数字图像处理实验五

09021227 金桥

2023 年 12 月 8 日

1 实验目标

1. 课本 410 页图 9.11，对指纹图分别进行 (d)(e)(f) 三步操作。最终结果图还是有断开，思考如何把断开处连接。
2. 对指纹图分别进行 Global Thresholding 操作和 Otsu's Thresholding 操作，并对性能进行分析。
3. 使用 Canny 算子检测边界，可调用函数，分析每一步操作作用，思考为什么会比 Sobel 方法更好。

2 过程与方法

2.1 膨胀与腐蚀

膨胀与腐蚀参照课本提供算法，结构元素均为 3×3 大小的矩阵。

2.2 阈值

2.2.1 Global Thresholding

Global Thresholding 算法如下：

- 为全局阈值 T 选择一个初始的估计值。
- 使用初始值 T 进行阈值分割，此时图像分为两组像素：
  - 大于阈值的像素组 G_1 以及小于阈值的像素组 G_2.
- 分别计算属于 G_1, G_2 像素的平均灰度值 m_1, m_2.
- 对 m_1, m_2 计算一个新的阈值 $T = \frac{m_1+m_2}{2}$.
- 重复之前的步骤，直到旧阈值与新阈值的差小于预设的值 Δ 为止。这里设置 $\Delta = 1$.

2.2.2 Otsu's Thresholding

Otsu's Thresholding 算法如下：

- 计算所有灰度的概率分布，设灰度为 i 的概率为 P_i.
- 计算累计和 $S(k) = \sum_{i=0}^{k} P_i$.
- 计算累计均值 $m(k) = \sum_{i=0}^{k} iP_i$.
- 计算全局灰度均值 m_G.
- 计算类间方差 $\sigma_B^2(k)$，公式如下：

$$\sigma_B^2(k) = \frac{(m_G S(k) - m(k))^2}{S(k)(1 - S(k))}$$

.

- Otsu 阈值 k^* 为使得类间方差最大的 k，即：

$$k^* = \arg\max \sigma_B^2(k)$$

2.2.3 Canny 算子

采用 OpenCV 的 `cv::Canny` 函数。

3 结果与分析

3.1 膨胀与腐蚀

图 1: 课本 410 页图 9.11，对指纹图分别进行 (d)(e)(f) 三步操作。最左侧为原图，右侧依次是 (d)(e)(f).

图 2: 调大图像后应用多次膨胀与腐蚀的效果。指纹基本没有断开，但是也造成了噪声的保留。

通过调大图像或调小结构元，可以控制膨胀与腐蚀的力度，从而达到更精确的控制，使得指纹不会有断开的地方。但与此同时会造成一些噪点的保留。

3.2 阈值处理

Global Thresholding 的时间复杂度为 $O(KNM)$, k 为迭代次数，N, M 为图像宽度与高度。

Otsu's Thresholding 的时间复杂度为 $O(NM)$, N, M 为图像宽度与高度。

3.3 检测边界

Canny 算子检测边界的主要步骤与作用如下：

- 去噪。噪声会影响边缘检测的准确性，因此首先要将噪声过滤掉。
- 计算梯度的幅度与方向。
- 非极大值抑制，即适当的让边缘变瘦。
- 确定边缘。使用双阈值算法确定最终的边缘信息。

相较于 Sobel 方法，Canny 的优势在于：它能够尽可能多地标识出图像中的实际边缘，且标识出的边缘与实际图像中的实际边缘更接近。

图 3: 对指纹图分别进行 Global Thresholding 操作和 Otsu's Thresholding 操作。从左到右分别是原图，采用 Global Thresholding 处理的图像与采用 Otsu's Thresholding 处理的图像。由于图像本身的性质，所以这两种方式处理的效果基本一致。但实际上两幅图的直方图存在细微的差别。

图 4: 采用 OpenCV 的 `cv::Canny` 进行边缘检测。左侧为原图，右侧为处理后的图像。