哈尔滨工业大学（深圳）

图像处理实验报告

实验一：指纹图像脊线提取

学号: SZ160110227

姓名: 石嘉晗

日期: 2019.4.15

目录

一、 问题描述

指纹图像处理是指纹图像识别中非常重要的一部分。一个指纹图像由脊（ridge）和谷（mask）组成，通常脊更暗，谷更亮。

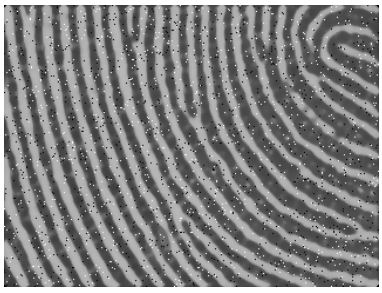
在本次实验中，指纹图像被随机地加入了噪声：椒、盐、椒盐以及高斯噪声。首先要去除噪声，然后再利用特定算法将指纹图像中的脊部分提取出来。

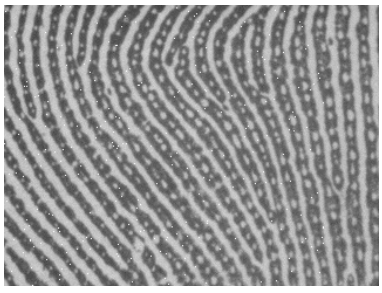
可以使用多种工具，例如：Matlab，OpenCV 等。不限制编程语言。

二、 原始图像

根据要求，我要处理的三张图片如下：


7.bmp


17.bmp


27.bmp

可以看到，这三张图片都有明显的噪声，以及较为明显的汗孔。

三、 实验过程

本次实验我采用 Python3.6 和 OpenCV 作为编程框架。

1. 图像读入：

```
# 读入
im = cv2.imread('./im/' + str(num) + '.bmp')
```

2. 转为灰度图：

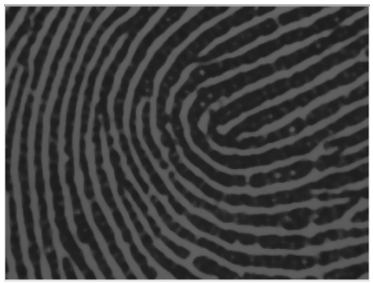
```
# 变为灰度图
im_gray = cv2.cvtColor(im, cv2.COLOR_BGR2GRAY)
```

使用 cvtColor 方法将彩色图像转为灰度图

3. 去噪声

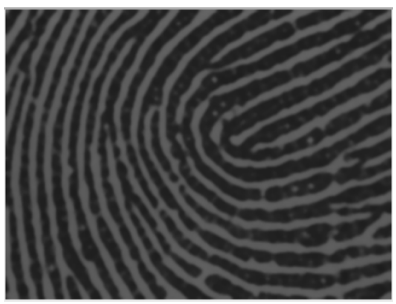
- 椒盐噪声：采用均值滤波器，尺寸为 5*5

  滤波后结果（以下均以 7.bmp 为例）



- 高斯噪声：采用高斯滤波器，尺寸为 3*3

  滤波后结果



4. 二值化：采用 OSTU 方法

OTSU 方法：将图像分成背景和前景两部分看待，前景就是我们要按照阈值分割出来的部分。背景和前景的分界值就是我们要求出的阈值。遍历不同的阈值，计算不同阈值下对应的背景和前景之间的类内方差，当类内方差取得极大值时，此时对应的阈值就是大津法（OTSU 算法）所求的阈值。

```
# 二值化：OTSU方法
ret, im_thresh = cv2.threshold(im_gauss, 0, 255, cv2.THRESH_OTSU)
```

第一个返回值 ret 为使用 OTSU 方法寻找到的阈值，第二个返回值为二值化后的图像。

结果如下

经过二值化后，图像还有汗孔和其他细节需要处理。

5. 汗孔处理

在这里，使用寻找孤立点的方法来处理汗孔。

首先是汗孔定位，利用 cv2.findContours()函数检测所有的外轮廓。

findContours

Finds contours in a binary image.

C++: void **findContours**(InputOutputArray **image**, OutputArrayOfArrays **contours**, OutputArray **hierarchy**, int **mode**, int **method**, Point **offset**=Point())

C++: void **findContours**(InputOutputArray **image**, OutputArrayOfArrays **contours**, int **mode**, int **method**, Point **offset**=Point())

Python: cv2.**findContours**(image, mode, method[, contours[, hierarchy[, offset]]]) → contours, hierarchy

再逐个取出轮廓，计算面积并与汗孔面积的经验值进行比较，如果为汗孔的话则进行填补。

完整函数如下：

```
# 去除汗孔
def remove_pore(im, pore_size_max):
    '''
    :param im: 需要去除汗孔的图像
    :param pore_size_max: 汗孔面积的最大值（经验值）
    :return: 处理后图像
    '''
    # cv2.RETR_EXTERNAL：只检测外轮廓
    # cv2.CHAIN_APPROX_NON：存储所有的轮廓点
    image, contours, hierarchy = cv2.findContours(im, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_NONE)
    for i in range(len(contours)):
        area = cv2.contourArea(contours[i])
        if area <= pore_size_max:
            cv2.drawContours(image, [contours[i]], 0, 0, -1)
    return image
```

经过第一次处理后效果如下

6. 细节处理

采用形态学操作，在这里经过多次比较，选择用一次闭操作来处理，使得细节更为完整。

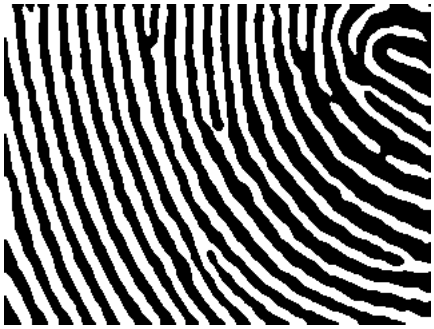
得到最终图片如下

四、 完整代码与结果

```
import cv2
import numpy as np


# 去除汗孔
def remove_pore(im, pore_size_max):
    '''
    :param im: 需要去除汗孔的图像
    :param pore_size_max: 汗孔面积的最大值（经验值）
    :return: 处理后图像
    '''
    # cv2.RETR_EXTERNAL：只检测外轮廓
    # cv2.CHAIN_APPROX_NON：存储所有的轮廓点
    image, contours, hierarchy = cv2.findContours(im, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_NONE)
    for i in range(len(contours)):
        area = cv2.contourArea(contours[i])
        if area <= pore_size_max:
            cv2.drawContours(image, [contours[i]], 0, 0, -1)
    return image


def preprocess(num):
    """
    对图像进行处理，彩色图变为二值图，移除汗孔，细节处理
    :param num: 要处理的图像编号
    :return: 处理后的图像，图像编号
    """
    # 读入
    im = cv2.imread('./im/' + str(num) + '.bmp')
    # 变为灰度图
    im_gray = cv2.cvtColor(im, cv2.COLOR_BGR2GRAY)
    # 均值滤波：去除椒盐噪声
    im_median = cv2.medianBlur(im_gray, 5)
    # 高斯滤波：去除高斯噪声
    im_gauss = cv2.GaussianBlur(im_median, (3, 3), 0)
    # 二值化：OTSU方法
    ret, im_thresh = cv2.threshold(im_gauss, 0, 255, cv2.THRESH_OTSU)
    # 移除汗孔
    im_rp1 = remove_pore(im=im_thresh, pore_size_max=36)
    # 形态学变换
    closing = cv2.morphologyEx(im_rp1, cv2.MORPH_CLOSE, kernel=np.ones((3, 3), np.uint8), iterations=1)
    im_final = closing
    return im_final, num


def img_write(image, num):
    '''
    将处理后的图像写入
    :param image:处理后图像
    :param num:图像编号
    :return:成功信息
    '''
    cv2.imwrite('./im_process/result_' + str(num) + '.bmp', image)
    return "Picture {} .".format(num)


if __name__ == '__main__':
    img_list = [7, 17, 27]
    for img in img_list:
        final, num = preprocess(img)
        img_write(final, num=num)
```

处理后的结果如下

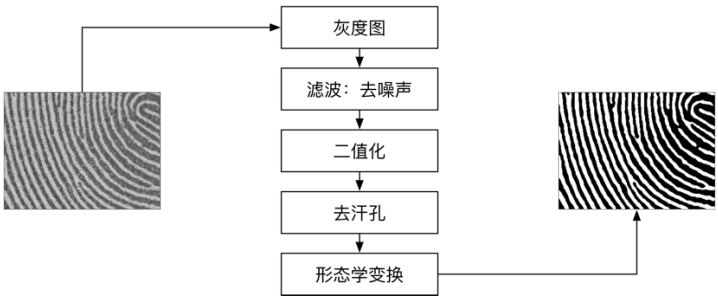
7



17

27

五、 总结



在本次实验中，我实现了完整的指纹图像预处理流程，理解了教材上图像处理的方法如何在实际中运用，并掌握了 python-opencv 的使用方法。