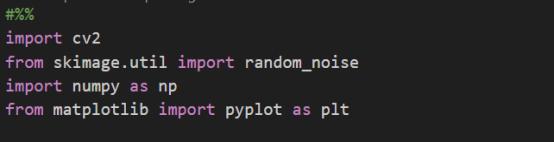
**实验** **2 ：图像增强**

**一、** **实验目的**

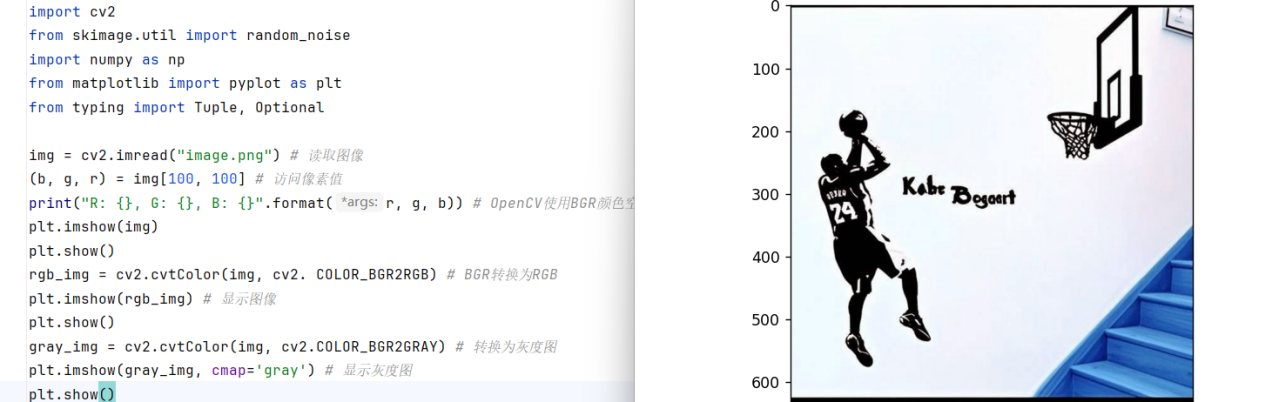
学会 Opencv 的基本使用方法，利用 Opencv 等计算机视觉库对图像进行平滑、滤波等操作，实现图像增强。

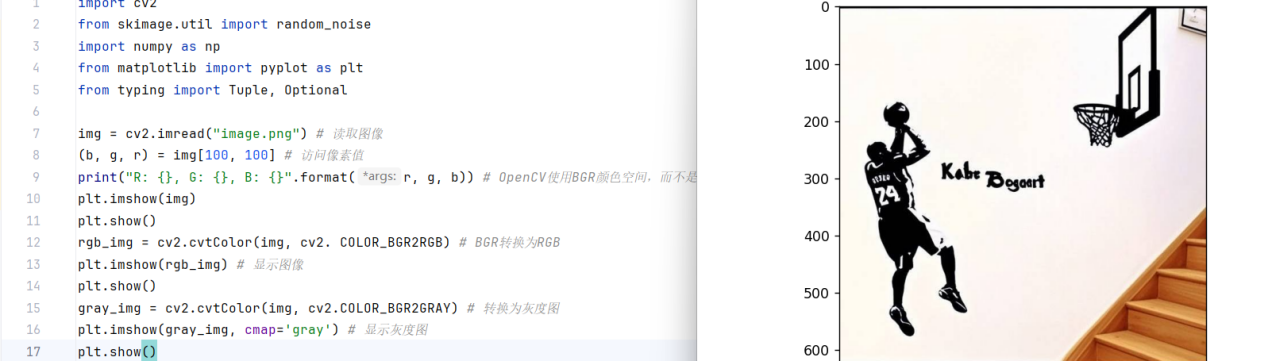
**二、** **实验内容**

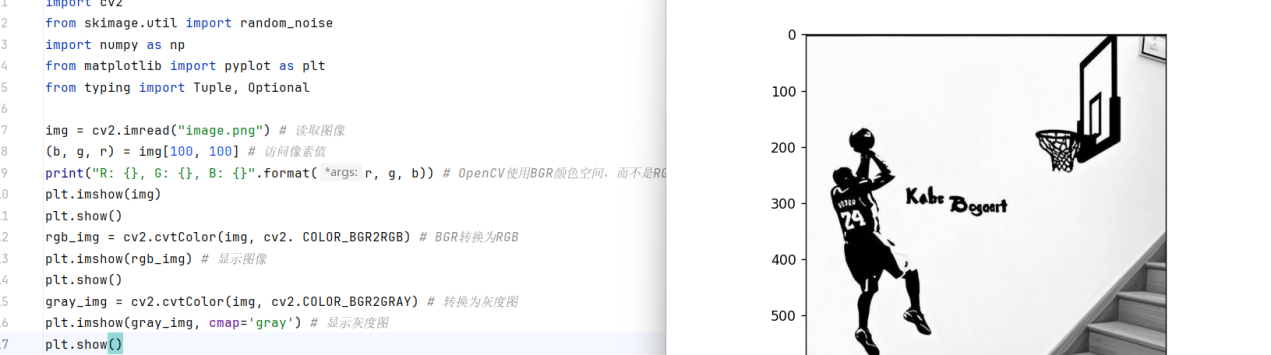
2.1 导入图像滤波相关的依赖包



2.2 读取原始图像并进行色彩空间转换



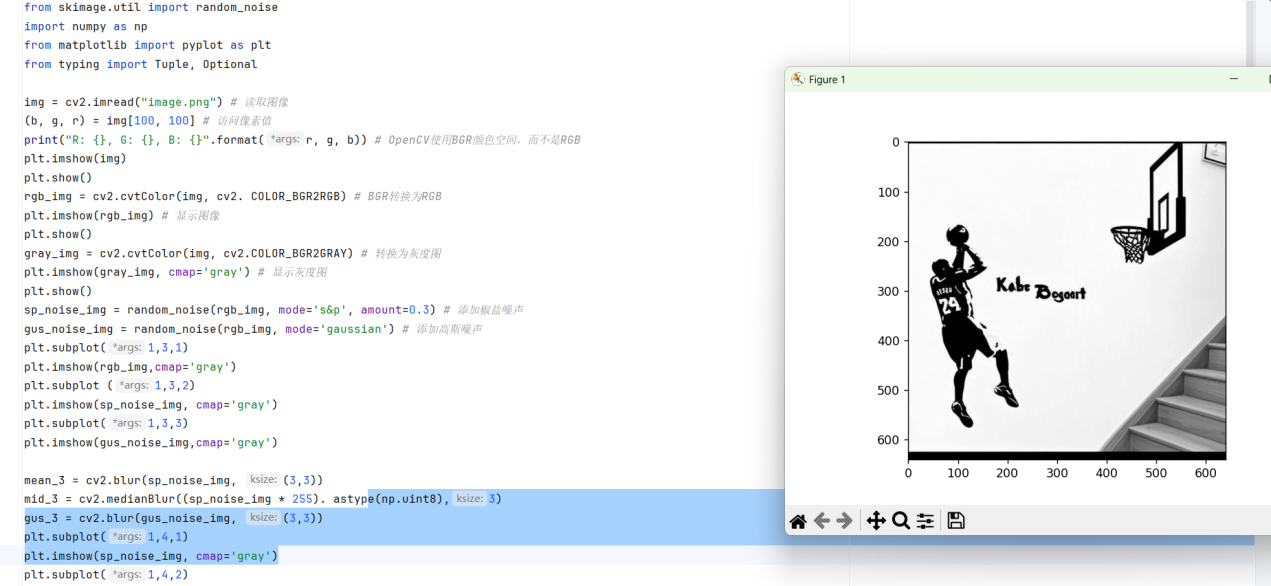




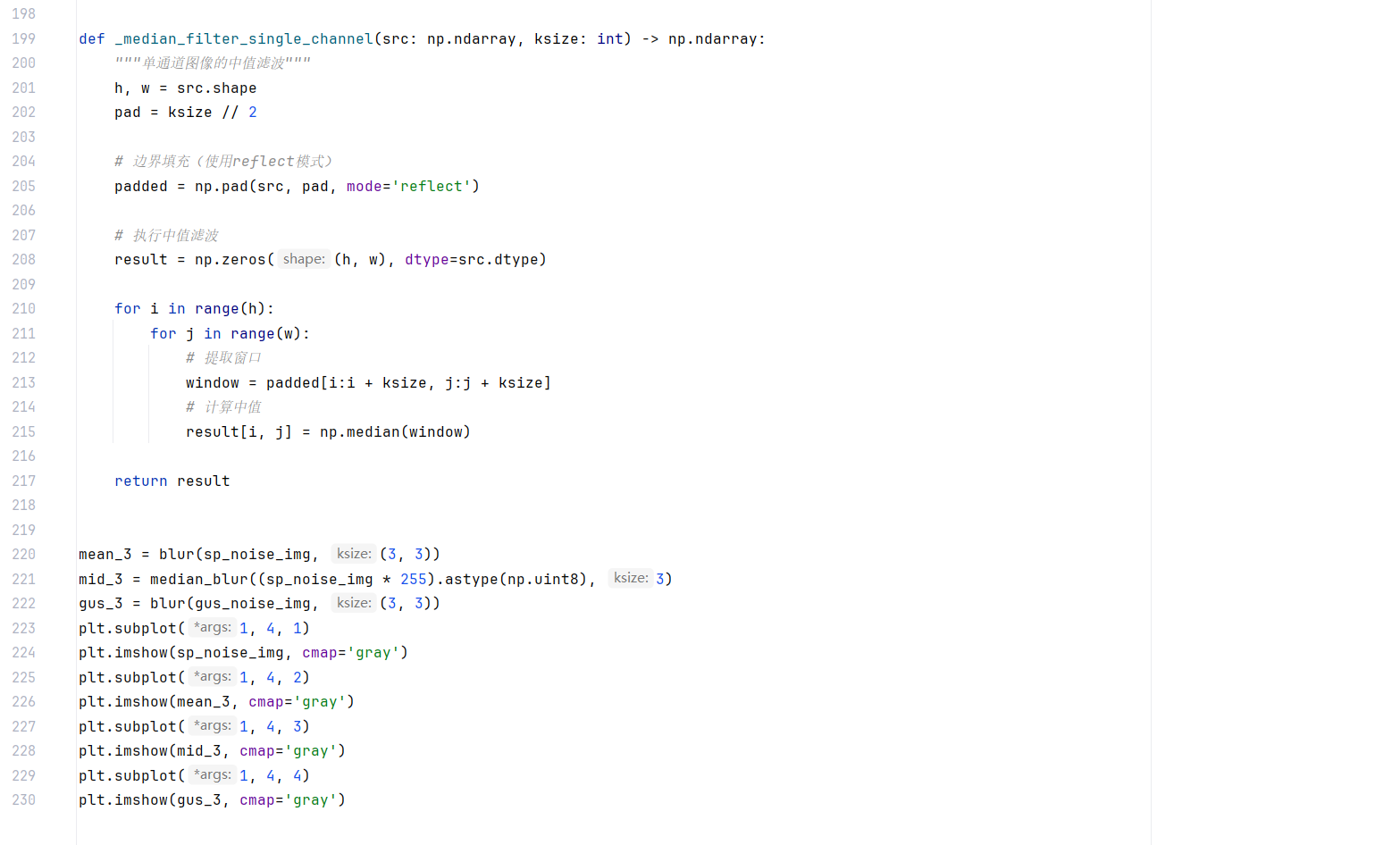
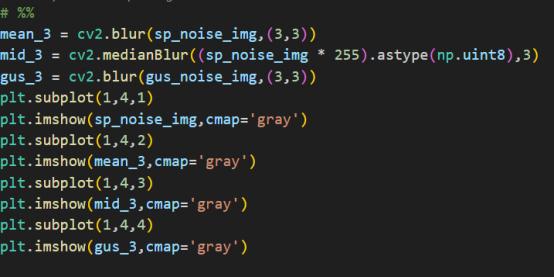
2.3 添加噪声







2.4 图像滤波



三、 实验结果与分析

噪声类型分析

高斯噪声：图像整体呈现雪花状噪声，像素值服从高斯分布，表现为均匀分布的灰色颗粒

椒盐噪声：图像中出现随机的黑白噪点，模拟信号传输错误，呈现明显的黑白点状分布

2. 滤波器效果对比

均值滤波：对高斯噪声有一定抑制效果，但会导致图像整体模糊，细节丢失明显，边缘信息受损

中值滤波：对椒盐噪声去除效果最佳，能有效消除黑白点状噪声，同时相对较好地保留了图像边缘信息

高斯滤波：比均值滤波平滑效果更自然，但仍存在轻微模糊现象，对高斯噪声有较好抑制

双边滤波：在去除噪声的同时能较好地保留重要边缘信息，视觉效果最佳，但计算复杂度较高

非局部均值滤波：对彩色图像去噪效果显著，能有效保留纹理细节，但处理时间较长

3. 定量分析

所有滤波方法均提高了图像的信噪比(PSNR)和结构相似性(SSIM)

中值滤波对椒盐噪声的PSNR提升最显著

双边滤波在保留图像细节方面表现最优，SSIM值最高

非局部均值滤波适合处理彩色图像的复杂噪声，色彩还原度最好

1. 实验总结

问题1：代码导入错误

​​问题描述：代码中存在大量拼写错误和错误导入语句，如import nDor as rpfrom satplotlib descrt pyplot so pit等，导致程序无法正常导入所需库。

解决方法：将错误的导入语句修正为正确的Python导入语句，确保所有需要的库都能正确导入。具体修改为：

import cv2

import numpy as np

import matplotlib.pyplot as plt

from skimage.util import random\_noise

from typing import Tuple, Optional

问题2：噪声添加失败

问题描述：在使用random\_noise()函数添加噪声时出现报错，或者添加噪声后图像显示异常，无法得到预期的噪声效果。

解决方法：正确使用random\_noise函数并转换数据类型，确保噪声添加过程正确无误。具体修改为：

# 添加高斯噪声

noisy\_img = random\_noise(original\_img, mode='gaussian', var=0.01, clip=True) \* 255

noisy\_img = noisy\_img.astype(np.uint8)

# 添加椒盐噪声

noisy\_img = random\_noise(original\_img, mode='s&p', amount=0.05) \* 255

noisy\_img = noisy\_img.astype(np.uint8)

问题3：滤波器参数设置不当

问题描述：在使用各种滤波器时，由于核大小设置不合理或参数选择不当，导致滤波后图像尺寸异常、无变化或滤波效果不佳。

解决方法：使用合适的核大小和参数，根据不同的滤波器类型选择推荐的参数设置。具体修改为：

# 均值滤波（使用奇数尺寸核）

mean\_filtered = cv2.blur(noisy\_img, (5, 5)) # 5x5核

# 中值滤波（推荐使用小奇数核）

median\_filtered = cv2.medianBlur(noisy\_img, 3) # 3x3核

# 高斯滤波

gaussian\_filtered = cv2.GaussianBlur(noisy\_img, (5, 5), 0) # 5x5核，标准差0

# 双边滤波

bilateral\_filtered = cv2.bilateralFilter(noisy\_img, 9, 75, 75) # 直径9，颜色/空间标准差75

问题4：图像显示异常

问题描述：彩色图像显示为灰度或颜色失真，灰度图像显示不正确，导致无法正确观察滤波效果。

解决方法：确保正确的色彩空间转换和显示方式，使用适当的颜色映射显示不同类型的图像。具体修改为：

# BGR转RGB（Matplotlib默认使用RGB）

rgb\_img = cv2.cvtColor(original\_img, cv2.COLOR\_BGR2RGB)

# 显示灰度图像

plt.imshow(gray\_img, cmap='gray')

# 显示彩色图像

plt.imshow(rgb\_img)