数字图像处理综合作业 2——指纹特征提取综合算法

陈昭熹 2017011552

2019 年 11 月 20 日

目录

1 运行说明

1.1 文件结构

1.1.1 基础算法类

OpenProc.m 自己实现的形态学开运算

CloseProc.m 自己实现的形态学闭运算

Erode.m 自己实现的形态学腐蚀

Dilate.m 自己实现的形态学膨胀

GeodesicDilation.m 自己实现的测地学膨胀算法

1.1.2 各流程功能函数

RidgeSegmentation.m 脊线粗分割

FillHoles.m 脊线细分割 (填补空洞、平滑毛刺、去除空岛)

FeatureDetect.m 细节点检测

NeighborPostProc.m 基于细节点的邻域形态学算法 (去除桥接，优化脊线图)

EndValidation.m 细节点验证

ShowFeature.m 细节点可视化

1.2 使用方法

entry_25.m 处理 r2_5.bmp 的入口文件

entry_964.m 处理 r96_4.bmp 的入口文件

2 算法流程与实现

2.1 算法框架

Algorithm 1: 指纹特征提取综合算法

Input: I - Enhanced fingerprint Image

Output: $feature$ - fingerprint Image contains feature points

```
if I not empty then
    bin_I ←imbinarize(I)
    segment ←RidgeSegmentation(bin_I)
    thined ← bwmorph('thin',segment)
    feature ← FeatureDetect(thined)
    feature ← Validation(feature)
end
```

2.2 脊线分割

脊线分割主要分为两步，即**粗分割和细分割**。

粗分割主要通过形态学闭运算来去除二值化图像中的桥接，生成尽可能处于正确脊线位置的图像，在利用原二值化图像作为 mask，闭运算之后的图像作为 marker 进行测地学膨胀，从而达到去除空岛的目的。其原理为，由于闭运算能够将较多的非脊线像素去除，且结构元素设置的比较 aggressive，这使得留下来的部分一定是原图中较为明显的脊线特征 (连续且宽度足够)，这样一来使用这样的图像作为种子来在原图的 mask 上做测地学膨胀就可以将闭运算中被腐蚀的脊线还原为原来的宽度和长度，而由于空岛与脊线的拓扑不连通性，就不会将空岛恢复出来。由于算法比较过程比较简单，只需要调用本文实现的形态学算法即可，算法流程伪代码在此不做展示。

细分割则通过 3x3 的小邻域遍历粗分割图像，通过邻域内的像素和以及相应阈值来判断该区域是否是空洞或者是毛刺，从而完成填补空洞，并移

除部分毛刺的功能。具体算法流程如下：

Algorithm 2: 细分割算法

Input: I - rough segmentation

Output: R - filted segmentation

```
for each 3 × 3 region in I do
    if region(2,2) == 1 then
        if sum(region) ≤ 4 then
            region(2,2) ← 0
        end
    end
    if region(2,2) == 0 then
        if sum(region) ≥ 5 then
            region(2,2) ← 1
        end
    end
end
```

2.3 脊线细化

脊线细化算法分为两步，**脊线细化以及基于细节点的邻域形态学算法**。这个步骤可以迭代多次，以达到最好的效果。针对给出的两张图均迭代了两次。

脊线细化按照要求调用库函数，无须赘述。

基于细节点的邻域形态学算法，是本文实现的一种用于去除前一步分割中所难以去除的顽固桥接、毛刺和不合理增强的形态学算法。其基本原理是建立在一个**假设**上，即指纹脊线的分叉点与端点或者分叉点与分叉点之间的距离不能过近。这也可以说是一个统计学先验知识，几乎没有指纹会呈现出连续两三个像素宽度内出现两个分叉点或者同时出现分叉点和端点的。导致这种情况的只能被认为是图像中的毛刺、桥接或者不合理增强所导致的结果。基于这样的原理，可以使用一张细分割图以及其关键点图，通过一个 9x9 的邻域搜索算法，寻找细分割图中的关键点。当当前邻域中心像素匹配到分叉点时，观察邻域内是否存在其他细节点 (分叉点或端点均可)，若出现了则应当视为一个不合理邻域，说明原细分割图中出现了桥接、毛刺或不合理增强。此时选择这两个点作为**矩形对角线上的两个顶点**，对原细分割

图的这个小矩形区域进行形态学膨胀，即可消除桥接、毛刺或不合理增强。具体算法流程如下:

Algorithm 3: 基于细节点的邻域形态学算法

Input: I - filted segmentation, feature - feature points

Output: R - refined segmentation

```
for each 9 × 9 region in I do
    if region(5,5) == bifurcation point then
        candidate ← find(region == ending point)
    end
    if candidate is not empty then
        Do Dilate in rectangle area between candidate(1) and
         region(5, 5)
    end
end
```

2.4 细节点检测

细节点检测使用作业文档中提示的算法，直接在细化图上进行邻域操作，效果很好，在此不做赘述，只给出流程:

Algorithm 4: 细节点检测算法

Input: I - thined fingerprint

Output: feature - sparse Image contains feature points

```
for each 3 × 3 region in I do
    if region(2,2) == 0 then
        CrossNum ← (1/2)Σ|f(p_{(i+1) mod 8}) − f(p_i)|
        if CrossNum == 1 then
            feature at region(2, 2) ← 0 //ending point
        end
        if CrossNum == 3 then
            feature at region(2, 2) ← 128 //bifurcation point
        end
    end
end
```

2.5 细节点验证

值得注意的是，传统的细节点验证算法需要考虑边缘的非端点以及内部的一些错误细节点。但在本文的实现过程中，指纹区域内部的错误细节点均已经通过基于细节点的邻域形态学算法予以过滤和清除，**因此本环节不需要考虑指纹区域内部，只需要着重去除边缘的端点即可**。细节点验证算法的原理基于一个先验知识——边缘的端点其周围至少存在一侧的像素均为白色，这巧妙地利用了给定的增强图在非指纹区域都是白色填充的特点。细节点验证只需要进行一个稀疏的大邻域搜索，在每一个端点处取 31x31 邻域内的所有像素并求和，其小于特定阈值的时候则判定为边缘的端点，直接将其置为背景即可。算法流程如下：

Algorithm 5: 细节点验证算法

Input: I - segmentation, feature - sparse feature points

Output: newfeature - Validated feature

```
for each 31 × 31 region in I do
    if feature at region(16,16) == 0 then
        /
    end
    /ending point  if sum(region) < threshold then
        feature at region(16, 16) ← 255 //non-feature point
    end
end
```

3 中间过程及结果

3.1 r96_4.bmp

3.1.1 关键流程节点

图 1: r96_4.bmp 脊线分割结果

图 2: r96_4.bmp 脊线细化结果

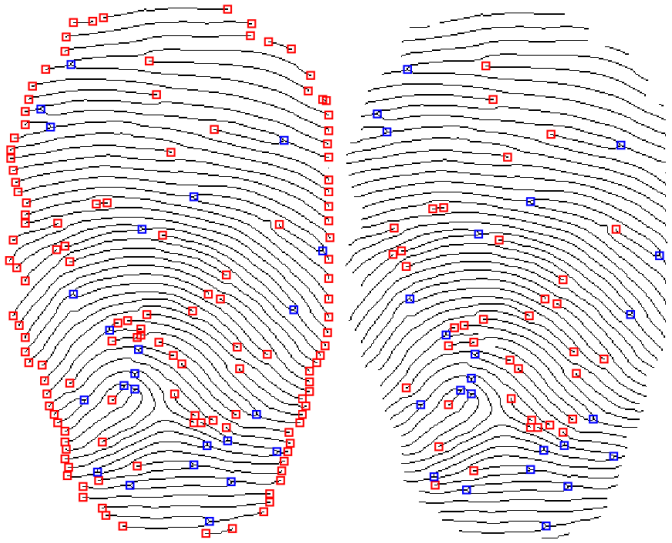

图 3: r96_4.bmp 细节点检测 (左) vs 细节点验证 (右)

3.1.2 部分算法细节

这里可以注意到，直接对原始增强图进行二值化会产生空洞、毛刺和不当桥接。

图 4: r96_4.bmp 二值化后原始分割图像

在使用了细分割算法之后，效果如下所示，图中较为明显的绿框所在的三个明显瑕疵被去除。

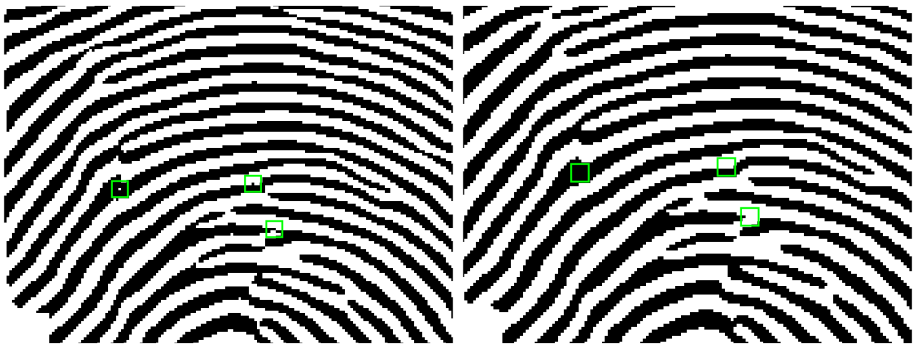

图 5: r96_4.bmp 填充空洞、滤除毛刺效果 before vs after

由于细分割算法仍不能去除一些顽固的桥接、毛刺以及原增强图中存在不合理的增强，会导致细化图产生如下左所示的奇怪特征，这可以通过基于细节点的邻域形态学算法进行去除和优化。图中显示了该算法迭代一次和两次细化图的变化。

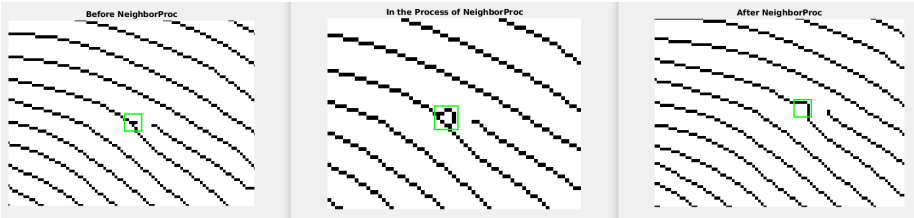

图 6: r96_4.bmp 基于细节点的邻域形态学算法——同时去除原图和细节图中的桥接、毛刺

3.2 r2_5.bmp

3.2.1 关键流程节点

图 7: r2_5.bmp 脊线分割结果

图 8: r2_5.bmp 脊线细化结果

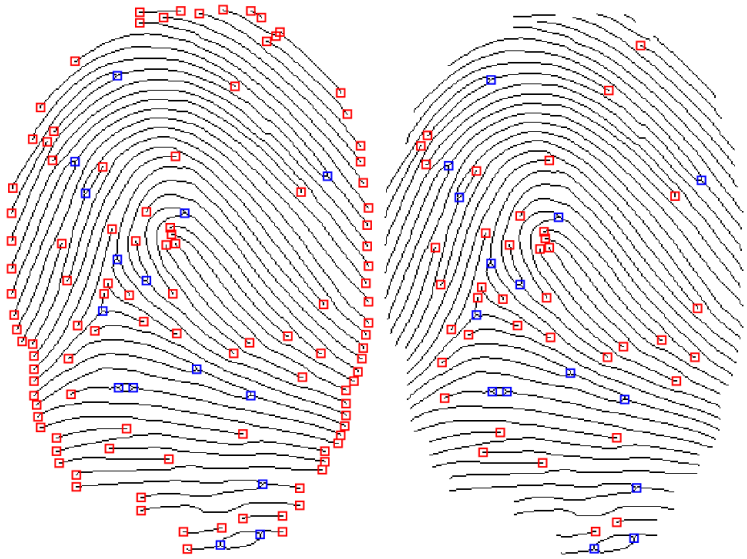

图 9: r2_5.bmp 细节点检测 (左) vs 细节点验证 (右)

3.2.2 部分算法细节

与上一张图片一样，直接二值化后的图像如下左所示存在空洞、空岛和毛刺。值得注意的是，图中的**蓝色框所示区域**非常有意思，它在原图中是一个良好恢复出的空岛特征，理论上是不应该使用算法去填充的，否则会让原有指纹图像减少两个特征，这在刑侦上是极为不利的。而本文结合所实现的细分割算法和基于细节点的邻域形态学算法能够较好的保留关键细节，同时去除不该有的元素。

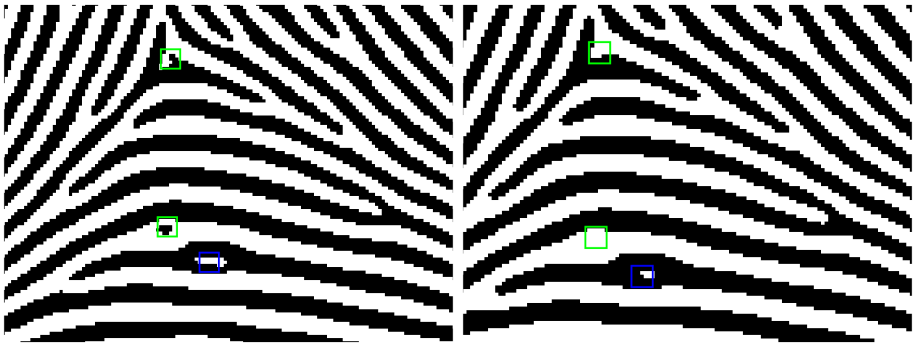

图 10: r2_5.bmp 填充空洞、去除空岛、平滑毛刺效果 1 before vs after

下图可见，蓝色框区域在原图中是一个明显的双分叉点形成的空岛结构，此前咨询冯老师也反馈道这虽然不普遍，但也是在指纹中会常出现的一种现象，而这种特征一旦保留下来将对应用中的刑侦大为有帮助，因此不能将其与其他空岛等量齐观。

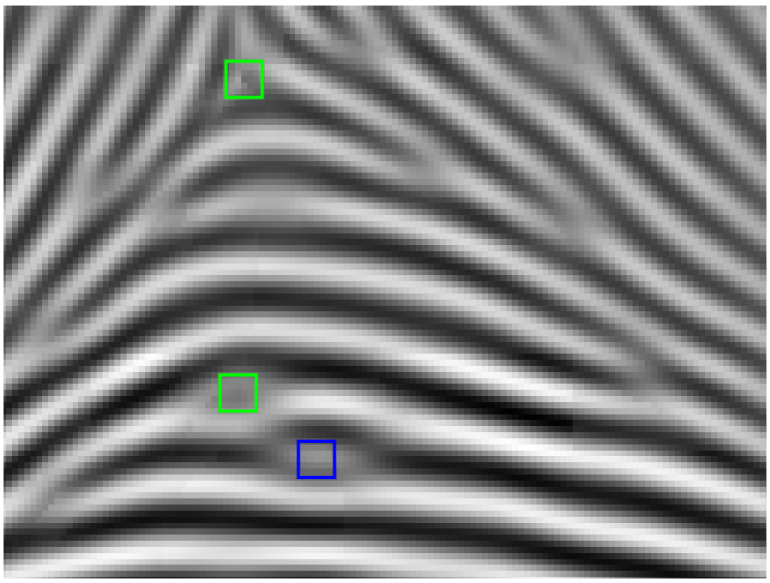

图 11: r2_5.bmp 鲁棒分割算法，合理保留原图细节

下图是一些较为困难的情形，也可以很好的 work。

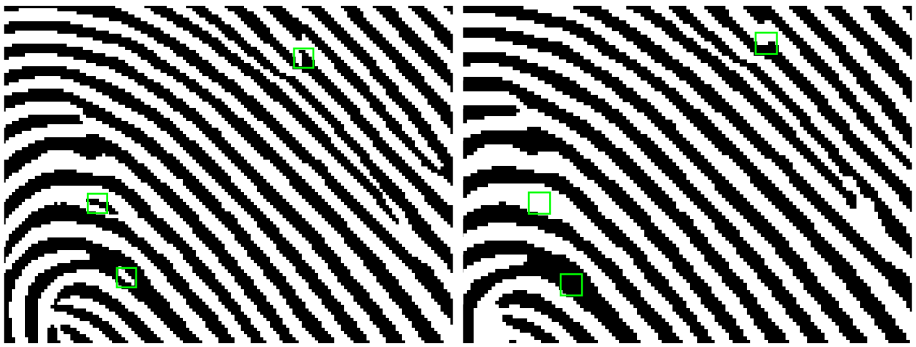

图 12: r2_5.bmp 填充空洞、去除空岛、平滑毛刺效果 2 before vs after

下图展示了本文实现的基于细节点的邻域形态学算法的强大能力。在

一些较为狭窄的区域 (图中脊线之间仅有一个 pixel 距离)，直接使用腐蚀、膨胀、开闭运算等方法会必然导致桥接，而使用本文算法则可以较好的将图像优化，得出正确的特征结果。

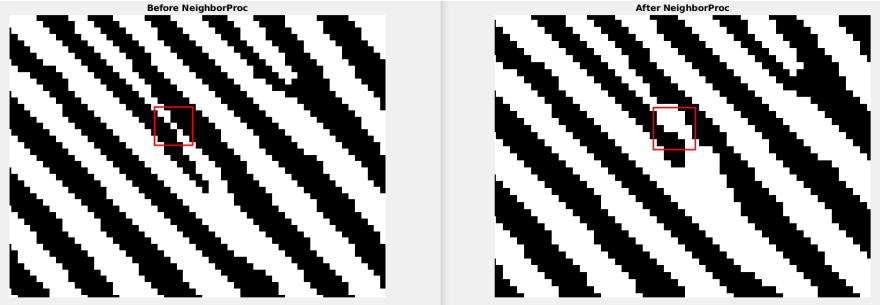

图 13: r2_5.bmp 基于细节点的邻域形态学算法——去除桥接: 脊线图变化

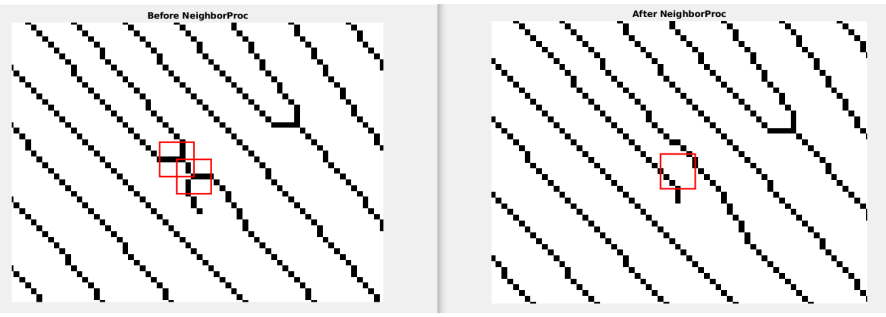

图 14: r2_5.bmp 基于细节点的邻域形态学算法——去除桥接：细化图变化