作业一

一、编写程序，对一副彩色图像进行灰度化、二值化、均值化操作。

### 1. 灰度化原理

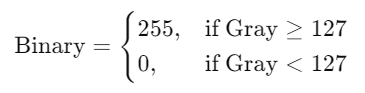
彩色图像通常是由红色（R）、绿色（G）、蓝色（B）三个通道组成的。在灰度图像中，这些颜色信息被转换为单个强度值。通常的转换公式是：

Gray=0.299×R+0.587×G+0.114×B

这个公式根据人眼对不同颜色的敏感度来加权。用这个公式可以将每个像素的 RGB 值转换为灰度值。

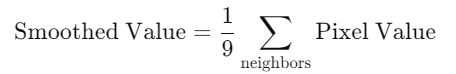
### 2. 二值化原理

二值化是将灰度图像中的像素根据阈值分为两类：将灰度值高于某个阈值的像素设为白色（1或255），低于阈值的像素设为黑色（0）。常见的选择是使用一个中间值作为阈值：



### 3. 均值化原理

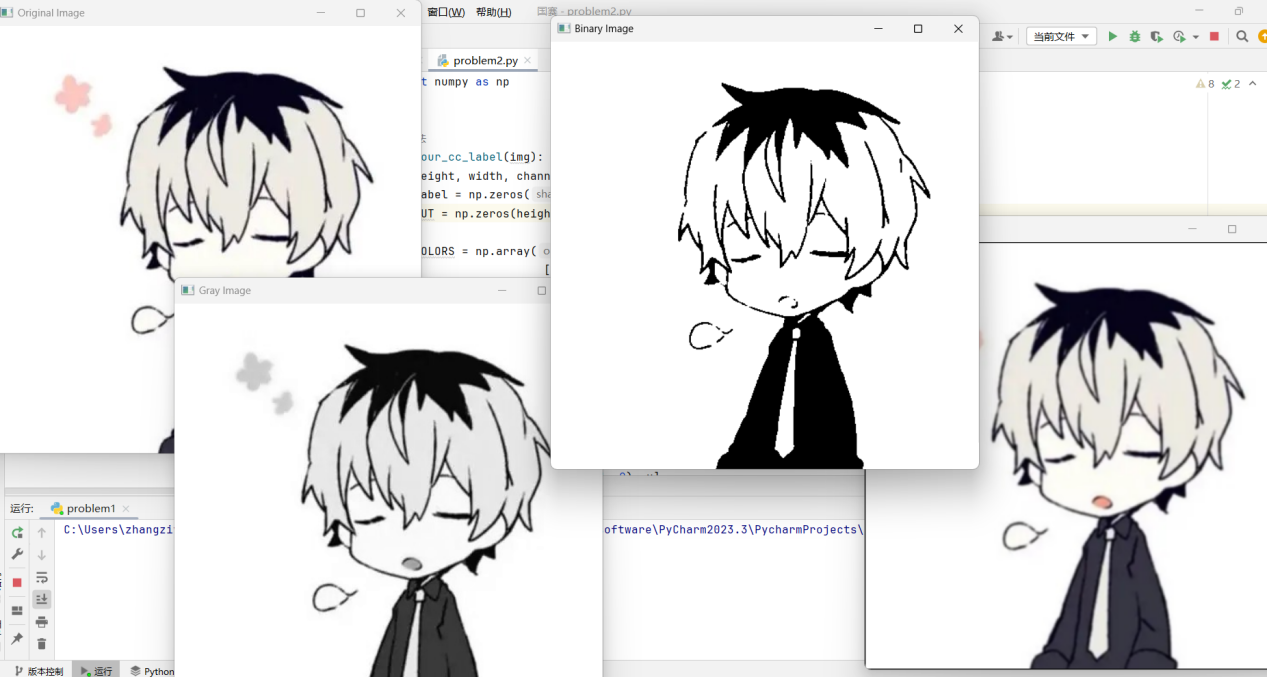
均值化（或平滑化）是通过对像素及其邻居的平均值来减少图像中的噪声，常见的做法是使用卷积操作。对每个像素，取它周围像素的平均值作为新的像素值。比如使用 3×3的邻域，公式为：



代码：

import numpy as np  
import cv2  
  
*# 读取彩色图像（为 RGB 格式）*image = cv2.imread('图片一.png')  
  
*# 获取图像的高度和宽度*height, width = image.shape[:2]  
  
*# 1. 灰度化操作*gray\_image = np.zeros((height, width), dtype=np.uint8)  
for i in range(height):  
 for j in range(width):  
 r, g, b = image[i, j]  
 *# 按照公式计算灰度值* gray\_value = 0.299 \* r + 0.587 \* g + 0.114 \* b  
 gray\_image[i, j] = int(gray\_value)  
  
*# 2. 二值化操作*binary\_image = np.zeros((height, width), dtype=np.uint8)  
threshold = 127  
for i in range(height):  
 for j in range(width):  
 *# 根据灰度值进行二值化* binary\_image[i, j] = 255 if gray\_image[i, j] >= threshold else 0  
  
*# 3. 均值化操作 (简单平均滤波)*smoothed\_image = np.zeros\_like(image, dtype=np.uint8)  
kernel\_size = 3  
offset = kernel\_size // 2  
  
for i in range(offset, height - offset):  
 for j in range(offset, width - offset):  
 *# 获取邻域* region = image[i-offset:i+offset+1, j-offset:j+offset+1]  
 *# 计算每个通道的均值* r\_mean = np.mean(region[:, :, 0])  
 g\_mean = np.mean(region[:, :, 1])  
 b\_mean = np.mean(region[:, :, 2])  
 *# 更新图像像素值* smoothed\_image[i, j] = [r\_mean, g\_mean, b\_mean]  
  
*# 显示结果*cv2.imshow('Original Image', image)  
cv2.imshow('Gray Image', gray\_image)  
cv2.imshow('Binary Image', binary\_image)  
cv2.imshow('Smoothed Image', smoothed\_image)  
  
cv2.waitKey(0)  
cv2.destroyAllWindows()

结果：



## 二、编写程序，对图像进行4邻域连通区域标记操作。

首先将二值矩阵进行 4-邻域的连通区域标记，得到标记矩阵。然后显示结果，并为每个像素块添加网格线和其所属连通区域的编号，最终保存为图像文件。

### 1.输入矩阵

二值矩阵：这个矩阵表示图像中的黑白像素，其中 1 表示白色（用255表示），0 表示黑色（用0表示）。

乘以255：cv2.connectedComponents 处理的是灰度图像，像素值必须是0和255，所以将 1 变成 255，保持 0 为黑色。

### 2. 连通区域标记

* **cv2.connectedComponents()：这是一个核心函数，用于计算图像中的 连通区域。**
* binary\_image：输入二值图像，0表示背景，255表示前景。
* connectivity= 4：表示我们使用 4-邻域，即水平方向和竖直方向上的相邻像素会被认为是连通的。

**返回值：**

* num\_labels：表示图像中不同连通区域的数量（包括背景）。
* labels：这是一个与输入图像大小相同的矩阵，标记了每个像素属于哪个连通区域。像素值为相应连通区域的编号。

### 添加序号

for i in range(rows):

for j in range(cols):

label\_value = labels[i, j]

plt.text(j, i, str(label\_value), ha='center', va='center', color='white', fontsize=12)

* **嵌套循环遍历图像**：双层循环遍历 labels 矩阵中的每个元素，i 是行索引，j 是列索引。
* plt.text()：在每个像素格子内居中显示其对应的连通区域编号。

### 绘图

plt.imshow()：使用 matplotlib 显示连通区域标记后的矩阵图像。

plt.xticks() **和** plt.yticks()：调整刻度，使得网格线与像素格子完全对齐，隐藏默认的刻度标签。

np.arange(-0.5, cols, 1)：设置网格的间隔，使得网格线划分与图像的像素边界一致。

代码：

import cv2  
import numpy as np  
import matplotlib.pyplot as plt  
  
*# 二值矩阵 (0 表示黑，1 表示白)*binary\_image = np.array([  
 [1, 1, 0, 1, 1, 1, 0, 1],  
 [1, 1, 0, 1, 0, 1, 0, 1],  
 [1, 1, 1, 1, 0, 0, 0, 1],  
 [0, 0, 0, 0, 0, 0, 0, 1],  
 [1, 1, 1, 1, 0, 1, 0, 1],  
 [0, 0, 0, 1, 0, 1, 0, 1],  
 [1, 1, 1, 1, 0, 0, 0, 1],  
 [1, 1, 1, 1, 0, 1, 1, 1]  
], dtype=np.uint8) \* 255 *# 将 1 转换为 255（白），0 保持 0（黑）  
  
# 使用 cv2.connectedComponents 进行连通区域标记（4-邻域）*num\_labels, labels = cv2.connectedComponents(binary\_image, connectivity=4)  
  
*# 显示结果并添加网格线和编号*plt.figure(figsize=(10, 5))  
  
*# 绘制标记后的连通区域图像*plt.imshow(labels, cmap='nipy\_spectral', interpolation='none')  
  
*# 添加网格线*plt.grid(which='both', color='black', linestyle='-', linewidth=2)  
  
*# 获取图像大小*rows, cols = labels.shape  
  
*# 添加每个小格子的序号*for i in range(rows):  
 for j in range(cols):  
 *# 获取当前像素的连通标记* label\_value = labels[i, j]  
 plt.text(j, i, str(label\_value), ha='center', va='center', color='white', fontsize=12)  
  
*# 设置刻度以适应网格线*plt.xticks(np.arange(-0.5, cols, 1), [])  
plt.yticks(np.arange(-0.5, rows, 1), [])  
  
*# 设置网格线*plt.gca().set\_xticks(np.arange(-0.5, cols, 1), minor=True)  
plt.gca().set\_yticks(np.arange(-0.5, rows, 1), minor=True)  
plt.gca().grid(which='minor', color='black', linestyle='-', linewidth=2)  
  
plt.show()  
  
*# 保存结果图像*cv2.imwrite('labeled\_image\_with\_grid.png', (labels \* (255 // num\_labels)).astype(np.uint8))

结果：

