浙江大学

计算机视觉(本科)作业报告

作业名称：	HW#2: 椭圆拟合
姓　　名：	高铭健
学　　号：	3210102322
电子邮箱：	1473760386@qq.com
联系电话：	19550212596
指导老师：	宋明黎

2022 年 12 月 10 日

1. 实验实现的功能简述及运行说明

1.1 实验实现功能

调用CvBox2D cvFitEllipse2(const CvArr* points)实现椭圆拟合

1.2 运行说明

- 最终的可执行文件在exe文件夹中

  在命令行中输入：`path to 3210102322_高铭健\exe\main.exe`，`path to 3210102322_高铭健`表示到当前提交文件夹的路径（例如 D:\study\CV\3210102322_高铭健\exe\main.exe）即可运行程序

- 运行可执行文件后会处理文件夹根目录下的所有文件名以`img`开头的`.jpg`图片

- 生成的图片在当前文件夹根目录下，包括：

  第一张图片拟合后并与原图重叠的结果：output_image_overlap_0.jpg

  第二张图片拟合后并与原图重叠的结果：output_image_overlap_1.jpg

  第一张图片拟合后的结果：output_image_fit_0.jpg

  第二张图片拟合后的结果：output_image_fit_1.jpg

- 可以在`reslut`文件夹中直接查看生成好的拟合图片

2.作业的开发与运行环境

- win11 系统
- 使用python + Opencv库

3.主要用到的函数与算法

3.1 颜色空间转换

`cv2.cvtColor(src, code[, dst[, dstCn]])`是 OpenCV 库的一个函数，可以进行图片颜色空间的转换

其主要参数为：

- `src`: 输入的图像
- `code`: 转换代码，指定了原图和目标图像之间的关系，可以选择`cv2.COLOR_BGR2GRAY`: 转换为灰度图像等

3.2 高斯模糊

`cv2.GaussianBlur(src, ksize, sigmaX[, dst[, sigmaY[, borderType]]])`函数可以对图像进行高斯模糊处理，降低图像中噪声

其主要参数为：

- `src`: 输入的图像
- `ksize`: 模糊核的大小（只能为奇数），它用来改变模糊的程度
- `sigmaX`/Y: X /Y方向上的标准差，可以确定函数处理在水平方向上的权重分布

3.3 边缘处理

`cv2.Canny(image, threshold1, threshold2[, edges[, apertureSize[, L2gradient]]])`函数执行 Canny 边缘检测算法来对图像进行边缘处理

其主要参数为：

- `image`: 输入的图像
- `threshold1` 和 `threshold2`: 非极大值抑制阈值，可以改变边缘检测的灵敏度。阈值较低检测效果更强，而阈值较高可以连接弱边缘以形成完整的边界

3.4 二值化函数

`cv2.threshold(src, thresh, maxval, type[, dst])` 函数对图像进行二值化处理，将图像转换为只有两个像素值（黑和白）的图像，二值化后更容易进行形状识别

其主要参数为：

- `src`: 输入的图像（通常为灰度图像）
- `threshold1` 和 `threshold2`: 非极大值抑制阈值，可以改变边缘检测的灵敏度。阈值较低检测效果更强，而阈值较高可以连接弱边缘以形成完整的边界
- `thresh`： 一个像素值的阈值
- `maxval`： 当图像的像素值大于等于等于阈值时，这些像素被赋予的值，通常情况下是 255
- `type`: 阈值处理类型，决定将像素与阈值比较的方式，比如`cv2.THRESH_BINARY`（如果像素值大于阈值，则设置为 `maxval`，否则为 0）

3.5 提取轮廓

`cv2.findContours(image, mode, method[, contours[, hierarchy[, offset]]])`函数将图像视为一系列连续的点，并返回表示组成轮廓的数据结构。

其主要参数为：

- `src`: 输入的图像输入的图像（通常为二值化图像）
- `mode`: 轮廓检索模式，常用的模式有：
  - `cv2.RETR_EXTERNAL`: 返回最外部轮廓
  - `cv2.RETR_LIST`: 返回所有轮廓
  - `cv2.RETR_TREE`: 返回所有轮廓以及之间层级关系

- `method`：轮廓逼近方法，可以对检测到的轮廓进行近似，减少其数量。常用方法有：
  - `cv2.CHAIN_APPROX_NONE`：保留所有轮廓点
  - `cv2.CHAIN_APPROX_SIMPLE`：仅保留端点。

3.6 椭圆拟合函数

`cv2.fitEllipse(points)`用于将一组点拟合成椭圆形状

其主要参数为：

- `points`：要拟合椭圆的点集（通常是通过 `cv2.findContours()` 等函数获得）

返回值表示拟合的椭圆信息(center, axes, angle)，其中：

- `center`：椭圆的中心坐标
- `axes`：椭圆的长轴和短轴的长度，是一个元组 `(majoraxis, minoraxis)`
- `angle`：椭圆相对于水平轴的旋转角度

3.7 腐蚀函数

`cv2.erode(src, kernel, iterations)` 可以腐蚀图像，去除图像中的小狭窄的连接（对于相邻的形状识别有很大帮助）

其主要参数为：

- `src`：输入图像
- `kernel`：腐蚀操作的内核，可以用不同的形状和大小
- `iterations`：腐蚀操作的迭代次数

4. 实验步骤及代码具体实现

3.1 编写代码。

- 将用到的`opencv`等库导入进来：

```
import cv2
import numpy as np
import glob
```

- 首先，使用 `glob` 库获取当前文件夹路径下所有以文件名以`img`开头的`.jpg`图片（避免运行多次后图片越来越多），在 `for` 循环中对每个图片进行处理：

```
# 获取所有以文件名以`img`开头的.jpg图片路径
image_paths = glob.glob("image*.jpg")
for idx, image_path in enumerate(image_paths):
```

- 预处理图像并进行边缘检测
  - 调用`cv2.cvtColor()`：将图像转换为灰度图像，便于后续处理
  - 调用`cv2.GaussianBlur()`：对灰度图像进行高斯模糊，减少图像中的噪点对拟合的影响（已经没有噪点的图片就不需要）
  - 调用`cv2.Canny()`：进行边缘检测，阈值参数分别设置为 100 和 130
  - 调用`cv2.erode()`：对图像应用腐蚀操作，迭代次数为 1（迭代过多会导致拟合的区域越来越狭窄），将相邻的连接去除

```
    # 灰度处理
    image_gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    if idx == 0:
        # 高斯模糊
        image_blurred = cv2.GaussianBlur(image_gray, (5, 5), 1)
        # 边缘检测
        image_edge = cv2.Canny(image_blurred, 100, 130)
    else:
        # 腐蚀
        image_Erosion = cv2.erode(image_gray, None, iterations=1)
        image_edge = cv2.Canny(image_Erosion, 100, 130)
```

- 获得图像轮廓点集
  - 调用`cv2.threshold()`：对边缘图像进行二值化，使边缘更加分明
  - 调用`cv2.findContours()`：提取轮廓，对于不同的图片使用不同的模式

```
    # 二值化
    _, image_threshold = cv2.threshold(image_edge, 150, 255,
cv2.THRESH_BINARY)

    if idx == 0:
        # 获得轮廓
        image_contours, _ = cv2.findContours(image_threshold,
cv2.RETR_CCOMP, cv2.CHAIN_APPROX_NONE)
    else:
        # 获得轮廓
        image_contours, _ = cv2.findContours(image_threshold, cv2.RETR_LIST,
cv2.CHAIN_APPROX_NONE)
```

- 进行椭圆拟合

- 遍历调用每个轮廓`cv2.fitEllipse()`进行椭圆n拟合，将结果存储在列表中（对于第二张图中过大的形状不进行拟合）

```
    ellipses = []
    for contour in image_contours:
        if len(contour) >= 20 and cv2.contourArea(contour) < 200 or idx ==
0:
            # 计算椭圆拟合结果
            ellipse = cv2.fitEllipse(contour)
            ellipses.append(ellipse)
```

- 进行结果的显示
  - 创建一个与原图大小相同的黑色背景
  - 使用`cv2.ellipse()`分别在原图和新建的黑色背景上绘制拟合图像（为更好地显示，两张图片的边框略有不同）

```
    result = np.zeros_like(image)  # 创建一个黑色背景
    if idx == 0:
        for ellipse in ellipses:
            cv2.ellipse(image_copy, ellipse, (255, 100, 255), 5)  # 在拟合后
的椭圆上绘制边框
            cv2.ellipse(result, ellipse, (255, 255, 255), 5)  # 在黑色画布上绘
制拟合边框
    else:
        for ellipse in ellipses:
            cv2.ellipse(image_copy, ellipse, (255, 50, 255), 2)
            cv2.ellipse(result, ellipse, (255, 50, 255), 1)

    # 保存结果图像
    output_image_overlap_path = f"output_image_overlap_{idx}.jpg"
    cv2.imwrite(output_image_overlap_path, image_copy)
    output_image_fit_path = f"output_image_fit_{idx}.jpg"
    cv2.imwrite(output_image_fit_path, result)
```

3.2 结果测试

- 创建.spec配置文件，输入`pyinstaller -F main.py`命令以生成.spec配置文件和可执行文件，如下图所示：

```
PS D:\study\CV\hw2> pyinstaller -F main.py
632 INFO: PyInstaller: 6.2.0
632 INFO: Python: 3.11.5
634 INFO: Platform: Windows-10-10.0.22621-SP0
634 INFO: wrote D:\study\CV\hw2\main.spec
648 INFO: Extending PYTHONPATH with paths
['D:\\study\\CV\\hw2']
```

```
30369 INFO: Copying 0 resources to EXE
30369 INFO: Embedding manifest in EXE
30451 INFO: Appending PKG archive to EXE
31681 INFO: Fixing EXE headers
35093 INFO: Building EXE from EXE-00.toc completed successfully.
PS D:\study\CV\hw2>
```

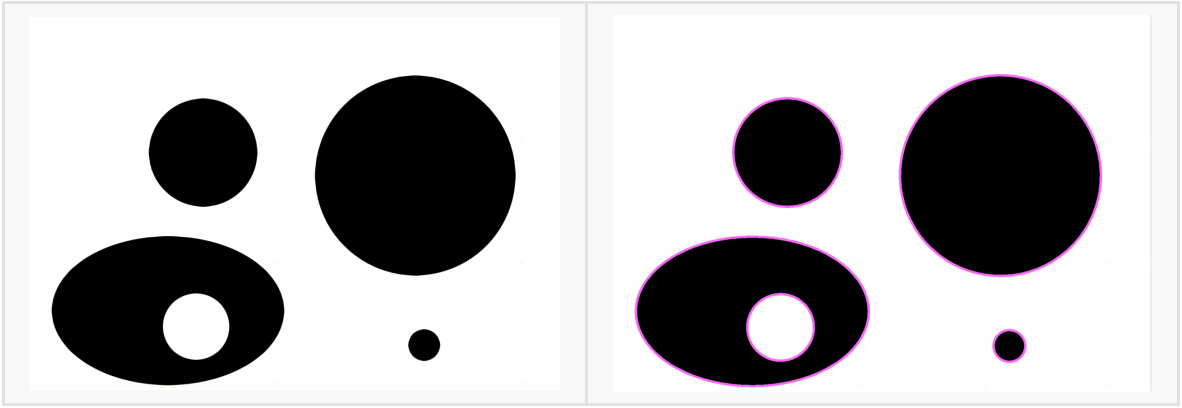
- 输入`path to exe\main.exe path to folder\hw1`运行生成出的可执行文件，如下图所示

```
PS D:\study\CV\hw2> D:\study\CV\hw2\dist\main.exe
PS D:\study\CV\hw2>
```

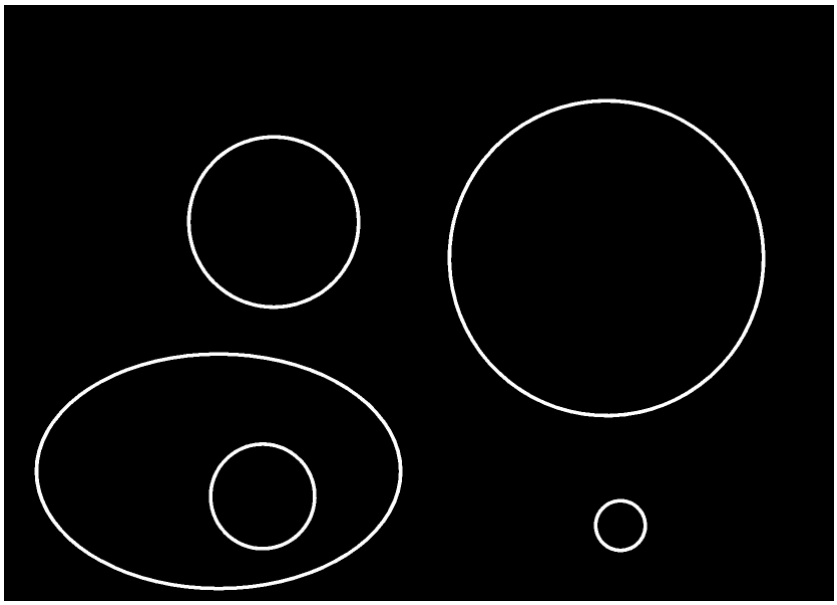
5. 实验结果与分析

5.1 图片一

- 首先是原图和绘制在原图上的拟合图像



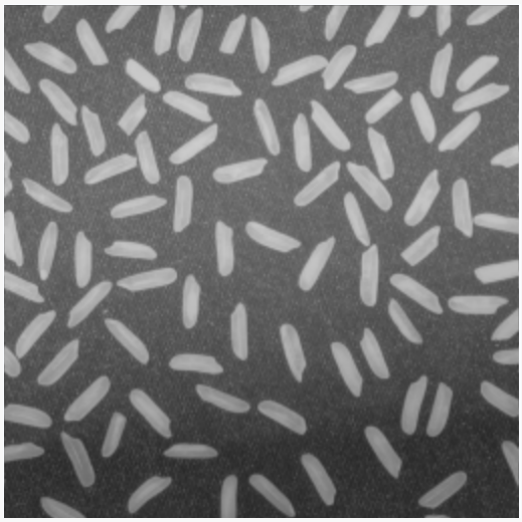
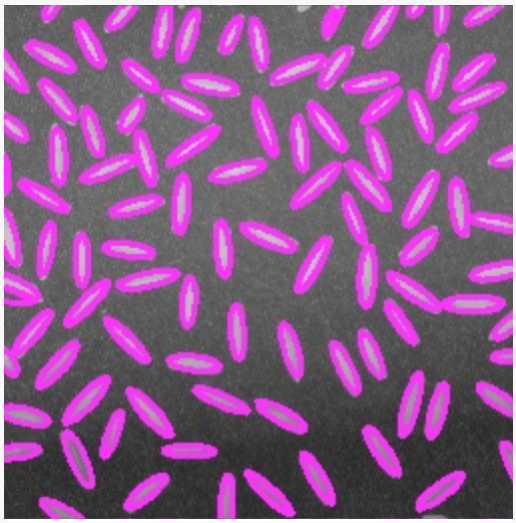
- 然后是绘制在黑色背景上的拟合图像



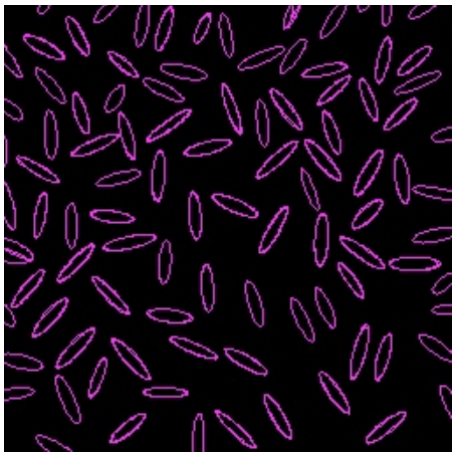
由于这张图片很标准，所以拟合没有什么问题

5.2 图片二

- 首先是原图和绘制在原图上的拟合图像

- 然后是绘制在黑色背景上的拟合图像



基本的椭圆都拟合出来了，并且经过处理，相邻的椭圆也会被分割开

6. 结论与心得体会

通过本次实验，我学习到了OpenCV库对于拟合图形时不同函数的使用方法，包括二值化、灰度处理、轮廓提取等等，同时最重要的是对相关参数的理解和使用，再反复调整参数后最终达到了比较满意的效果，在实验中，一开始的拟合效果不太好，在使用了更多处理方法后才慢慢得到更好的结果，比如在二值化操作后才能拟合出更多的椭圆，在使用腐蚀函数后才使相邻的椭圆没有拟合成一个大椭圆等等，最终形成了灰度处理→高斯模糊→腐蚀→边缘处理→二值化→提取轮廓→进行拟合的流程。

7. 参考文献

- OpenCV 4.0 中文文档 https://vip.minihuo.com/?go?#/chat/1702300763184

- OpenCV 进行图像分割 https://blog.csdn.net/qq_52309640/article/details/120941157