res1 = np.uint8(np.clip((cv.add(1 * img, 100)), 0, 255))  # 每个像素点加 100 点像素值
```

```
plt.imshow(res1,'gray')
plt.xlabel('灰度加 100',fontsize=10)
# 注意需要使用 cv.add(),不能直接 x+y
plt.subplot(1, 6, 3)
res2 = np.uint8(np.clip((cv.add(1 * img, -100)), 0, 255))  # 每个像素点加 -100 点像素值
plt.imshow(res2,'gray')
plt.xlabel('灰度减 100',fontsize=10)
# 增加图像对比度
plt.subplot(1, 6, 4)
res3 = np.uint8(np.clip((cv.add(0.01 * img, 0)), 0, 255))  # 整个像素缩小到原来的 1/100
plt.imshow(res3,'gray')
plt.xlabel('降低对比度',fontsize=10)
# 直方图均衡化
plt.subplot(1, 6, 5)
plt.imshow(cv.equalizeHist(cv.imread('..\\figures\\img\\lenna.png', cv.IMREAD_GRAYSCALE)), 'gray')  # 转为均衡图并显示
plt.xlabel('均衡化图像', fontsize=10)
gammaTransform('..\\figures\\img\\lenna.png')
plt.savefig('./灰度图.jpg')
plt.show()
```

(3)调试（编码实现过程中遇到的错误，及调试解决等内容）

问题 1：在编写代码过程中，在最后将 6 个图象同时显示在一张图上，图像过小，不美观，截图如下图 1 所示。

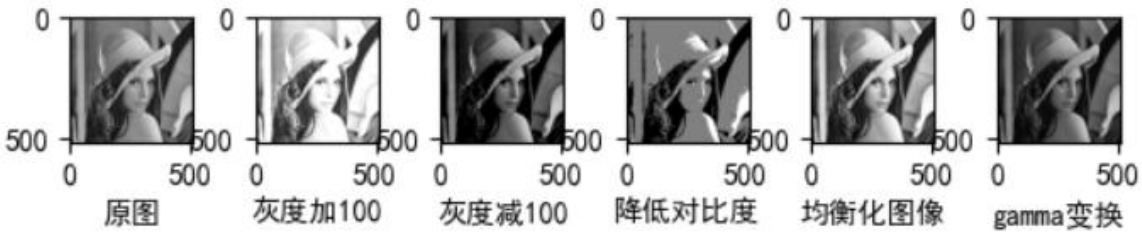

图 1 问题 1 截图

结果 1：经过在网络上搜寻资料，我明白了原来是默认画布过大，重新自定义画布就能使图片的相对大小有所增加。解决方案如下图 2 所示。

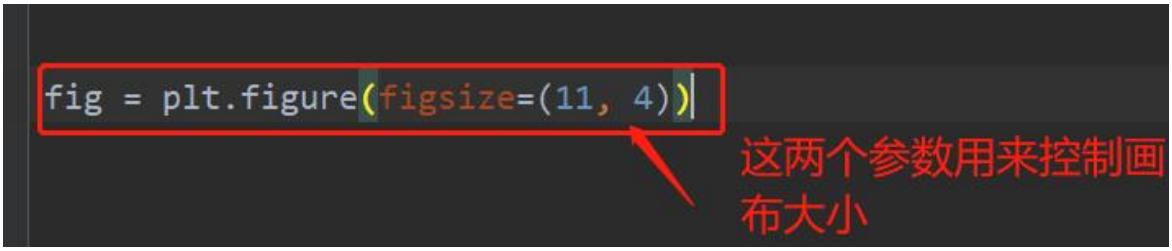

图 2 问题 1 解决方案

问题 2：不能输出灰度图，输出的图片是绿色背景图。

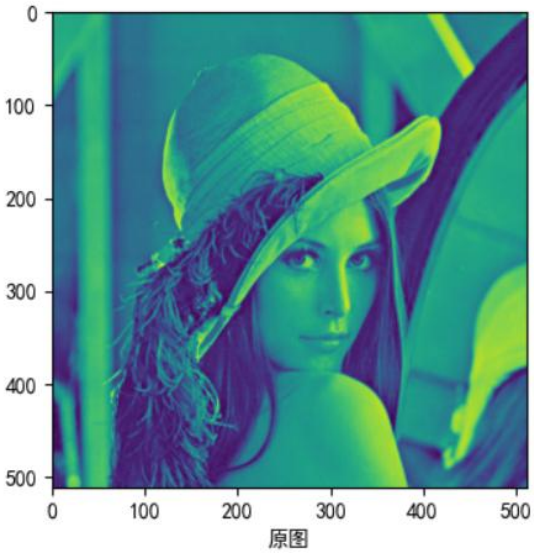

图 3 问题 2 截图

解决 2：通过查阅资料发现这是因为在 Matplotlib 工具包里，需要使用 cmap 来定义灰度图。因此我修改了代码入选图 4 所示成功解决问题。

图 4 问题 2 解决方案

(4)实验结果（实验运行结果截图等体现实验结果的内容）

结果 1：利用 OpenCV 读取图片数据，并将其转化为灰度图，最后使用 Matplotlib 工具包显示图片的结果如下图 5 所示。

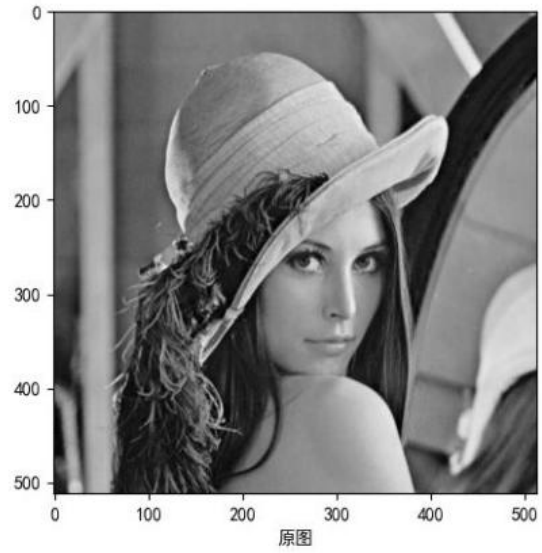

图 5 OpenCV 对原图进行灰度图像处理

结果 2：利用 OpenCV 库读取图片数据，并将其转化为灰度图后，对每个像素点的值加 100 点像素，得到的新的图片数据最后使用 Matplotlib 工具包显示图片的

结果如下图 6 所示。

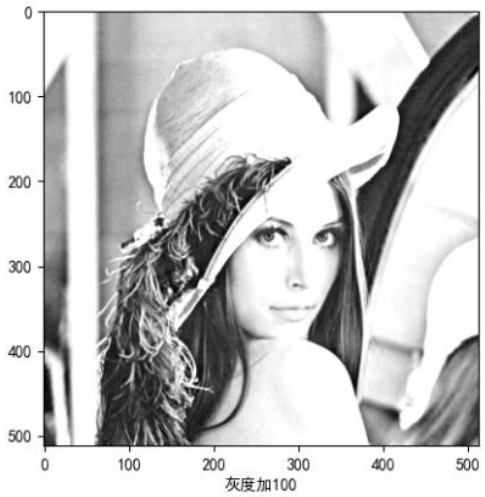

图 6 OpenCV 对原图进行灰度图像像素加 100 处理

结果 3：利用 OpenCV 库读取图片数据，并将其转化为灰度图后，对每个像素点的值减掉 100 点像素，得到的新的图片数据最后使用 Matplotlib 工具包显示图片的结果如下图 7 所示。

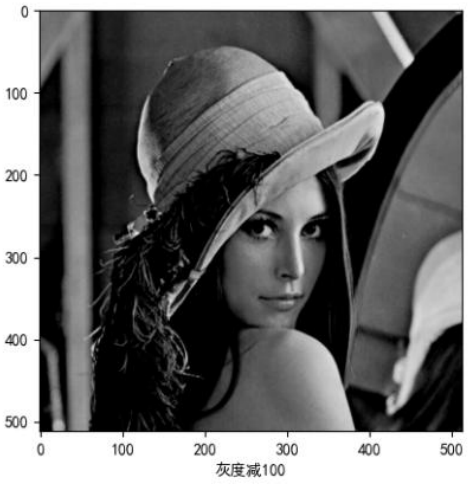

图 7 OpenCV 对原图进行灰度图像像素减 100 处理

结果 4：利用 OpenCV 库读取图片数据，并将其转化为灰度图后，降低对比度得到的新的图片数据最后使用 Matplotlib 工具包显示图片的结果如下图 8 所示。

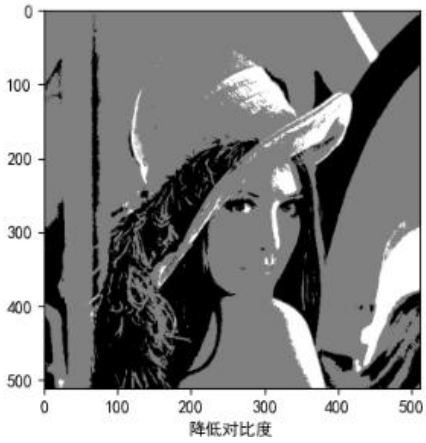

图 8 OpenCV 对原图进行灰度图像降低对比度处理

结果 5：利用 OpenCV 库读取图片数据，并将其转化为灰度图后，均衡化后得到的新的图片数据最后使用 Matplotlib 工具包显示图片的结果如下图 9 所示。

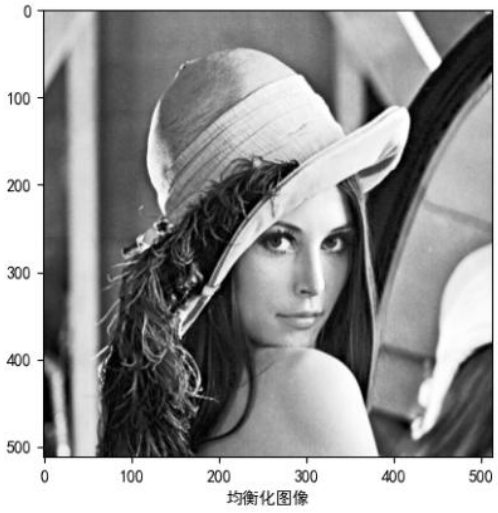

图 9 OpenCV 对原图进行灰度图像均衡化处理

结果 6：利用 OpenCV 库读取图片数据，并将其转化为灰度图后，归一化后经过 gamma 变换，再将像素还原到[0，255]后得到的新的图片数据最后使用 Matplotlib 工具包显示图片的结果如下图 10 所示。

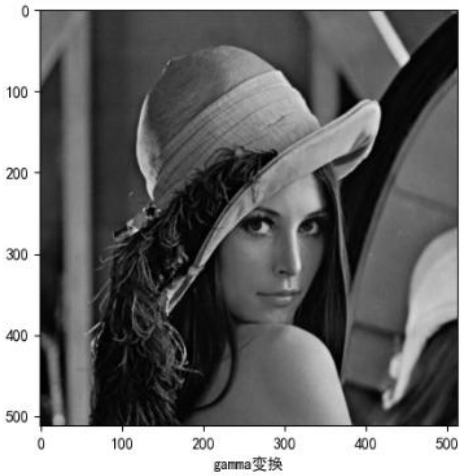

图 10 OpenCV 对原图进行灰度图像 gamma 变化处理

(5)分析与总结（与本次实验相关的分析与总结）

为了方便对比，将上述五副图片放在一张图上比较如下图 11 所示。

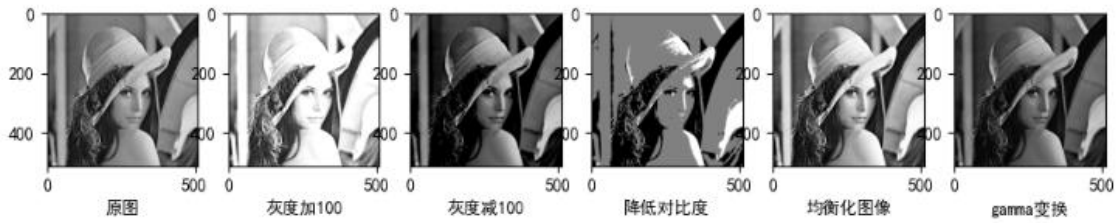

图 11 实验 1 的 5 个实验结果图

从图的原图和灰度增加 100 的结果图对比不难看出，当一幅图像的每一个像素点的灰度值增加，回使原来的图像更亮，但是当增加的值不合理时，会使原图像的彩色图片发绿，导致图片的色彩混乱。通过原图和灰度减 100 的结果图对比不难看出，当衣服图像的灰度值减小，会导致原来的图像偏暗，若减小的值高于某一值时，原来的图像会变得昏暗，识别度低。对比原图和降低对比度图不难看出，当图像对比度将对后，图像将变得灰蒙蒙的。因此提高图像对比度的目的一般有改善图像的视觉效果、转换为更适合于人或机器分析处理的形式、突出对人或机器分析有意义的信息抑制无用信息、提高图像的使用价值。但是通过提高对比度得到的图像一般不一定保真。通过原图与均衡化图像的对比不难看出经过均衡化处理的图像要比原来的图像更为清晰。这是因为直方图均衡化处理之后，原来比较少像素的灰度会被分配到别的灰度去，像素相对集中，处理后灰度范围变大，对比度变大，清晰度变大，所以能有效增强图像。通过原图和 gamma 处理的图像的对比不难看出，gamma 图像更接近自然，最接近人类感知。这是因为 gamma 函数用的幂函数运算方式接近人类感知亮度的方式。事实上当大部分图像的成色时以线性方式对光强编码，这导致了很多种情况下，肉眼就能分辨出真假图片。

通过本次实验我掌握了如何改变图像的灰度值来使他们在不同的场景下达到对应的效果。同时我还明白了，通过提高对比度和使用图像均衡化都能够使图像更为清晰。此外，通过对 gamma 变换处理我明白了人类对光的感知并非是线性的，而是一种类似于幂函数的关系，这个常识会在以后我进行计算机视觉的成像方面的领域研究时给予我极大的帮助！

2. 编程实现图像的空域滤波、拉普拉斯空域锐化

(1)实验方案（设计思路说明）

空域滤波

1.首先要设置滤波核，滤波核的格式必须为 np.array 格式，否则不能正常读取，参与运算会报错。

2.利用 OpenCV 工具包中的 cv2.filter2D 函数处理图像，选取 1 中卷积核为参数就能实现以 1 中滤波核为卷积核的卷积运算了。

拉普拉斯空域锐化

1. 首先要设置一个卷积核，让图像按照该卷积核进行卷积运算.
2. 让原图像的每个像素点和卷积后得到的图像的同一位置做减运算。
3. 将 2 中的每一个像素点进行遍历，当遍历的点的值小于 0，将其像素点置为 0，

若遍历的点的值大于 255，将其值设置为 255。

4. 将经过 3 变换的图像输出显示。

(2)代码编写（包含适量注释）

```
# 在这里编辑拉普拉斯锐化和拉普拉斯锐化增强的代码
import matplotlib.pyplot as plt
import cv2
import  numpy as np
img = cv2.imread("work/img/kennysmall.jpg") #使用此图像，需据实修改路径
img = cv2.cvtColor(img,cv2.COLOR_BGR2RGB)

# 设置滤波核
kernel1 = 1/16 * np.array([[1,2,1],
                           [2,4,2],
                           [1,2,1]])

kernel2 = 1/9 * np.array([[1,1,1],
                          [1,1,1],
                          [1,1,1]])

kernel3 = 1/10 * np.array([[1,1,1],
                           [1,2,1],
                           [1,1,1]])

kernel4 = np.array([[-1,-1,-1],
                    [-1,9,-1],
                    [-1,-1,-1]])
kernel = [kernel1,kernel2,kernel3,kernel4]
# 使用 OpenCV 的卷积函数
for i in range(0,4):
    ImgSmoothed =cv2.filter2D(img,-1,kernel[i], borderType=cv2.BORDER_DEFAULT)
    # 展示结果
    plt.figure()
    plt.subplot(4,2,1+2*i)
    plt.axis("off")
    plt.title('original image')
    plt.imshow(img)
    plt.subplot(4,2,2*i+2)
    plt.axis("off")
    plt.title('Smoothed image use kernel'+str(i+1))
    plt.imshow(ImgSmoothed)
plt.show()

```

```
# 在这里编辑拉普拉斯锐化和拉普拉斯锐化增强的代码
import cv2
import numpy as np
import matplotlib.pyplot as plt


def Laplace(img, kernel,i,name):  # 传入 i 的作用是控制画布显示，传入 name 是为了命名方便
    des_16S = cv2.filter2D(img, ddepth=cv2.CV_16SC1, kernel=kernel, borderType=cv2.BORDER_DEFAULT)
    g = img - des_16S
    g[g < 0] = 0    #变换后小于 0 的像素值置 0
    g[g > 255] = 255  #变换后大于 0 的像素值置 255

    plt.figure(figsize=(10, 14))

    plt.subplot(221+i)
    plt.imshow(img, cmap='gray')
    plt.title('origin')

    plt.subplot(222+i)
    plt.imshow(g, cmap='gray')
    plt.title(name+' Laplace')
    plt.show()
if __name__ == '__main__':
    img0 = 'work/img/Lenna.png'
    # 拉普拉斯算子锐化
    kernel1 = np.asarray([[0, 1, 0],
                          [1, -4, 1],
                          [0, 1, 0]])  # 定义拉普拉斯算子
    # 拉普拉斯算子锐化增强
    kernel2 = np.asarray([[0, -1, 0],
                          [-1, 5, -1],
                          [0, -1, 0]])  # 定义拉普拉斯算子
    f = cv2.imread(img0, cv2.IMREAD_GRAYSCALE)
    Laplace(f, kernel1,0,'kernel1')
    Laplace(f, kernel2,2,'kernel2')
```

(3)调试（编码实现过程中遇到的错误，及调试解决等内容）

问题 1：在调试代码的过程中，在 PyCharm 平台上运行代码正常，但是在代码提交平台 AI Studio 提交时，即使在已经引用了全局的字体设置也仍然会出现中文显示不出来的问题。截图如下图 12 所示。

图 12 显示图像时中文显示出错

解决 1：原因时由于 AI Studio 平台自身的原因，不能自动将全局的中文定义到每一个中文 label 上。解决方案是在每一次写中文标签时，额外定义一下该文字的字体。解决问题 1 后的截图如下图 13 所示。

图 13 问题 1 解决后的截图

(4)实验结果（实验运行结果截图等体现实验结果的内容）

结果 1：利用第一个卷积核做空域滤波变换之后的结果如下图 14 所示。

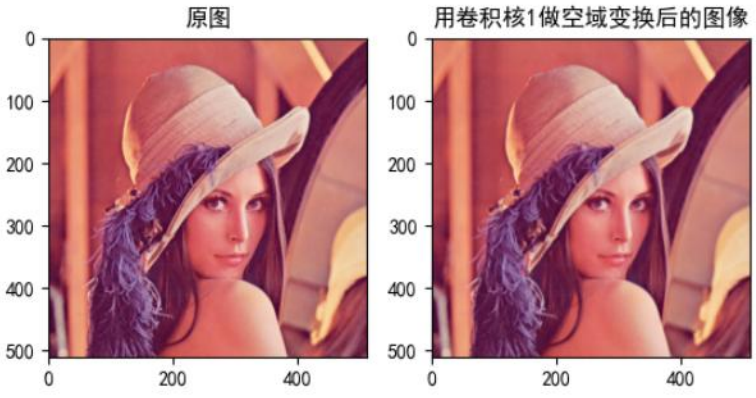

图 14 利用第一个卷积核做空域滤波的结果

结果 2：利用第二个卷积核做空域滤波变换之后的结果如下图 15 所示。

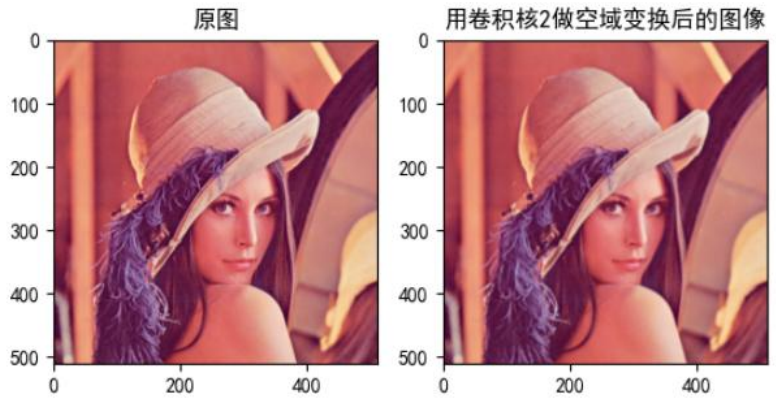

图 15 利用第二个卷积核做空域滤波的结果

结果 3：利用第三个卷积核做空域滤波变换之后的结果如下图 16 所示。

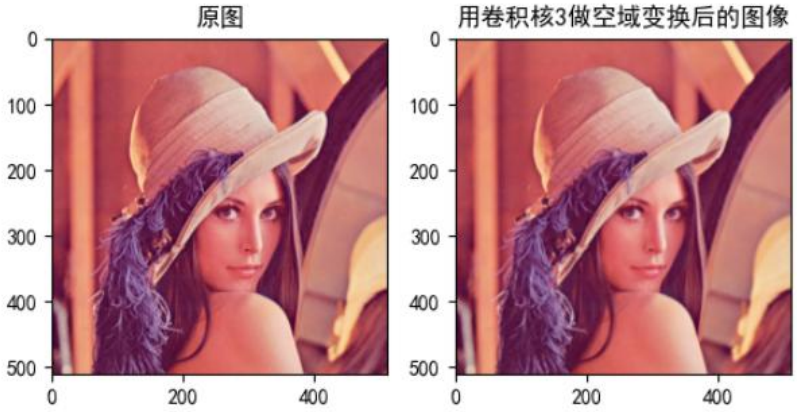

图 16 利用第三个卷积核做空域滤波的结果

结果 4：利用第四个卷积核做空域滤波变换之后的结果如下图 17 所示。

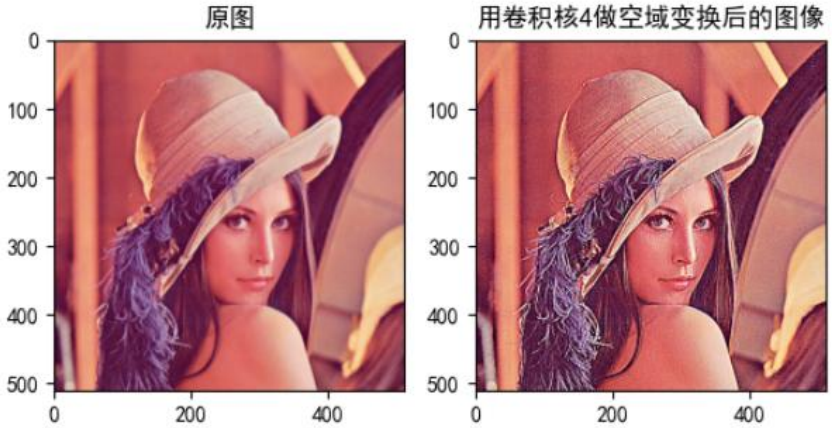

图 17 利用第四个卷积核做空域滤波的结果

结果 5：利用第一个卷积核做拉普拉斯空域锐化变换之后的结果如下图 18 所示。

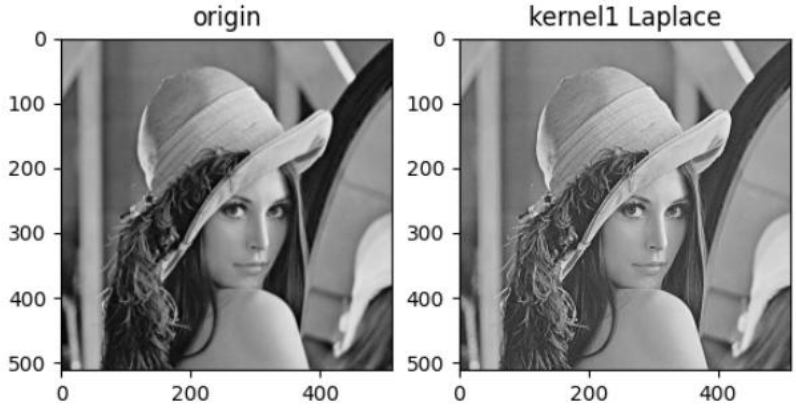

图 18 利用第一个卷积核做拉普拉斯空域锐化的结果

结果 6：利用第二个卷积核做拉普拉斯空域锐化变换之后的结果如下图 19 所示。

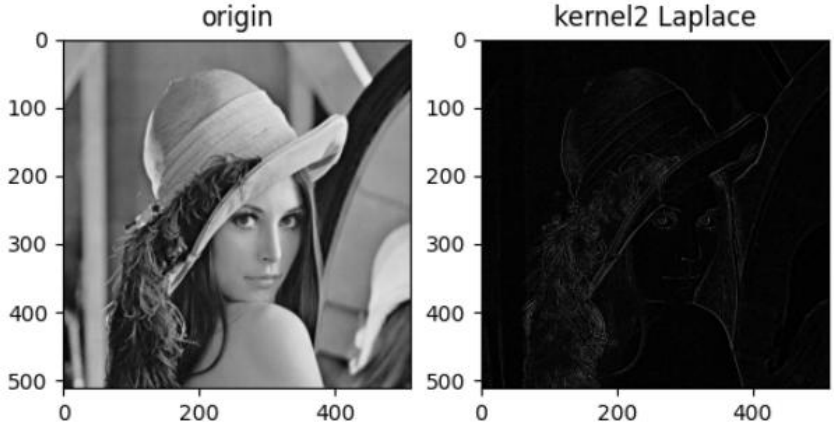

图 19 利用第二个卷积核做拉普拉斯空域锐化的结果

(5)分析与总结（与本次实验相关的分析与总结）

通过本次实验，我对空域滤波和拉普拉斯空域锐化的知识点更加牢固了。

所谓空域滤波，实际上就是图像基于领域处理的增强方法，直接在图像所在的二维空间进行处理，即对每一个像素的灰度值进行处理。它应用某一模版对每个像素与其周围邻域的所有像素进行某种数学运算得到该像素的灰度值，新的灰度值的大小不仅域该像素的灰度值有关，而且还与其领域内的像素值的灰度有关。一般来说，空域滤波是通过邻域操作来完成的，最常用的运算是卷积运算。

在做完拉普拉斯空域滤波之后我明白了：拉普拉斯微分处理后，有些点像素值为负值，需要对这些值进行处理，否则最后的图像中会有很多小白点。同样的，处理后的图像的灰度级可能大于 255，这时需要将其灰度级置为 255。拉普拉斯算子处理后的图像要和原图像相加，可以复原背景特性并保持拉普拉斯锐化处理的效果。如果所使用的模板中心是负数，那么必须将原图像减去经拉普拉斯变换后的图像。如果像我这里使用的模板中心是正数，那么就将原图像加上经过拉普拉斯变换后的图像。

3. 添加和抑制噪声实验

(1)实验方案（设计思路说明）

添加和抑制噪声分两块进行，分别是添加高斯噪声和其他滤波方式以及添加椒盐噪声和其他滤波方式。

1. 添加椒盐噪声：先设置需要设置多大范围的椒盐噪声，之后选取这么多个像素点进行修改。随机生成 1 中数据多个的点，对每一个点随机修改其像素点为 0 或者 255。将其数据返回生成椒盐噪声图。当进行均值滤波时，只需要调用 OpenCV 工具包中的 cv2.blur()函数即可；当进行中值滤波时，只需要调用 OpenCV 工具包中的 cv2.medianBlur()函数即可；当进行高斯滤波时，只需要调用 OpenCV 工具包中的 cv2.GaussianBlur()函数即可。
2. 添加高斯噪声：先将原图像的像素点的灰度值归一化处理，之后生成与原图像同样大小的高斯噪声，将生成的高斯噪声和原图点对点的相加得到含有高斯噪声的图像，但是在该图像中可能存在越界，因此需要判断像素点是否符合要求。最后返回的是含有高斯噪声的图像。当进行均值滤波时，只需要调用 OpenCV 工具包中的 cv2.blur()函数即可；当进行中值滤波时，只需要调用 OpenCV 工具包中的 cv2.medianBlur()函数即可；当进行高斯滤波时，只需要调用 OpenCV 工具包中的 cv2.GaussianBlur()函数即可。

(2)代码编写（包含适量注释）

```
1. import random
2. import cv2
3. import numpy as np
4. from matplotlib import pyplot as plt
5. from pylab import mpl
6. # 设置显示中文字体
7. mpl.rcParams["font.sans-serif"] = ["SimHei"]
8. # 设置正常显示符号
9. # plt.axis('off')
10. # plt.figure(figsize=(5,5))
11. mpl.rcParams["axes.unicode_minus"] = False
12. plt.subplots_adjust(left=None,bottom=None,right=None,top=None,wspace=0.5,h
    space=0.05)  # 设置子图左右和上下间隔，一般为 0.5 为佳
13.
14. def addSaltAndPepper(src, percentage):
15.     # NoiseImg = src    #使用此语句传递的是地址，程序会出错
16.     # 在此要使用 copy 函数，否则 src 和主程序中的 img 都会跟着改变
17.     NoiseImg = src.copy()
18.     NoiseNum = int(percentage * src.shape[0] * src.shape[1])
19.     for i in range(NoiseNum):
20.         # 注意需要引入 random 包
21.         # 产生[0, src.shape[0] - 1]之间随机整数
22.         randX = random.randint(0, src.shape[0] - 1)
23.         randY = random.randint(0, src.shape[1] - 1)
```

```
24.         if random.randint(0, 1) == 0:
25.             NoiseImg[randX, randY] = 0
26.         else:
27.             NoiseImg[randX, randY] = 255
28.     return NoiseImg
29.
30.
31. def addGaussianNoise(src, means, sigma):
32.     image = np.array(src / 255, dtype=float)
33.     noise = np.random.normal(means, sigma, image.shape)
34.     gauss_noise = image + noise
35.     if gauss_noise.min() < 0:
36.         low_clip = -1.
37.     else:
38.         low_clip = 0.
39.     gauss_noise = np.clip(gauss_noise, low_clip, 1.0)
40.     gauss_noise = np.uint8(gauss_noise * 255)
41.     return gauss_noise
42.
43.
44. # 均值滤波
45. def average(img):
46.     return cv2.blur(img, (3, 3))
47.
48.
49. # 中值滤波
50. def middle(img):
51.     return cv2.medianBlur(img, 3)
52.
53.
54. # 高斯滤波
55. def Goss(img):
56.     return cv2.GaussianBlur(img, (3, 3), 1)
57.
58.
59. if __name__ == '__main__':
60.     path = 'work/img/Lenna.png'
61.     img = cv2.cvtColor(addSaltAndPepper(cv2.imread(path), 0.2), cv2.COLOR_
    BGR2RGB)
62.     plt.subplot(241)
63.     plt.imshow(img)
64.     plt.title('添加椒盐噪声',fontsize=10, fontproperties=font)
65.
66.     plt.subplot(242)
```

```
    plt.imshow(average(img))
    plt.title('椒盐噪声的均值滤波',fontsize=10, fontproperties=font)

    plt.subplot(243)
    plt.imshow(middle(img))
    plt.title('椒盐噪声的中值滤波',fontsize=10, fontproperties=font)

    plt.subplot(244)
    plt.imshow(Goss(img))
    plt.title('椒盐噪声的高斯滤波',fontsize=10, fontproperties=font)

    img1 = cv2.cvtColor(addGaussianNoise(cv2.imread(path), 0, 0.1), cv2.CO
LOR_BGR2RGB)
    plt.subplot(245)
    plt.imshow(img1)
    plt.title('添加高斯噪声',fontsize=10, fontproperties=font)

    plt.subplot(246)
    plt.imshow(average(img1))
    plt.title('高斯噪声的均值滤波',fontsize=10, fontproperties=font)

    plt.subplot(247)
    plt.imshow(middle(img1))
    plt.title('高斯噪声的中值滤波',fontsize=10, fontproperties=font)

    plt.subplot(248)
    plt.imshow(Goss(img1))
    plt.title('高斯噪声的高斯滤波',fontsize=10, fontproperties=font)
    plt.show()
```

(3)调试（编码实现过程中遇到的错误，及调试解决等内容）

问题 1：在调试代码的过程中，我遇到如图 20 所示的错误。

```
Traceback (most recent call last):
  File "D:/2020185 _and_10208/Kernel_lessons/数字图像处理/实验2/test.py", line 126, in <module>
    img1 = cv2.cvtColor(addGaussianNoise(cv2.imread(path), 0, 0.1), cv2.COLOR_BGR2RGB)
  File "D:/2020185 _and_10208/Kernel_lessons/数字图像处理/实验2/test.py", line 82, in addGaussianNoise
    if gauss_noise < 0:
ValueError: The truth value of an array with more than one element is ambiguous. Use a.any() or a.all()
```

图 20 问题 1 的报错

解决 1：报错里面指的是，变量是一个 numpy 数组，里面不止一个元素，因此用它和一个数字来比，就会报错。修改意见是将原来的 numpy 中取出一个最小值与 0 相比。其修改后的截图如下图 21 所示。

```
noise = np.random.normal(means, sigma, image.s
gauss_noise = image + noise
if gauss_noise.min() < 0:
    low_clip = -1.
```

图 21 问题 1 的解决方案

(4)实验结果（实验运行结果截图等体现实验结果的内容）

结果 1：对 Lena 图添加椒盐噪声的结果如下图 22 所示。

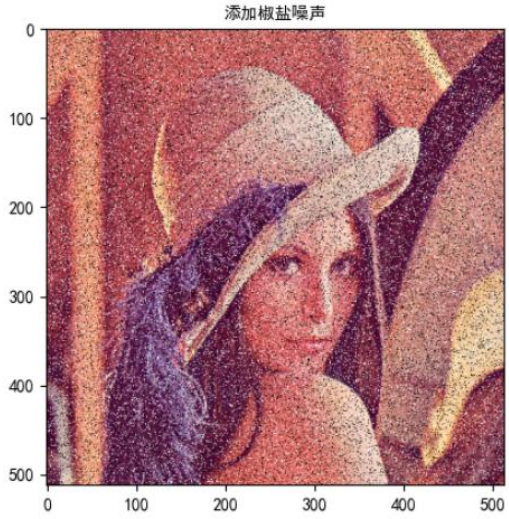

图 22 添加椒盐噪声结果图

结果 2：对 Lena 图添加椒盐噪声后的图像经过均值滤波得到的图像如下图 23 所示。

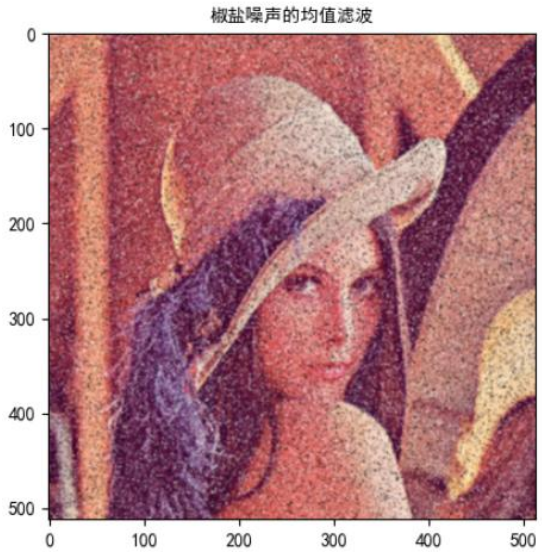

图 23 椒盐噪声的均值滤波

结果 3：对 Lena 图添加椒盐噪声后的图像经过中值滤波得到的图像如下图 24 所示。

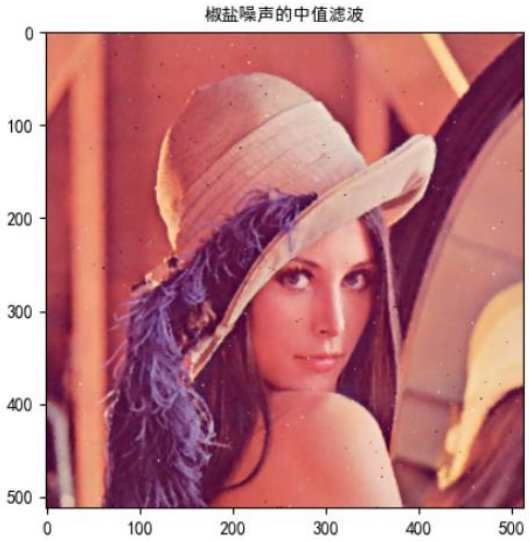

图 24 椒盐噪声的中值滤波

结果 4：对 Lena 图添加椒盐噪声后的图像经过高斯滤波得到的图像如下图 25 所示。

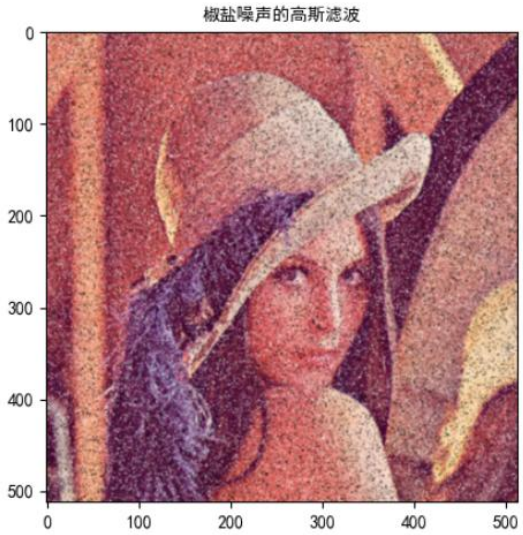

图 25 椒盐噪声的高斯滤波

结果 5：对 Lena 图添加高斯噪声的结果如下图 26 所示。

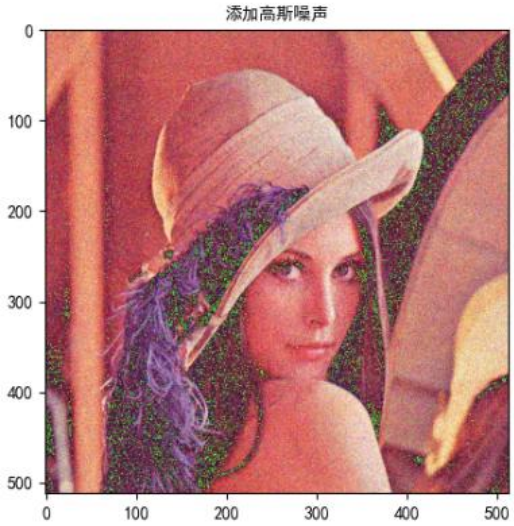

图 26 添加高斯噪声结果图

结果 6：对 Lena 图添加高斯噪声后的图像经过均值滤波得到的图像如下图 27 所示。

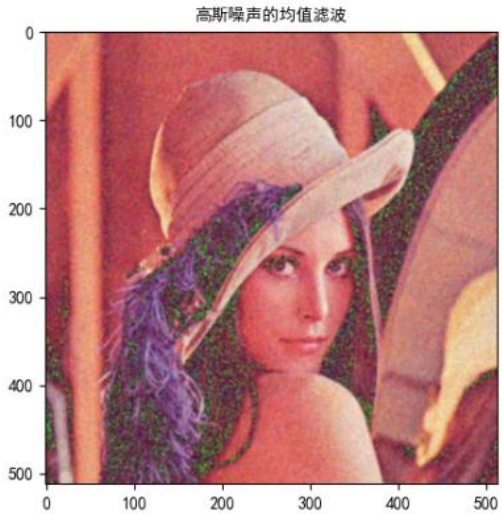

图 27 高斯噪声的均值滤波

结果 7：对 Lena 图添加高斯噪声后的图像经过中值滤波得到的图像如下图 28 所示。

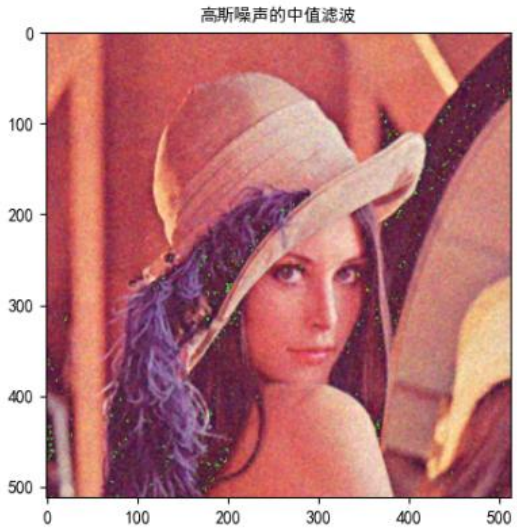

图 28 高斯噪声的中值滤波

结果 8：对 Lena 图添加高斯噪声后的图像经过高斯滤波得到的图像如下图 29 所示。

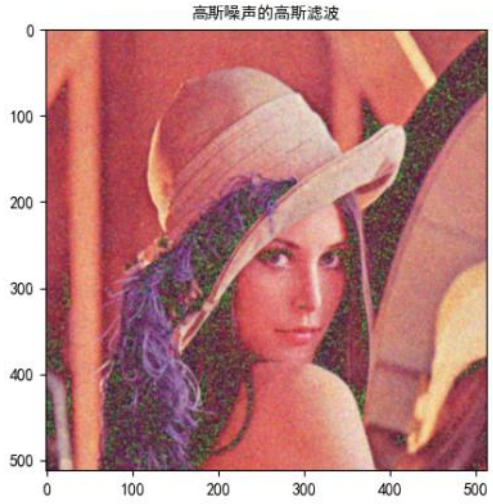

图 29 高斯噪声的高斯滤波

结果 9：结果 1 到结果 8 总览图。

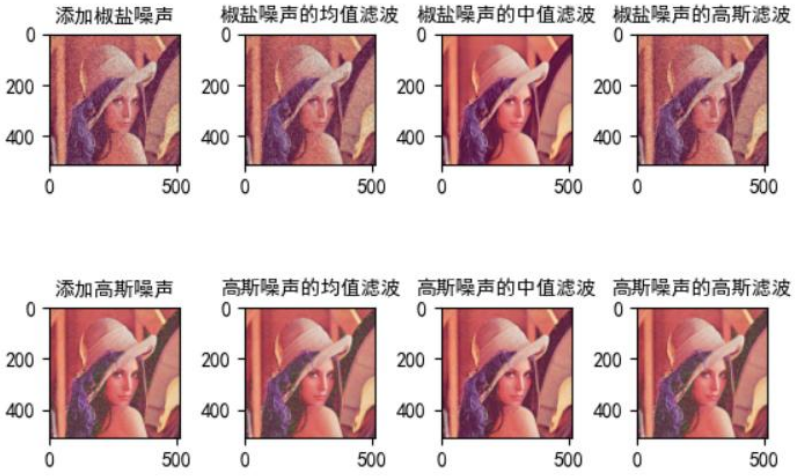

图 30 添加及去除椒盐噪声和高斯噪声的结果总览

(5)分析与总结（与本次实验相关的分析与总结）

通过本次实验，我明白了椒盐噪声和高斯噪声生成的原理，以及均值滤波、中值滤波和高斯滤波的使用方法。

均值滤波是典型的线性滤波算法，它是指在图像上对目标像素给一个模板，该模板包括了其周围的临近像素，再用模板中的全体像素的平均值来代替原来像素值。

中值滤波法是一种非线性平滑技术，它将每一像素点的灰度值设置为该点某邻域窗口内的所有像素点灰度值的中值。

斯滤波就是对整幅图像进行加权平均的过程，每一个像素点的值，都由其本身和邻域内的其他像素值经过加权平均后得到。高斯滤波的具体操作是：用一个模板扫描图像中的每一个像素，用模板确定的邻域内像素的加权平均灰度值去替代模板中心像素点的值。

所谓椒盐，椒就是黑，盐就是白，椒盐噪声就是在图像上随机出现黑色白色的像素。由添加椒盐噪声的实验可以看出椒盐噪声是一种因为信号脉冲强度（0和1）引起的噪声。此外，从结果2到结果4的实验结果的效果，明显的看出去除椒盐噪声的最好办法就是对其进行中值滤波。

高斯噪声是指它的概率密度函数服从高斯分布（即正态分布）的一类噪声。从结果6到结果8的实验结果的效果明显的可以看出去除高斯滤波的最好办法就是对其进行中值滤波。

4. 对本次实验的总结：

通过本实验，我对灰度变换的操作越发熟练了：当利用 Opencv 读取图像时，只需要在读取函数中加一个 0 参数，读取的就是一副灰度图像。

对图像做点运算，就是对图像中的每个像素依次进行同样的灰度变换运算。其是指就是对图像的各个像素点按照某一种相同的策略利用点运算算子将原始图像和数度图像建立某种灰度级映射的关系。这种关系在实验1中具体体现在同时加某一个灰度或减同一个灰度，或者是等比例的扩大或缩小。

空域平滑的实现方法将原图中的每一个点的灰度与它周围点的灰度进行加权和平均，作为新图中对应点的灰度，就能实现滤波的效果。若噪声是随机独立分布的，利用邻域平均或加权平均可以有效抑制噪声干扰。

拉普拉斯是一种微分算子，拉普拉斯图像强调原图中的灰度突变区域，衰减灰度变化慢区域，恒定区域变为0。其具体操作为：当邻域的中心像素灰度低于它所在邻域内的其他像素的平均灰度时，此中心像素的灰度应该进一步降低；当高于时进一步提高中心像素的灰度，从而实现图像锐化处理。

关于点运算和领域运算的差别与联系主要表现为：点运算包括：腐蚀、膨胀、开

运算、闭运算，对像素点的单独计算，不涉及到领域像素；领域运算：均值滤波器、高斯滤波器、中值滤波器、锐化滤波器等。计算像素点时涉及到领域像素。但是二者都能通过改变像素值从而达到改变像素的效果。