《数字图像处理》实验报告

时间: 2021 年 10 月 16 日（第 7 周，星期 六 ）

地点： 测绘工程实验室（B 区）-- B 实验大楼 202

学生姓名： 魏子继 学号： 20194947

一、 实验名称：

图像二值化

二、 实验目的：

两个目标：

1. 图像灰度化
2. 将灰度化的图像二值化

要求中，未对灰度化方法做出具体要求；二值化的阈值大小也是自己给定的，本实验中，使用 127 为正反二值化图像阈值。

三、 实验步骤：

根据实验目的和实现方法，该实验由以下三步完成：

1. 将图像灰度化。即根据三通道图像的值，依据一定的公式，计算出图像的灰度值，将图像转换为灰度图像。其中，有许多的灰度化方法，这里采用四种灰度化方法作为比较，分别为：opencv 自带的灰度化函数、采用最大值作为灰度值灰度化、采用平均值作为灰度值灰度化、使用 Gamma 校正灰度化

2. 将图像二值化。设定阈值，若图像像素灰度值大于阈值，则将像素灰度值设置为 255，即全白；若图像像素灰度值小于阈值，则将像素灰度值设置为 0，即全黑。
3. 图像反二值化则使用与图像二值化算法相反的步骤。设定阈值，若图像像素灰度值大于阈值，则将像素灰度值设置为 0，即全黑；若图像像素灰度值小于阈值，则将像素灰度值设置为 255，即全白。

四、 实验中的关键点分析（包括关键算法与代码实现）：

1. 图像灰度化（GrayImage.py）

```
"""
Created by Chloe on 2021/10/26
"""

import cv2.cv2 as cv2
import numpy as np
import matplotlib.pyplot as plt

plt.rcParams['font.sans-serif'] = ['SimHei']


# opencv 内置函数灰度化，即加权平均
def gray_function(img):
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)  # COLOR_BGR2GRAY 参数表示 BGR 图像转化为灰度图像
    gray = cv2.cvtColor(gray, cv2.COLOR_BGR2RGB)  # COLOR_BGR2RGB 参数表示 BGR 图像转化为 RGB 图像以在 plt 中显示
    return gray


# 使用最大值灰度化
def gray_Max(img):
    gray = np.zeros((height, width), img.dtype)  # 设置一个新的图像存放灰度图像，防止对原图进行修改
    for i in range(height):
```

```
24.         for j in range(width):
25.             gray[i, j] = max(img[i, j, 0], img[i, j, 1], img[i, j, 2])  # 最大值设置为灰度
26.     gray = cv2.cvtColor(gray, cv2.COLOR_BGR2RGB)
27.     return gray
28.
29.
30. # 使用平均值灰度化
31. def gray_Average(img):
32.     gray = np.zeros((height, width), img.dtype)
33.     for i in range(height):
34.         for j in range(width):
35.             gray[i, j] = (int(img[i, j, 0]) + int(img[i, j, 1]) + int(img[i, j, 2])) / 3  # 平均值作为灰度
36.     gray = cv2.cvtColor(gray, cv2.COLOR_BGR2RGB)
37.     return gray
38.
39.
40. # 使用 Gamma 校正进行灰度处理
41. def gray_Gamma(img):
42.     gray = np.zeros((height, width), img.dtype)
43.     for i in range(height):
44.         for j in range(width):
45.             a = img[i, j, 2] ** 2.2 + 1.5 * img[i, j, 1] ** 2.2 + 0.6 * img[i, j, 0] ** 2.2  # 分子
46.             b = 1 + 1.5 ** 2.2 + 0.6 ** 2.2  # 分母
47.             gray[i, j] = pow(a / b, 1.0 / 2.2)  # 开 2.2 次方根，即 Gamma 矫正灰度处理
48.     gray = cv2.cvtColor(gray, cv2.COLOR_BGR2RGB)
49.     return gray
50.
51.
52. # 主程序
53. if __name__ == '__main__':
54.     # 获取图像
55.     image = cv2.imread("D:\\pyfiles\\DigitalImageProcessing\\image\\sylm1.jpeg")
56.
57.     # 获取图像尺寸
58.     height, width = image.shape[0:2]
59.
60.     # 将图像灰度化
61.     gray_func = gray_function(image)
62.     gray_max = gray_Max(image)
```

```
    gray_ave = gray_Average(image)
    gray_gamma = gray_Gamma(image)

    # 显示图像
    titles = ['cv2.cvtColor', '最大值灰度化', '平均值灰度化', 'Gamma 校正灰度化']  # 标题
    images = [gray_func, gray_max, gray_ave, gray_gamma]  # 图像对比显示
    for i in range(4):      # 使用 matplotlib 绘图
        plt.subplot(1, 4, i + 1)
        plt.imshow(images[i])
        plt.title(titles[i])
        plt.axis('off')  # 关闭坐标轴
    plt.savefig("D:\\pyfiles\\DigitalImageProcessing\\result\\GrayImage.png", bbox_inches='tight')  # 文件保存
    plt.show()  # 图像展示
```

76. 图像二值化（BinaryImage.py）

```
"""
Created by Chloe on 2021/10/26
"""

import cv2.cv2 as cv2
import numpy as np
import matplotlib.pyplot as plt
import GrayImage as Gray

plt.rcParams['font.sans-serif'] = ['SimHei']


# 使用 opencv 算法求解全局阈值，作为参照
def threshold_Ref(img):
    gray = Gray.gray_function(img)  # 转为灰度图像，使用 opencv 自带算法
    threshold, binary = cv2.threshold(gray, 127, 255, cv2.THRESH_BINARY)  # 设 127 为阈值
    return binary


# 自定义二进制阈值，生成二值图像，大于阈值图像设置为 255
def threshold_self(img, threshold):  # threshold 为输入的阈值
    gray = cv2.cvtColor(Gray.gray_function(img), cv2.COLOR_BGR2GRAY)    # 灰度化
```

```
    binary = np.zeros((height, width), gray.dtype)  # 空白图像存放二值图像，避免对原图修改
    for x in range(height):
        for y in range(width):
            # 判断是否阈值，大于阈值设置为255(全白)，否则设置为0(全黑)
            binary[x, y] = 255 if gray[x, y] >= threshold else 0
    binary = cv2.cvtColor(binary, cv2.COLOR_BGR2RGB)
    return binary


# 自定义反二进制阈值，生成二值图像，大于阈值图像设置为0
def threshold_self_inv(img, threshold):
    gray = cv2.cvtColor(Gray.gray_function(img), cv2.COLOR_BGR2GRAY)
    binary = np.zeros((height, width), gray.dtype)
    for x in range(height):
        for y in range(width):
            # 判断是否阈值，大于阈值设置为0(全黑)，否则设置为255(全白)
            binary[x, y] = 0 if gray[x, y] >= threshold else 255
    binary = cv2.cvtColor(binary, cv2.COLOR_BGR2RGB)
    return binary


# 主程序
if __name__ == '__main__':
    # 读入图片
    image = cv2.imread("D:\\pyfiles\\DigitalImageProcessing\\image\\sylm1.jpeg")

    # 获取图像尺寸
    height, width = image.shape[0:2]

    # 二值化图像
    binary_ref = threshold_Ref(image)
    binary_self = threshold_self(image, 127)
    binary_self_inv = threshold_self_inv(image, 127)

    # 显示图像
    titles = ['cv2.threshold', '自定义二值图像', '自定义反二值图像']  # 标题
    images = [binary_ref, binary_self, binary_self_inv]  # 图像对比显示
    for i in range(3):  # 使用matplotlib绘图
        plt.subplot(1, 3, i + 1)
        plt.imshow(images[i])
        plt.title(titles[i])
```

```
140.         plt.axis('off')  # 关闭坐标轴
141.     plt.savefig("D:\\pyfiles\\DigitalImageProcessing\\result\\BinaryImage
    .png", bbox_inches='tight')
142.     plt.show()
```

五、 实验原始数据与实验结果：

源图像：（最爱的十元）

结果图像：

1. 灰度化图像

2. 二值化图像

六、 问题分析与心得体会：

在实验过程中，主要遇到了以下问题并做出总结：

1. 在用人眼辨别时，能够看出自己完成的代码与 opencv 内置的灰度化、二值化方法得出的图像相差不大，因此自己完成的代码有一定的可靠性。
2. 该算法的主要问题出现在对图像的像素进行遍历时。在利用 Python 对图像像素遍历时，使用了双 for 循环（for 循环中还嵌套着一个 for 循环）的方法，但是发现这样速度非常地慢，完成灰度化算法的时间甚至需要 3-4 分钟。查阅资料后，采用以下方法对算法进行改进：
   - 将二维数组转化为一维数组，使用了 numpy 库中的 reshape 函数，但是由于选择图像的分辨率太高，numpy 中的数组超限了，因此这个方法失败了
   - 将代码由 CPU 转至 GPU 中运行，使用 numda 库中 jit 模块对代码加速。即在每个使用了双 for 循环的 def 前使用

@jit(nopython=’True’)，但很遗憾，由于对此库之前没有了解过，因此由于未知原因，这个方法还是失败了。

- 已知 cv2 库读入的图像就已是一个 numpy 数组，因此可利用 numpy 数组的广播性对图像处理。即直接对图像像素进行灰度\二值化操作，numpy 数组会自动对所有的像素处理。但考虑到这个方法只能在原图上进行操作，因此还是放弃了

3. 在编写二值化算法时，选择利用之前编写灰度化图像（作为库导入）的结果作为图像处理的后续，但输出的图像一直都是全黑的，于是一直以为是算法设计的问题，不断地对二值化过程算法进行修改，但一直都无法正确处理。后来我查看了输入的灰度图像（PyCharm 能够查看作为图像的数组，这个功能对我的代码书写提供了很大的帮助），发现在之前灰度化函数时，为打印图像方便，已经将图像格式转化为了 RGB 格式，而非灰度化图像格式，因此他无法被二值化算法输入。在这一点上我浪费了很多时间，但也算是一个提醒，在代码书写和算法设计的过程应该像实地测量一样步步检核，对算法和代码的每一步均有一个较为清晰的把握，从而减少一些不该发生的错误发生。