暨南大学计算机科学系

《数字图像处理》课程设计

课程设计名称： 基于 OPENCV 的指纹图像骨骼化处理研究

院（系）: 信息计算科学系

专业： 计算机科学与技术

学号: 2014051742

姓名: 冯志平

指导老师: 张庆丰

日期： 2017.6.29

基于 OpenCV 的指纹图像骨骼化处理研究

一、 概述

如今，指纹在刑侦、安全方面均起到十分重要的作用。指纹由于其生理特点，往往不能清晰地获取。指纹图像常常伴随有许许多多的断开处和污渍等等。本文将对这些方面的处理进行研究，并且以获得一幅细线化（即骨骼化）的清晰图像为目的。

二、 条纹滤波算法（Stripe Filter）

“条纹滤波”是本文提出的一个基于指纹图像特点的特殊滤波算法。我们知道指纹之所以是“纹”，正是因为它常常是由同方向的条纹所构成的。基于这一考虑，本文利用统计学上“方差”的概念，来求出条纹的方向。

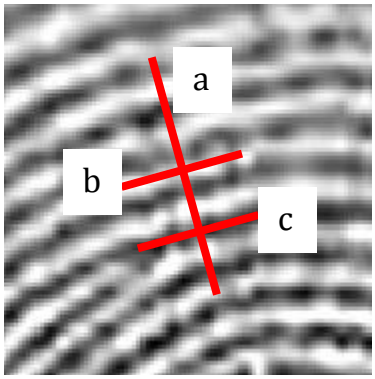

Figure 2-1 指纹图像局部

上图，是指纹图像的一个局部区域，我们看到，其中的横向的条纹十分多。我们如果在 a 线的方向上沿着这条线取像素点，我们可以画出它的图像，大概是这样的：

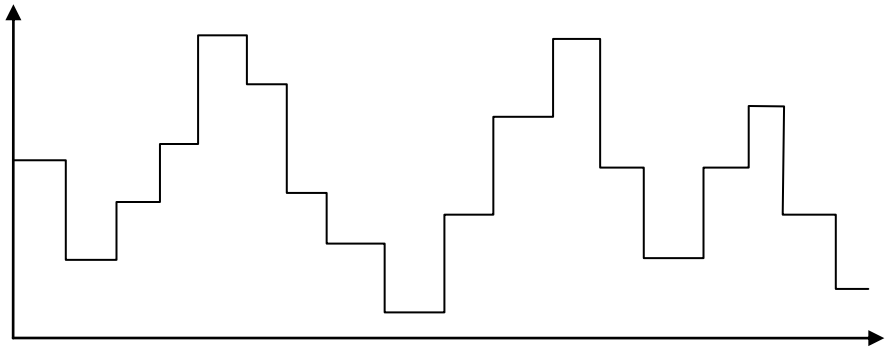

Figure 2-2. 起伏十分大的 a 线取样点

可以看到，a 线的起伏十分大，因为它横跨了几条条纹，其中每一个低谷点，就是一条指纹纹路的中心。但是当我们观察 b 线、c 线上的取样时，其起伏明显平缓许多：

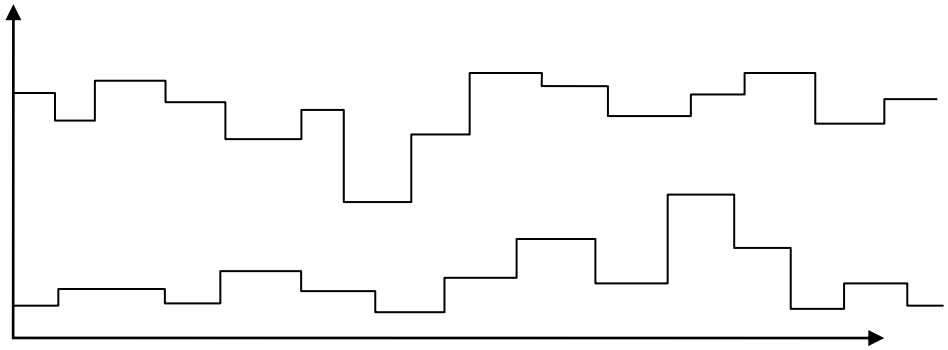

Figure 2-3. 起伏显然更小的 b、c 线取样

因为硬件上的问题，或者图像不清晰、手指上有污渍或划伤所造成的起伏虽然存在，但是 b、c 线很明显保持了黑白分明的特征，而且起伏比 a 要小得多。起伏在统计上表现为方差，方差越小起伏越小，反之则越大。

我们定义，在指纹图像上的每一点 P 为中点的、长度为 2r+1 的线段为 P 上的采样线。采样线绕 P 转动。若一条采样线在 P 上所有采样线中，它的样品有最小的方差（比如 b、c 线），那么定义它为 P 点上的切线。若一条采样线在 P 上所有采样线中，它的样品有最大的方差（比如 a 线），那么定义它为 P 点上的法线。

在最后，我们认定一点切线上均值的数值比法线上均值的数值要大时，P 点是白的，否则 P 点是黑的。下面用伪代码描述这个算法：

```
StripeFilter(In, Out)
    foreach P in the range of image In
        tangent, normal = GenSamplingLine(In, P, 5)
        tcolor = 0
        ncolor = 0
        foreach c along tangent
            tcolor += In(c)
        foreach c along normal
            ncolor += In(c)
        if tcolor > ncolor
            Out(P) = 1
        else
            Out(P) = 0

GenSamplingLine(In, P, r) //本文中默认 r 是 5
    min_rel, max_rel
    min_var = ∞, max_var = 0
    foreach α  between [0,π ] //离散地在 0 到π 之间取角度
        rel = 中心点为 P、角度为α 、长度为 2r+1 的采样坐标集
        aver = 0, var = 0
        foreach c along rel
            aver += In(c)
        aver /= Len(c)
        foreach c along rel
            var += (In(c) - aver)^2
        // 求出方差，并比较大小
        if var > max_var
            max_var = var
            max_rel = rel
        if var < min_var
            min_var = var
            min_rel = rel

    return min_rel, max_rel
```

Figure 2-4. Pseudo-code List

三、 骨骼化算法（SKELETONAZATION）

本文采用了 T.Y. ZHANG 和 C.Y. SUEN 的快速骨骼化模式算法。下面简要描述一下该算法的内容。首先，文章定义了一个点 P 的邻域为 P1 ~ P9，如下表所示：

P9	P2	P3
P8	P1	P4
P7	P6	P5

Firgure 3-1. 邻域标号

首先，程序移除二值图像中所有满足下列条件之一的白点（称为第一个子迭代的条件或 First Iteration Condition）：

- 2 <= B(P1) <= 6
- A(P1) = 1
- P2*P4*P6 = 0
- P4*P6*P8 = 0

其中函数 A 为 P1~P9 依次包含 01 模式的数量，比如 011101010 中，B 的值便为 3。而函数 B 则为 P1~P9 之加和或白点的数量。

在做完这一轮迭代之后，将移除满足下列条件之一的白点（称为第二个子迭代的条件或 Second Iteration Condition）：

- P2*P4*P8 = 0
- P2*P6*P8 = 0

在本实现中，计算函数 A 使用了一个自动状态机（初始状态为 1）：

当前状态	输入字符	下一状态	动作
0	0	0	-
0	1	1	a++
1	0	0	-
1	1	1	-

Firgure 3-2. 状态机表

状态机运行完成后，a 值便为函数 A 的值。

四、 程序构成与解释

本节须结合程序源代码阅读。

程序中有这些函数，它们的作用分别为：

- safeAt 是用于取得图像中的点，但是为了保证坐标越界不会产生段错误，该函数进行了越界检查，并且在越界时返回默认值。
- getbool 是用于将骨骼化算法中的 Pn 转化为坐标上的偏移。
- genSamplingLine 用于产生采样线中方差最大和最小的两条，也就是切线和法线。
- stripeFilter 用于对图像进行条纹滤波（见第二节内容）
- skeletonization 是骨骼化算法的实现（见第三节内容）
- eliminateHoles 是一个对黑色像素点更加有优势的去噪算法。这是一个非线性滤波器，它计算核内黑色点是否超过 c 个，如果超过则当前点的卷积值为 0（黑色），否则为保持原本的颜色（留意不是白色，因为算法对黑色像素更有优势）。

对指纹的处理有若干个阶段组成：

1. stripeFilter：第一次条纹滤波
2. gaussian：高斯滤波上一阶段的结果
3. stripeFilter：对高斯后的结果进行第二次条纹滤波
4. elimhole：去掉孔洞
5. skeleton：细线化

每一个阶段都有输出结果。这是由于可以用命令行参数来控制阶段的运行。程序接受这些参数：

```
./skeletonization <stage_name> <inimg> <outimg>
```

stage_name 可以是上面的阶段名。后两个参数分别是输入图像和输出图像。下一个阶段的输入往往就是上一个阶段的输出，所以只需要指定正确的文件名即可。预设的执行命令串如下：

```
$ ./skeletonization stripefilter skelet-img.jpg skelet-out-filtered.jpg
$ ./skeletonization gaussian skelet-out-filtered.jpg skelet-out-gaussian.jpg
$ ./skeletonization stripefilter skelet-out-gaussian.jpg skelet-out-
filtered2.jpg
$ ./skeletonization elimhole skelet-out-filtered2.jpg skelet-out-elimhole.jpg
$ ./skeletonization skeleton skelet-out-elimhole.jpg skelet-out-final.jpg
```

Figure 4-1. Executed Command

在这里，skelet-img.jpg 为输入图像，skelet-out-final 为输出图像，其余均是中间产物。

五、 算法结果与对照

本文的处理在这一幅图像下进行。左边两个是相同手指指纹的两次采集：

Figure 5-1. 原始指纹图像

第一次条纹滤波的结果：

Figure 5-2. 条纹滤波结果

可以看到，条纹滤波保持了线条的圆滑和连续。如果仅仅用 Naive 的二值化，将会是这样的结果：

Figure 5-3. Threshold 结果

中间比较清晰地部分并没有什么问题，但是出现了大量的断裂。条纹滤波很好地解决了这一问题。在两次的条纹滤波之后，结果的图像已经相当清晰完整。之后去除噪点和孔洞等等，作一系列增强处理：

Figure 5-5. 第二次条纹滤波

Figure 5-6. 去除孔洞和背景

Figure 5-7. 中值滤波（对比）牺牲了许多细节，并不理想

Figure 5-7. 本图原是黑底白图，为了方便印刷作了反色处理

六、 参考文献

[1] T.Y. ZHANG, C.Y. SUEN. A Fast Parallel Algorithm for Thinning Digital Patterns, Mar. 1984, Communication of ACM

[2] 李 利 , 范九伦, 一种有效的指纹图像方向滤波增强算法, Oct 2008