目录

一、项目概况.....1
1.1 项目背景.....1
1.2 项目分析.....1
1.2.1 项目意义.....1
1.2.2 任务要求.....2
1.2.3 项目难点.....2
1.3 解决思路（概述解决方法）.....4
1.4 技术优势分析（与其他方法相比有什么优势）.....5
1.4.1 基于 Sobel 算子、积分模板和 Garbo 滤波的多重图像增强 ..5
1.4.2 基于 Phase 算法的脊线方向计算 5
1.4.3 基于 MCC 技术的指纹图像匹配.....6
1.4.4 全步骤图像展示的交互界面.....7
二、技术路线及实现方案.....7
2.1 开发环境.....7
2.2 技术方案.....8
2.3 核心技术（把前面 1.3 提到的技术详细写一写）.....8
2.3.1 AS608 模块录入指纹.....8
2.3.2 上位机读取指纹数据并存储为图像.....9
2.3.3 基于 Sobel 算子的图像分割 11
2.3.3 指纹局部脊线方向估计.....12
2.3.4 指纹局部脊线频率的估计.....13
2.3.5 基于 Gabor 滤波的图像增强.....14
2.3.6 检测特征点位置.....15
2.3.7 估计特征点方向.....17
2.3.8 基于 MCC 创建局部结构.....18
2.3.9 基于 LSS 的指纹图像对比.....19

三、系统成果展示.....20
3.1 系统展示.....20
3.2 基础场景测试.....26
3.2.1 录入指纹.....26
3.2.2 指纹识别.....26
六、参考文献.....28

一、项目概况

1.1 项目背景

指纹识别技术作为生物识别技术中最为具有应用前景的技术之一，近年来取得了长足的发展，并广泛应用于各种场合。由于指纹所具有的唯一性和不变性，以及指纹识别技术具有很高的可行性和实用性，指纹识别成为目前最流行、最可靠的个人身份认证技术之一。所以对指纹识别技术的研究具有重要的理论和实际意义，指纹识别的一般性过程分为三步：指纹图像的预处理，指纹特征提取以及特征匹配。

然而现有的角点检测算法的鲁棒性和适应性并不能非常好的满足上述产业的发展需求，这样的市场条件环境下，MCC（Minutia Cylinder-Code）指纹特征表征匹配算法成为了当下的研究热点。

且随着指纹识别应用的不断推广，大众对与指纹识别的流程和原理了解甚少，因此未免产生误会和猜疑，这也使得更加需要进行对带有教学展示功能的指纹识别系统的开发。

MCC 特征匹配算法作为指纹识别技术中的核心算法，对提升匹配准确率是至关重要的。尤其是针对指纹图像的旋转不确定性，传统的特征匹配算法难以实现理想的效果，而 MCC 技术，则能够比较便利、优越地解决相关问题。

且一步一图的展示过程，可以使得没有技术基础的大众了解指纹识别的过程，从而增加对指纹识别的信任度。

1.2 项目分析

1.2.1 项目意义

目前，基于角点检测等算法进行指纹识别匹配的系统，当遇到指纹图像随机旋转或指纹图像成像不清晰时，便难以识别，例如指纹锁应用中，易于发生主人无法开门的情况，严重降低了用于的使用体验，同时部分用户对内部原理了解较少，导致对指纹识别系统的信任度低，解决这两个问题是当下的市场所需。

利用基于 MCC 的纯图像处理识别技术和一步一图的展示平台来帮助指纹识

别系统提高识别准确率和交互性，是此项目的研究方向，然而由于目前还没有出现基于 MCC 技术且具有展示功能的指纹识别系统出现，因此此项目由零开始从头构建，通过此项目，不仅能够较为理想地解决由于指纹旋转随机性导致识别率下降这一问题，还有极高的交互性，基于此进行进一步的开发，就可以进一步地减少误差，提高用户使用体验和信任度，有可观的商业竞争力。

1.2.2 任务要求

客户期望通过本系统实现以下几点功能：

（1）使用 MCC 技术提高指纹识别的准确率。

（2）使用一步一图的方式增加交互性，使用户了解指纹识别的内部原理。

（3）用户可以选择是否进入交互式展示功能，防止展示时间过长，影响用户体验。

（4）避免使用深度学习等可解释性差的方法，尽量使用数字图像处理技术。

1.2.3 项目难点

（1）指纹图像旋转不确定性

指纹本身的特点虽然不会改变，但是每次进行指纹识别的角度和方向均不相同，导致硬件模块读入的图片本身形态各异，没有固定的易于辨别的特征，且每次读入的指纹位置也会有部分，部分情况下还会有噪声影响，运用传统的特征点匹配算法难以将准确率提高。

图 1-2-1、图 1-2-2 与图 1-2-3 为同一指纹在不同方向上提取到的图片，可以看到由于手指位置也并不相同，导致图片之间差异较大，特征点之间的关系较为复杂。

图 1-2-1 方向 1 指纹图

图 1-2-2 方向 2 指纹图

图 1-2-3 方向 3 指纹图

（2）存在环境干扰

指纹识别系统作为面向大众的工程系统，需要进行匹配识别的图像可能是由指纹磨损严重的工人提供，也可能是由手指表面皮肤破皮较多的农人提供，甚至还要面临不法分子的违法行为，例如使用胶布等工具进行指纹采集，不清晰或仿造的图像为指纹识别增加了难度，具体情况如图所示：

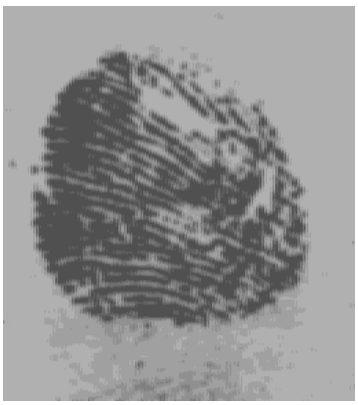


图 1-2-4 具有干扰的指纹图

（3）业内暂无结合交互展示的系统

由于携带交互式展示的指纹识别系统的理念较为新颖，因此业内几乎没有类似功能的系统可供参考，此项目基于 MCC 技术，从零开始构建，实现交互性较好的系统平台需要各方向的学习和研究。

1.3 解决思路（概述解决方法）

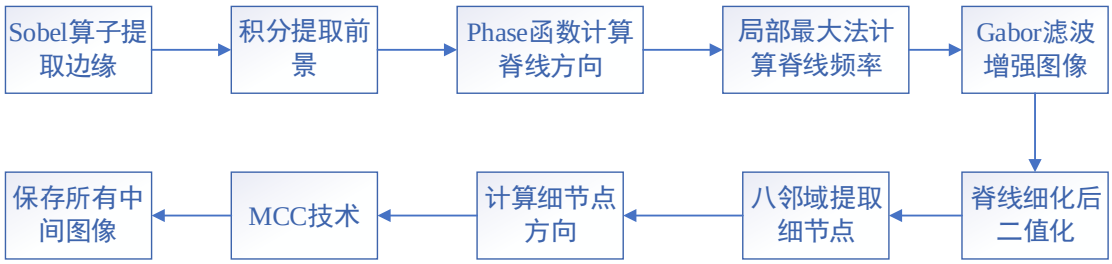


图 1-3-1 解决思路示意图

（1）利用 Sobel 算子提取 x 和 y 两个方向的边缘梯度信息，并求其梯度的平方和得到较为清晰的指纹图像边缘信息。

（2）利用积分模块积分后根据设定的合适的阈值来将前景与背景进行区分。

（3）通过 phase 函数预估图像的局部脊线方向，phase 函数基于数学上的 atan2(y,x)函数，完成后将方向映射到-Π 到 Π 的距离上。

（4）根据局部最大值算法计算局部脊线的频率。

（5）构建八方向的 Gabor Filter 进行 contextual convolution 实现指纹图像增强，对于每个像素，找到和这个像素的脊线方向最接近的 Filter 后将相应卷积之后的结果得出，合并到最后的结果。

（6）通过细化函数将脊线细化后再进行二值化，为下一步特征点提取做准备。

（7）根据八邻域 crossing number 算法得出两种形式的特征点。

（8）去除边缘的错误特征点得到特征点图。

（9）对每一个特征点按照方向计算算法计算方向。

（10）基于 MCC 技术计算每一个特征点对生成的所有小立方体的影响，即“柱体中各个小长方体的权值由该结构中的其他特征点共同决定”[1]，即一个特征点生成一个局部结构，利用局部结构进行匹配。

（11）根据距离匹配算法对两张指纹图片的匹配度进行打分。

（12）对上述每一步的中间过程图像进行保存，用于交互式界面的演示分析。

1.4 技术优势分析（与其他方法相比有什么优势）

1.4.1 基于 Sobel 算子、积分模板和 Garbo 滤波的多重图像增强

在影响因素众多的指纹识别环境中，硬件模块返回的图像一般呈现出较为模糊，粘连的特点，无法得到线性特征明显的脊线信息，因此为了便于提取脊线用于后续计算，首先要对图像进行边缘提取和增强。而常见的图像边缘提取算法例如 Roberts 算法以及图像增强算法例如直方图均衡化在此处均效果不佳。

为了解决这个问题，我们在系统中，使用了综合、改进了多种不同的算法，在不同的阶段，对图像进行了多重边缘提取和增强。这些算法分别是：基于双方向的 Sobel 算子的边缘提取算法（详见本文 2.3.3、2.3.4），以及积分模板和 Garbo 滤波的图像增强算法，从而获得了一个比较满意的最终结果。

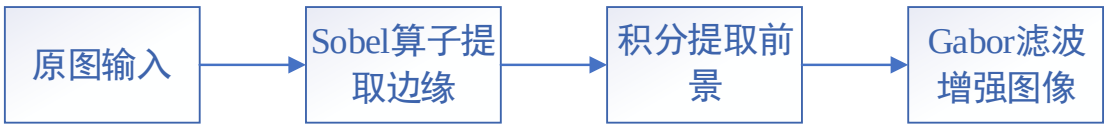


图 1-4-1 图像增强过程

(a)原图　　(b) sobel 边缘提取图　　(c) 积分模板前景图　　(d) Gabor 滤波增强图

图 1-4-2 图像增强

1.4.2 基于 Phase 算法的脊线方向计算

脊线方向的计算在 Gabor 滤波中起着至关重要的作用，因此需要准确率较高的脊线方向提取算法。

位置(x,y)的局部脊线的方向在-180°-180°之间，是脊线和水平轴形成的，

对于每一个像素点，都可以根据其梯度 Gx,Gy,计算得出梯度方向的正交方向，并使这个角度在一个预定的窗口中进行平均后得到方向。

图 1-4-3 脊线方向图

1.4.3 基于 MCC 技术的指纹图像匹配

基于特征点局部拓扑结构(local minutia topologic structures) 的指纹匹配算法是当前研究的热点。特征点局部拓扑结构具有对噪声点和非线性形变更好的适应能力。

2000 年，X. Jiang 提出了特征点 k-近邻结构，2004 年，X. Chen 提出了一种特征点局部拓扑结构，2007 年，W. Xu 提出了特征点三角形结构，并通过融合得到更大区域的特征点结构。2009 年，J. Cao 对局部特征点结构进行了扩展，提出了包含更多局部细节信息的星形结构，总结来说，特征点局部拓扑结构增强了对噪声和非线性形变的适应能力，但仍然存在着相似度难以度量的问题。

2010 年，R. Cappelli 等人提出新的特征点柱形编码 (Minutia Cylinder Code) 结构。将特征点局部结构由传统的二维扩展到准三维。

MCC 事实上是对这个“图像”作了高斯滤波，增强其抗噪能力。本文通过对特征点柱形编码进行分析，对 MCC 结构进行了显式的数学表达，定义了新的 MCC 距离函数，通过一系列的数学推导，得到了简化的距离度量表达式。从而避免了直接求解特征点 MCC 结构，在保证匹配准确率的同时，有效地降低了时间复杂度和空间代价。针对稀疏特征点，在匹配过程结束后，引入了基于最小移动误差的非线性形变模型，找回在匹配过程中丢失的稀疏特征点对。

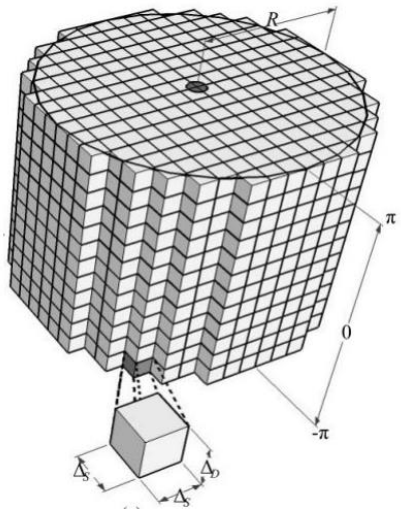


图 1-4-4 MCC 示意图

1.4.4 全步骤图像展示的交互界面

此项目不仅可以用于普通的指纹识别场所，还可以用于交互式学习平台，用户可以选择是否打开交互方式来获取不一样的服务，若用户打开了交互式选项，则界面将会输出指纹识别匹配算法的每一步处理过程，帮助用户了解增长算法知识的同时，增加对指纹识别技术的信任度；当然用户可以选择关闭交互式选项而将其当成一种安全准确的指纹识别系统。

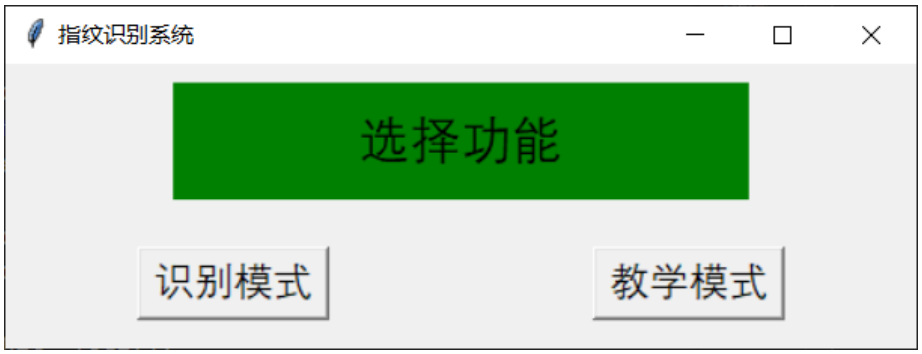


图 1-4-5 交互界面示意图

二、技术路线及实现方案

2.1 开发环境

- 开发平台：Windows10

- 开发语言：python
- 开发工具：AS608 指纹模块、OpenCV 3.4.0、tinker

2.2 技术方案

我们通过根据不同位置的脊线方向采用不同的模板进行 contextual convolution，提升了图像增强效果，提取的脊线更加精确。

同时采用 MCC 圆柱形编码算法，结合特征点的距离和方向构造指纹特征表征，提高了匹配精度。具体流程图见图 2-2-1。

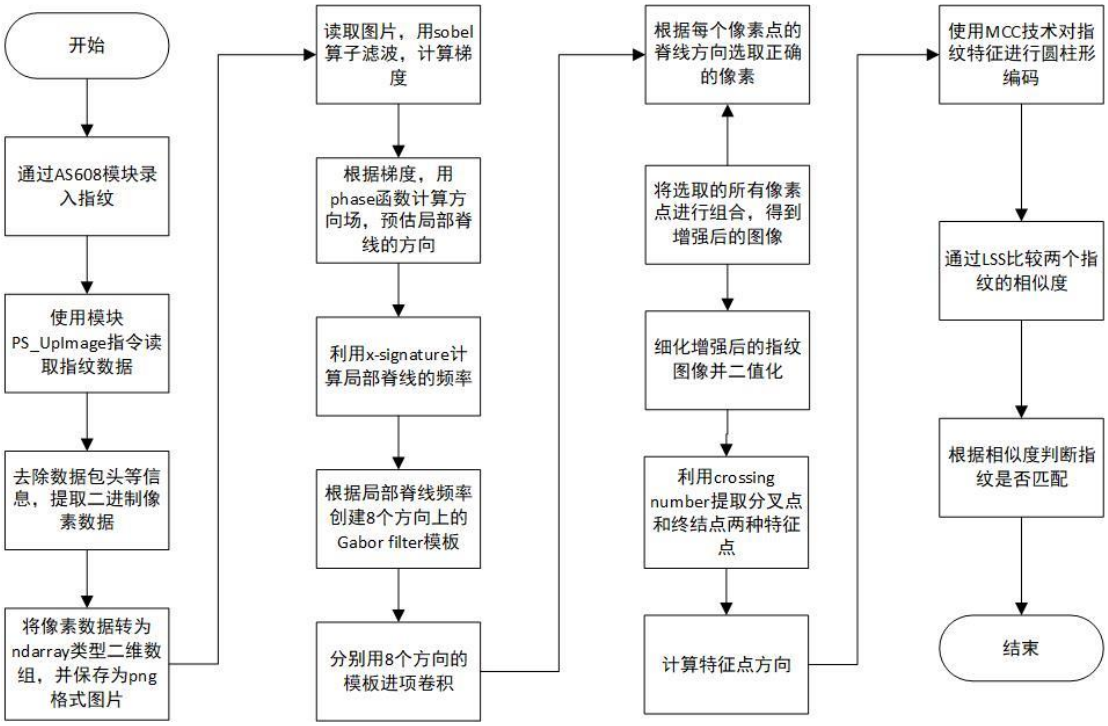


图 2-2-1 项目流程图

2.3 核心技术（把前面 1.3 提到的技术详细写一写）

2.3.1 AS608 模块录入指纹

ATK-AS608 指纹识别模块是 ALIENTEK 推出的一款高性能的光学指纹识别模块。集成了指纹识别算法，能高效快速采集图像并识别指纹特征。在本次实验中我们仅使用 AS608 采集指纹图像的功能。

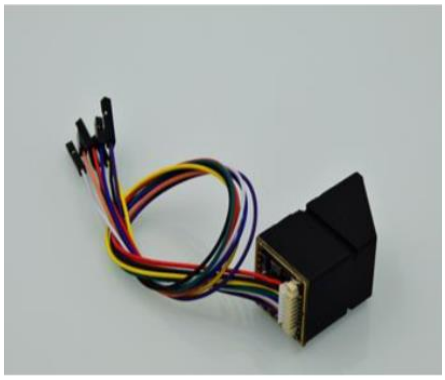


图 2-3-1

AS608 模块与主机之间通过串口通信。模块始终处于从属地位（Slave mode），主机（Host）需要通过不同的指令让模块完成各种功能。主机的指令、模块的应答以及数据交换都是按照规定格式的数据包来进行的。

首先打开串口，指定串口号和波特率。然后使用录入指纹指令 PS_GetImage 录入指纹。其指令代码为 01H，指令包格式如下：

2 bytes	4bytes	1 byte	2 bytes	1 byte	2 bytes
包头	芯片地址	包标识	包长度	指令码	校验和
0xEF01	xxxx	01H	03H	01H	05H

图 2-3-2

芯片地址采用默认地址 0xFFFF。向模块发送上述格式的指令包，然后将手指置于指纹采集器上，模块将探测指纹并采集指纹数据，采集到的指纹图像将存于 ImageBuffer 中。模块采集数据后将会返回一个应答包，应答包格式如下：

2 bytes	4bytes	1 byte	2 bytes	1 byte	2 bytes
包头	芯片地址	包标识	包长度	确认码	校验和
0xEF01	xxxx	07H	03H	xxH	sum

图 2-3-3

其中确认码表示录入指纹结果，确认码=00H 表示录入成功，确认码=01H 表示收包有错，确认码=02H 表示传感器上无手指，确认码=03H 表示录入不成功。

2.3.2 上位机读取指纹数据并存储为图像

通过 PS_GetImage 指令已经将指纹图像录入到图像缓冲区 ImageBuffer，接着需要通过上传图像指令 PS_UpImage 将缓冲区图像传给主机。该指令的指令代

码为 0aH，指令包格式如下：

2 bytes	4bytes	1 byte	2 bytes	1 byte	2 bytes
包头	芯片地址	包标识	包长度	指令码	校验和
0xEF01	xxxx	01H	03H	0aH	000eH

图 2-3-4

模块接受到指令后将发送应答包和数据包给上位机，应答包确认指令执行结果，数据包中包含指纹像素数据。应答包格式如下：

2 bytes	4bytes	1 byte	2 bytes	1 byte	2 bytes
包头	芯片地址	包标识	包长度	确认码	校验和
0xEF01	xxxx	07H	03H	xxH	sum

图 2-3-5

其中确认码=00H 表示可以接收后续数据包，确认码=01H 表示收包有错，确认码=0eH 表示不能接收后续数据包。

应答包后接受数据包，数据包分为一般的数据包（包标识=02）和结束包（包标识=08），结束包表示这是最后一个数据包，数据传输结束。数据包和结束包格式如下：

02 数据包格式：

字节数	2bytes	4bytes	1 byte	2 bytes	N bytes…	2 bytes
名称	包头	芯片地址	包标识	包长度	数据	校验和
内容	0xEF01	xxxx	02			

图 2-3-6

08 结束包格式：

字节数	2bytes	4bytes	1 byte	2 bytes	N bytes…	2 bytes
名称	包头	芯片地址	包标识	包长度	数据	校验和
内容	0xEF01	xxxx	08			

图 2-3-7

通过 PS_UpImage 指令读取到了指纹数据，但该数据包含了数据包头等无用数据，只有每个数据包中数据部分才是真正的指纹图像数据。因此需要解析此数据，提取出真正的指纹数据。观察数据包格式，发现其包头+芯片地址始终是 0xEF01FFFF02 或 0xEF01FFFF08，将此作为需要匹配的十六进制串。从前往后按字节遍历整个数据，每次连续完整地匹配到包头串，则表明数据包头得到确认，之后的字节就是包长度和真正的图像数据。然后根据包长度，向后读取相应长度

的字节保存下来。接着匹配下一个包头，直到遍历整个原始数据后，所提取到的数据即是真正的指纹图像数据。

最后将提取的指纹数据转换成 naddary 形式，通过 cv2.imwrite()函数，即可将 AS608 模块采集到的指纹保存成图像到主机上。

2.3.3 基于 Sobel 算子的图像分割

图像分割就是把图像分成若干个特定的、具有独特性质的区域并提出感兴趣目标的技术和过程。在此处，我们希望通过图像分割技术将图像中的前景（指纹所在区域）与背景分离。前景的特点是存在许多曲线条纹，背景的特点是一片空白，因此，我们采用 Sobel 算子来计算图像灰度的近似梯度，梯度越大越有可能是边缘。

首先，利用 Sobel 算子分别计算水平和垂直方向的梯度（G_x和G_y），结果如下所示：

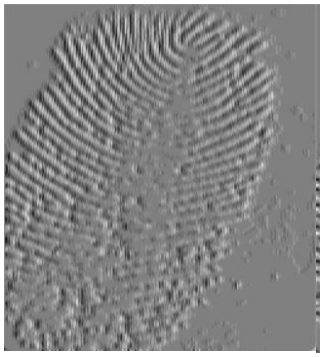


图 2-3-8 x 方向

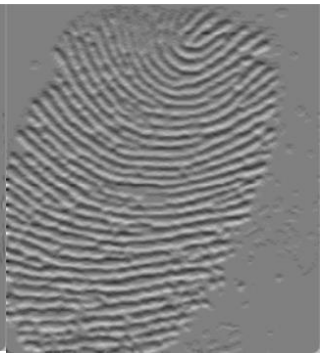


图 2-3-9 y 方向

在图像的每一像素上，结合以上两个结果求出近似梯度：

$$G = \sqrt{G_x^2 + G_y^2} \tag{1}$$

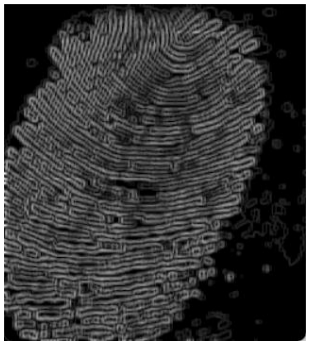


图 2-3-10 G

接下来通过盒式滤波器对其进行模糊处理：

图 2-3-11 模糊处理

最后通过简单的阈值分割技术将其分割：

图 2-3-12 阈值分割

2.3.3 指纹局部脊线方向估计

指纹图像中(j,i)处的局部脊线方向是角度$\theta_{j,i}\in[0,\pi]$，指纹脊线与水平轴在以(j,i)为中心的任意小邻域中形成。

对于每个像素，我们将从梯度$[G_x,G_y]$中估计局部方向，我们已经在 2.3.1 图像分割步骤中计算了该梯度。在每个像素点上，脊线方向θ被估计为与梯度方向正交，在23×23的窗口W上平均：

$$\begin{cases}\theta=\dfrac{\pi}{2}+\dfrac{\text{phase}(G_{xx}-G_{yy},2G_{xy})}{2}\\ G_{xx}=\sum_W G_x^2\\ G_{yy}=\sum_W G_y^2\\ G_{xy}=\sum_W G_xG_y\end{cases}\tag{2}$$

对于每个方向，我们还将计算一个置信值s，它衡量W中所有梯度共享相同

方向的程度：

$$s=\frac{\sqrt{(G_{xx}-G_{yy})^2+(2G_{xy})^2}}{G_{xx}+G_{yy}} \tag{3}$$

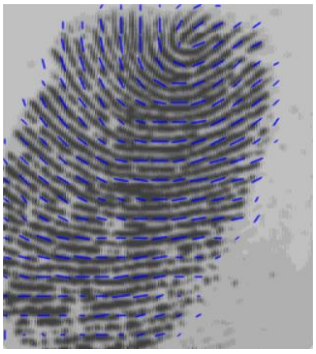


图 2-3-13 脊线方向

2.3.4 指纹局部脊线频率的估计

图像中位于(j,i)处的局部脊线频率$f_{j,i}$是沿着以像素点(j,i)为中心并与局部脊线方向$\theta_{j,i}$正交的假设段上每单位长度的脊数。

接下来以下图这一段为例，计算脊线频率。

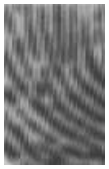


图 2-3-14 局部指纹

然后从该区域计算 x 特征，并将脊线周期估计为两个连续峰值之间的平均像素数。不过在这之前，我们需要对该区域进行平滑操作以减少噪声，这里我们采用的均值滤波 blur。得到的 x 特征图如下图所示：

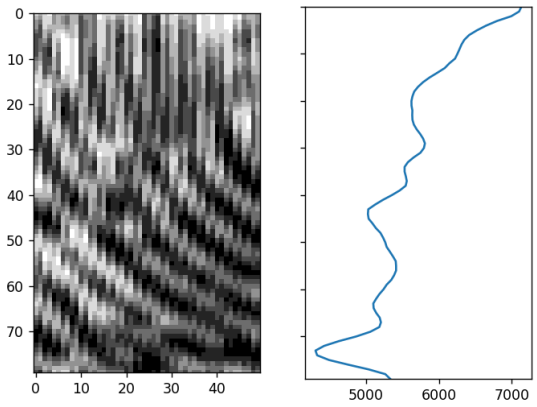


图 2-3-15 累计分布

接着由上图得出 x 特征值局部最大值的索引：

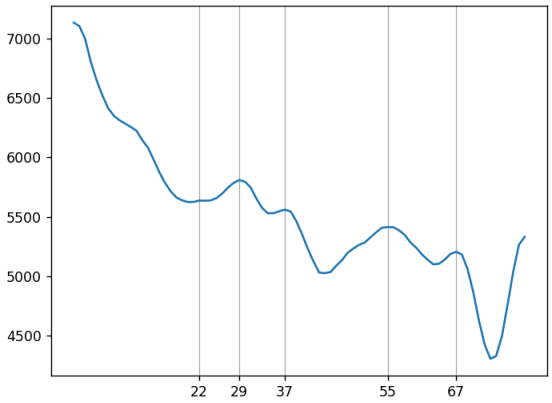


图 2-3-16 累计分布 2

然后由该图计算连续峰之间的所有距离，这里得到的结果为：

7 8 18 12

将脊线周期估计为上述距离的平均值：

$$\hat{f} = \frac{7+8+18+12}{4} = 11.25$$

2.3.5 基于 Gabor 滤波的图像增强

为了增强指纹模式，我们将使用一组 Gabor 滤波器对图像进行 contextual convolution。为了便于计算，我们假设脊线频率处处相等，从而使得我们所有的过滤器都将具有相同的周期，唯一的参数是方向的数量。

在进行滤波之前，我们需要先创建一个滤波器组，这里我们创建了角度从 0° 到 180° 的 8 个滤波器：

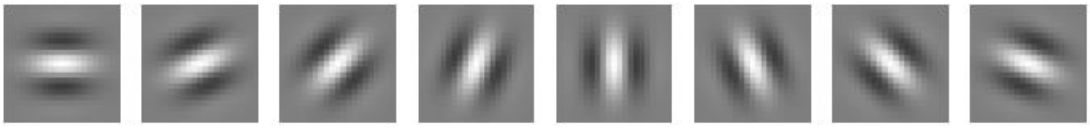


图 2-3-17 滤波器组

然后使用每个过滤器过滤图像：

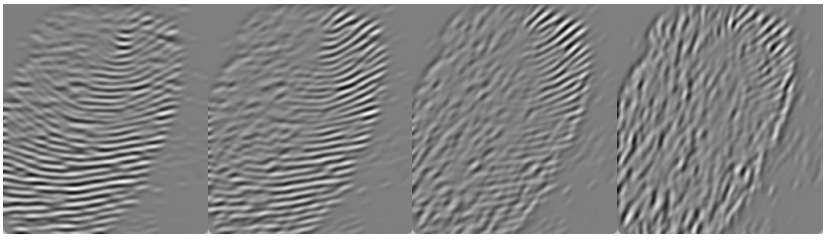


图 2-3-18 过滤结果图

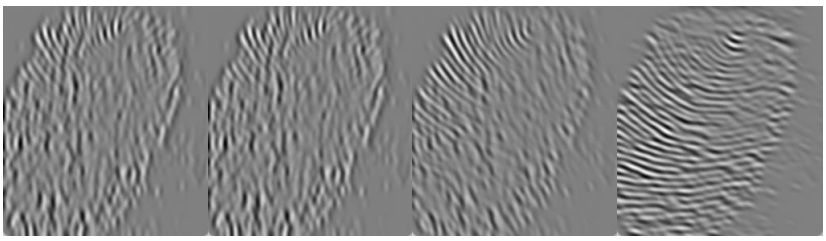


图 2-3-19 过滤结果图（续）

对于每个像素，在滤波器组中找到最接近的方向的索引，对每个像素取对应的卷积结果，组装成最终结果，最终转换为灰度图：

图 2-3-20 指纹图像增强

2.3.6 检测特征点位置

为了便于寻找特征点，我们需要对增强的脊线进行二值化和细化，从而获得脊线的骨架。对 2.3.4 中得到的增强图像进行二值化，得到的二值图像如下：

图 2-3-21 二值图像

对获得的二值化增强图像进行细化操作，得到脊线骨架图：

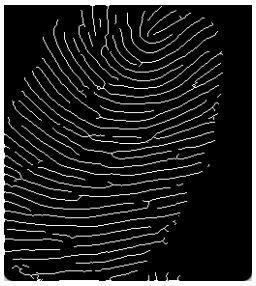


图 2-3-22 脊线骨架图

然后，对于骨架的每个像素p，我们使用交叉数$cn(p)$来表示其 8 邻域中从黑

色到白色像素的转换数。像素 8 邻域可以表示为如下形式：

$$\begin{matrix} v[0] & v[1] & v[2] \\ v[7] & p & v[3] \\ v[6] & v[5] & v[4] \end{matrix}$$

由此我们可以得到交叉数$cn(p)$的计算公式：

$$cn(v)=\sum_{i=0}^{7}\begin{cases}1 & \text{if } v[i]<v([i+1]\bmod 8) \\ 0 & \text{otherwise}\end{cases} \tag{4}$$

为了更加高效地计算所有交叉数，我们可以构建一个 3x3 的卷积核，通过将每个像素视为一位，从而把每个可能的 8 邻域转换为一个可以用字节表示的特定值。卷积核结构如下：

$$\begin{matrix} 1 & 2 & 4 \\ 128 & 0 & 8 \\ 64 & 32 & 16 \end{matrix}$$

然后，我们可以构建一个查找表，将每个字节值$[0,255]$映射到相应的交叉数，从而实现邻域到交叉数的快速变换。我们所需要的特征点是交叉数为 1 或 3 的坐标，其中交叉数为 1 的称为终结点，为 3 的称为分叉点。每个特征点都存储为一个元组(x,y,t)，若$t = true$则说明该点是终止点。

结果如下图所示，其中蓝色代表终止点，红色代表交叉点；左图代表在原图上画点，右图则是在骨架图上画：

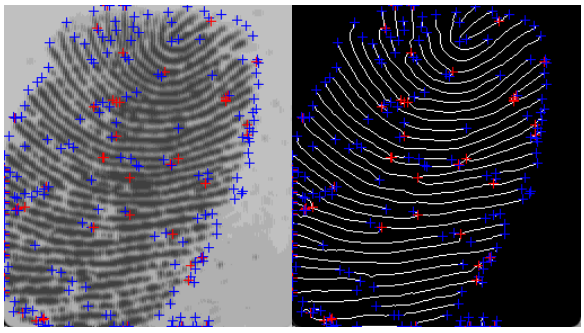


图 2-3-23 特征点

从上图中我们可以发现，在指纹区域的边界附近检测到了许多错误的特征点——这实际上并不是真正的终结点，而是指纹信息未采集完全导致的中断点。我们可以通过计算 mask 的距离变换并选择一个阈值来去除它们，以便排除太靠近指纹区域边界的特征点。

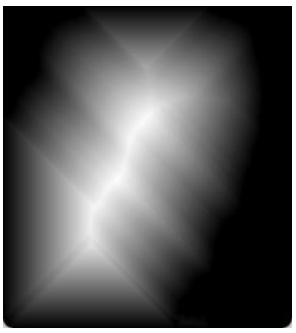


图 2-3-24 距离信息

最终效果如下所示：

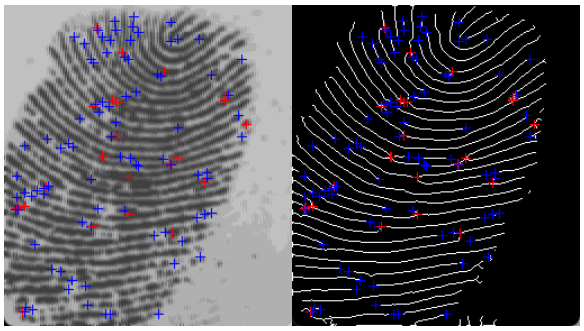


图 2-3-25 特征点

2.3.7 估计特征点方向

对于一个终结点，顺着脊线寻找直到另外一个细节点被找到，或者最远 20 像素被搜索到，这个时候的点和该细节点之间的角度就记录为该细节点的方向。而对于一个分叉点，则需要考虑其三个分支的方向，并最近的两个分支的平均值作为该点的方向。

最终，每个特征点存储为一个元组 (x,y,t,d)，其中如果t是$true$则表示终止点，d是以弧度表示的特征点方向。

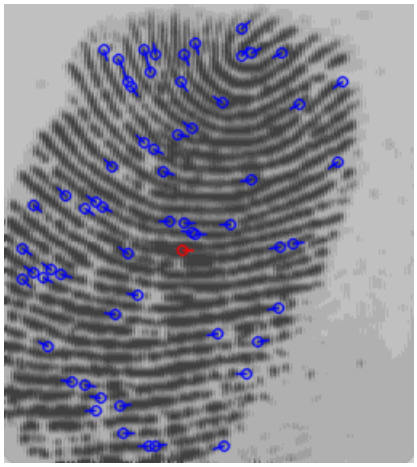


图 2-3-26 特征点方向

2.3.8 基于 MCC 创建局部结构

接下来，从特征点位置和方向开始，我们将创建平移和旋转不变的局部结构，可用于比较指纹而无需预对齐步骤。

我们将使用 Minutia Cylinder-Code 的简化版本：MCC 局部结构可以表示为 3D 结构（圆柱体），其中 base 编码了特征点和高度方向关系之间的空间关系，这里我们将只考虑圆柱的底部，它根据特征点方向旋转并离散成固定数量的单元格。

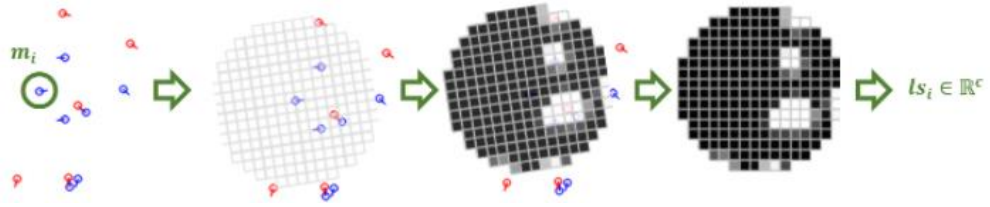


图 2-3-27 Minutia Cylinder-Code

首先，计算通用局部结构的单元坐标，并通过 2.3.6 得到的特征点方向构建旋转矩阵处理坐标，使局部结构在方向上一致。

定义以 0 为均值，以σ_s为标准差的高斯函数：

$$Gs(t)=\frac{1}{\sigma_s\sqrt{2\pi}}e^{-\frac{t^2}{2\sigma_s^2}} \tag{5}$$

和用于限制密集特征点簇的贡献的 Sigmoid 函数：

$$\text{Sigmoid}(t)=\frac{1}{1+e^{-t}} \tag{6}$$

由此，我们可以得到每个细节点m局部结构中每个单元(i,j)的 MCC 权值：

$$C_m(i,j)=\begin{cases}\text{Sigmoid}(\sum_{m_t}C_m^S(m_t,p_{ij}^m)) & \varepsilon(p_{ij}^m)\\ 0 & \text{else}\end{cases} \tag{7}$$

$$C_m^S=Gs(d_s(m_t,p_{ij}^m)) \tag{8}$$

其中，m表示中心细节点，m_t表示m的周围细节点，p_{ij}^m表示长方形(i,j)的中心点，$\epsilon(p_{ij}^m)$有效表示该方格落在指纹分割的前景上，$C_m^S(m_t,p_{ij}^m)$表示周围细节点m_t对长方形(i,j)的坐标贡献值。值得一提的是，在标准的 MCC 算法中，这里不仅仅

是$C_m^S(m_t, p_{ij}^m)$，而是它和角度贡献值$C_m^d(m_t, p_{ij}^m)$的乘积，但由于我们只考虑圆柱的底部，故该项置 1。

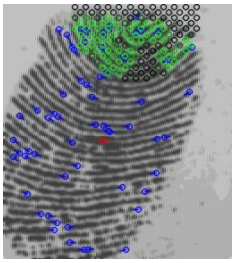


图 2-3-28 特征点的局部结构

2.3.9 基于 LSS 的指纹图像对比

根据 2.3.7，我们从一张指纹图像开始，获得了一个特征点列表和它们对应的局部结构。因为需要对比，这里我们再加载另一个指纹的数据。

这里定义两个局部结构 $r1$ 和 $r2$ 之间的相似度可以简单地计算为一个减去它们的归一化欧几里德距离：

$$1-\frac{\|r_1-r_2\|}{\|r_1\|+\|r_2\|} \tag{9}$$

计算两张指纹图像中局部结构之间所有归一化欧几里得距离的矩阵后，我们可以通过使用局部相似性排序 (LSS) 技术来比较两个指纹，该技术仅选择 n（这里我们选取 4 个）个最高相似度（即最小距离）的特征点进行比较，并将比较分数计算为它们的平均值。相应特征点对的索引成对存储，并在下图中用于显示结果。

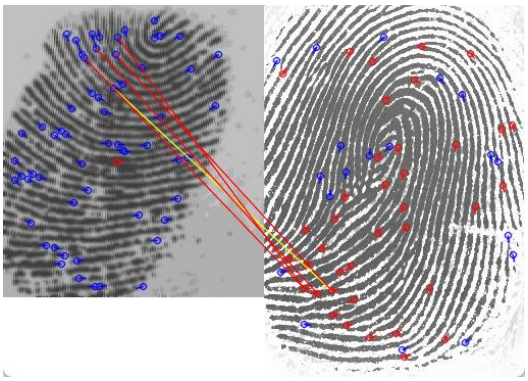


图 2-3-29 指纹比对

三、系统成果展示

3.1 系统展示

3.1.1 主界面展示

图 3-1-1

用户可以选择识别模式直接进行指纹识别；也可以选择教学模式，在这种模式下界面将会输出指纹识别匹配算法的每一步处理过程。

3.1.2 教学模式展示

选择图片

图 3-1-2

指纹分割

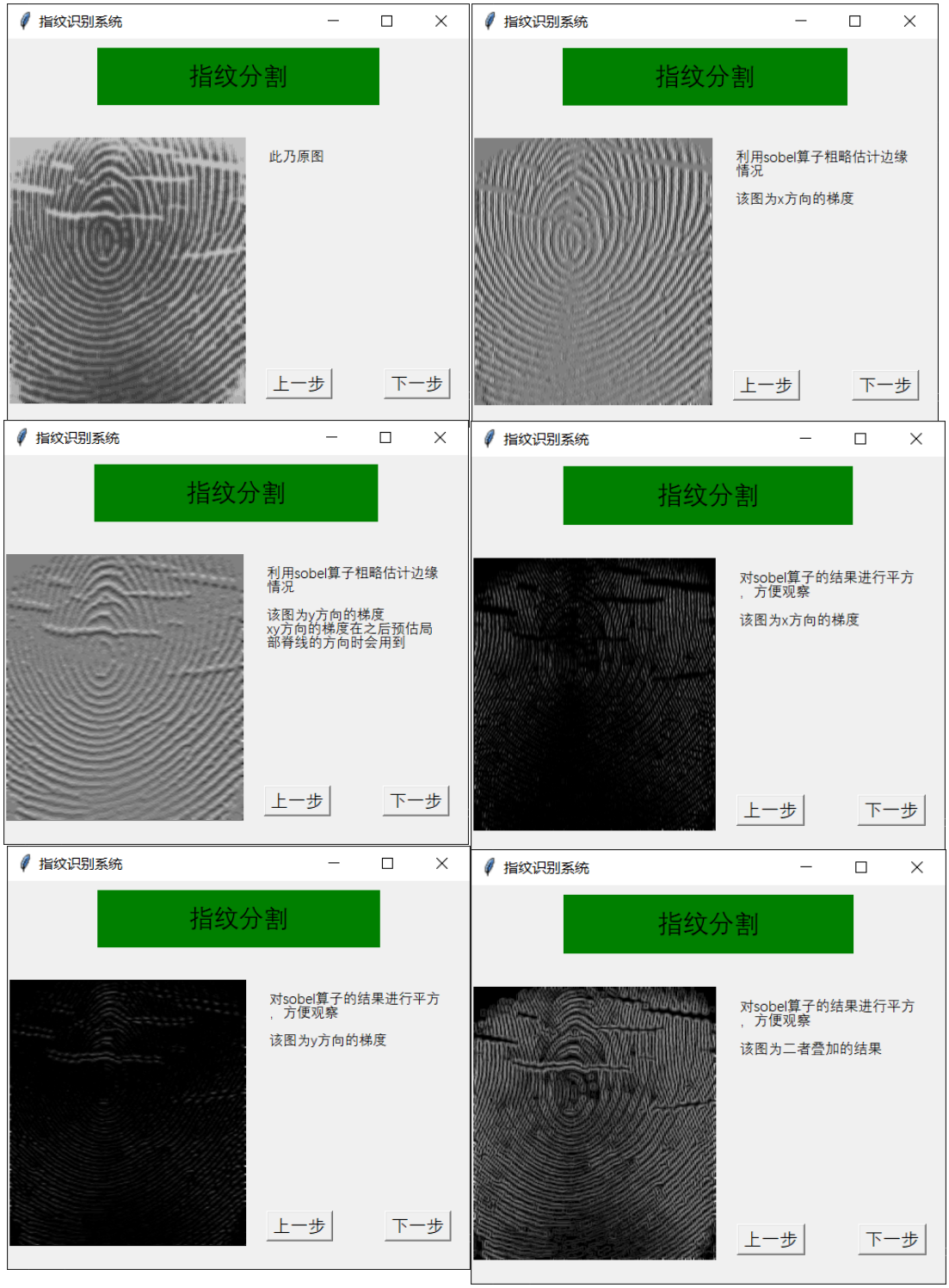
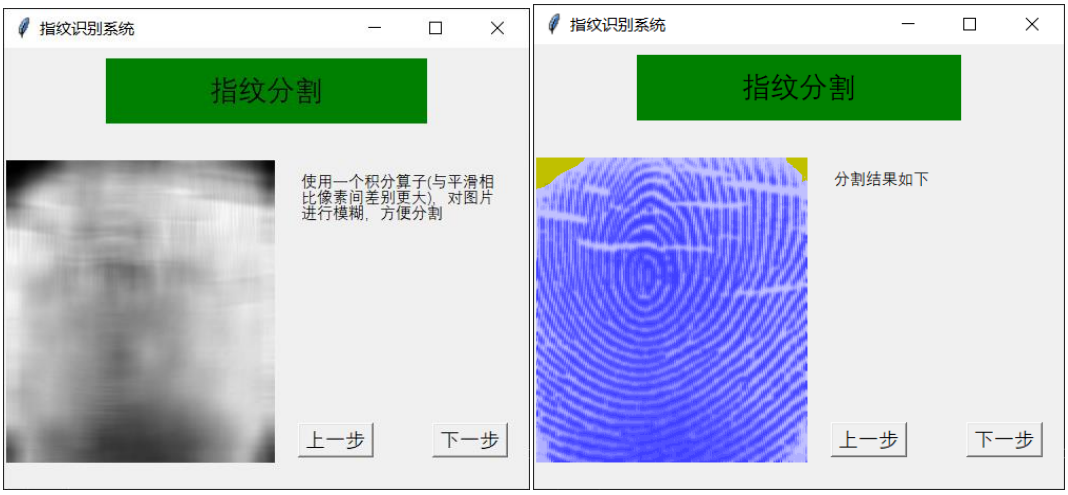

指纹识别系统
指纹分割
此乃原图
上一步
下一步
指纹识别系统
指纹分割
利用sobel算子粗略估计边缘情况
该图为x方向的梯度
上一步
下一步
指纹识别系统
指纹分割
利用sobel算子粗略估计边缘情况
该图为y方向的梯度
xy方向的梯度在之后预估局部脊线的方向时会用到
上一步
下一步
指纹识别系统
指纹分割
对sobel算子的结果进行平方，方便观察
该图为x方向的梯度
上一步
下一步
指纹识别系统
指纹分割
对sobel算子的结果进行平方，方便观察
该图为y方向的梯度
上一步
下一步
指纹识别系统
指纹分割
对sobel算子的结果进行平方，方便观察
该图为二者叠加的结果
上一步
下一步

图 3-1-3



预估脊线方向

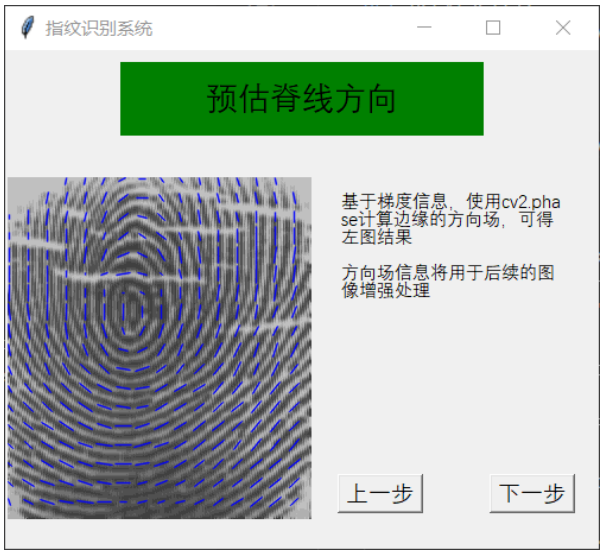


图 3-1-4

预估脊线频率

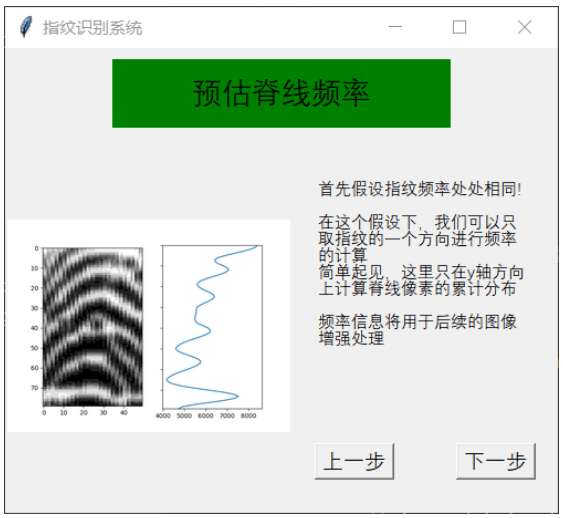


图 3-1-5

指纹图像增强

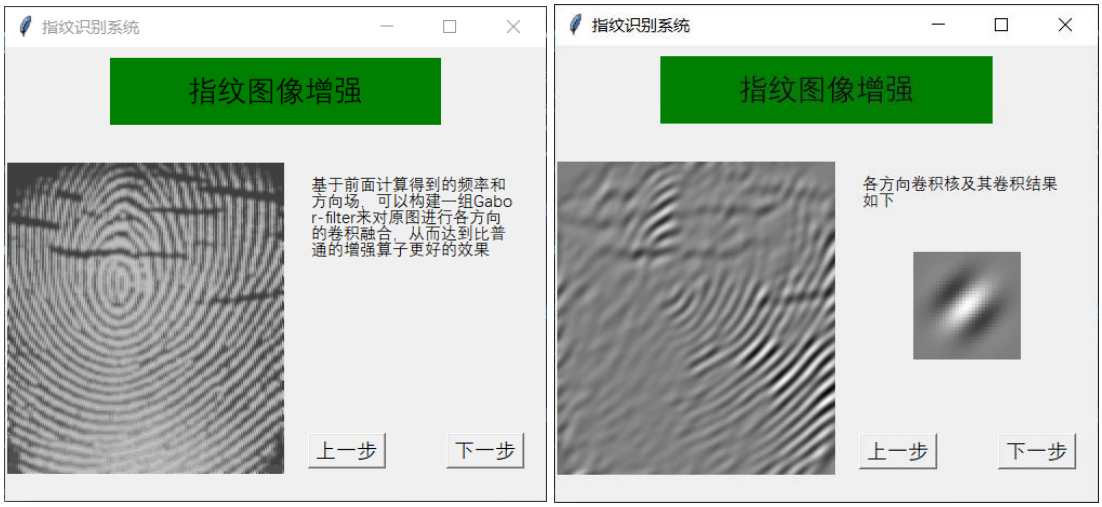

指纹识别系统
指纹图像增强
基于前面计算得到的频率和方向场，可以构建一组Gabor-filter来对原图进行各方向的卷积融合，从而达到比普通的增强算子更好的效果
上一步
下一步
指纹识别系统
指纹图像增强
各方向卷积核及其卷积结果如下
上一步
下一步

图 3-1-6


指纹识别系统
指纹图像增强
对于每个像素，找到与其脊线方向最接近的filter，并将相应的卷积结果合并到最后的图像中，如此组装所有的像素点，得到左图结果
上一步
下一步

特征点提取

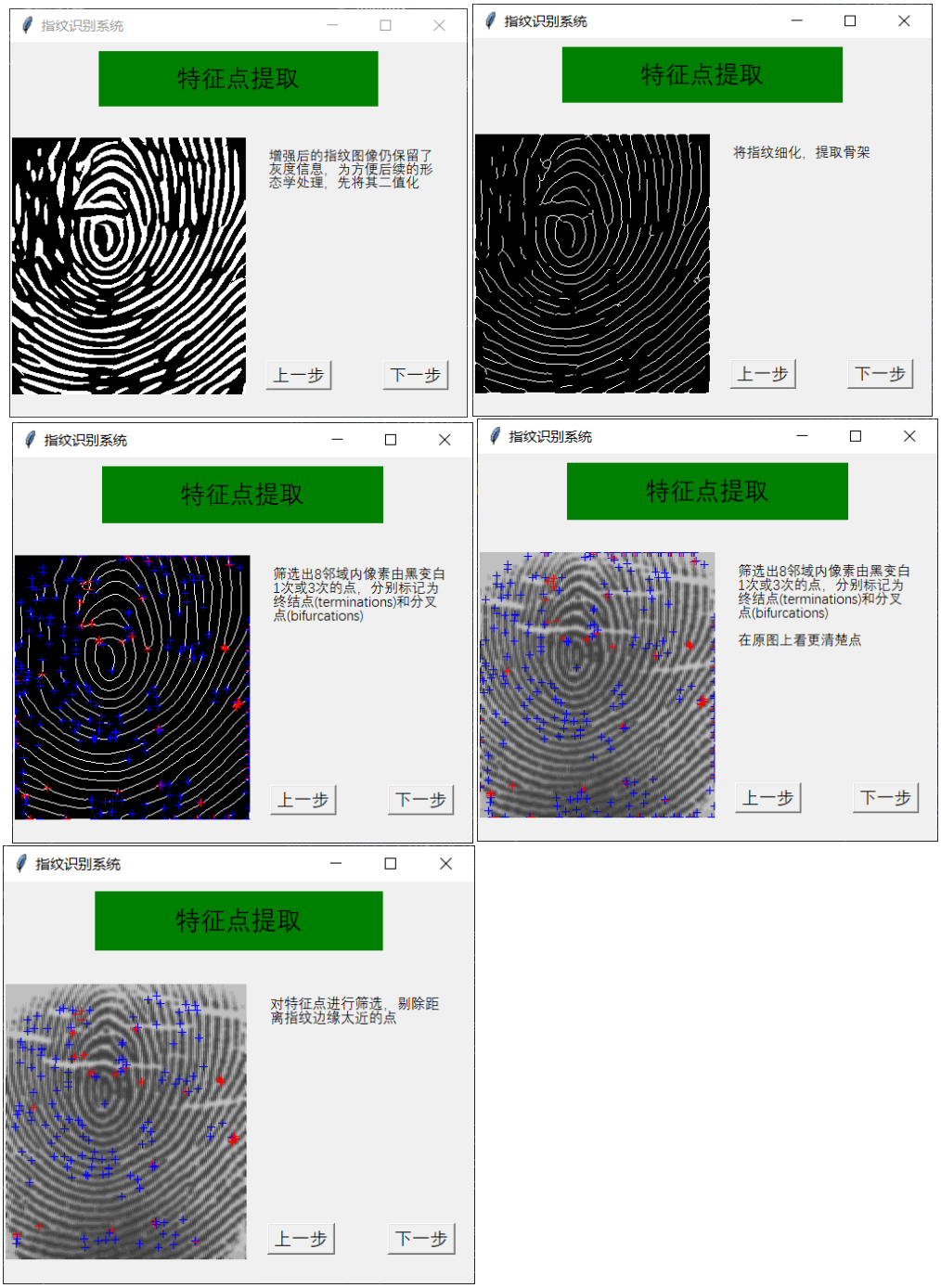

指纹识别系统
特征点提取
增强后的指纹图像仍保留了灰度信息，为方便后续的形态学处理，先将其二值化
上一步
下一步
指纹识别系统
特征点提取
将指纹细化，提取骨架
上一步
下一步
指纹识别系统
特征点提取
筛选出8邻域内像素由黑变白1次或3次的点，分别标记为终结点(terminations)和分叉点(bifurcations)
上一步
下一步
指纹识别系统
特征点提取
筛选出8邻域内像素由黑变白1次或3次的点，分别标记为终结点(terminations)和分叉点(bifurcations)
在原图上看更清楚点
上一步
下一步
指纹识别系统
特征点提取
对特征点进行筛选，剔除距离指纹边缘太近的点
上一步
下一步

图 3-1-7

计算特征点

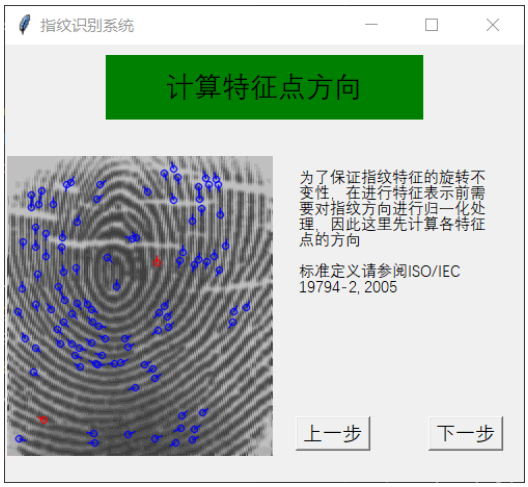


图 3-1-8

Minutia Cylinder Code 编码

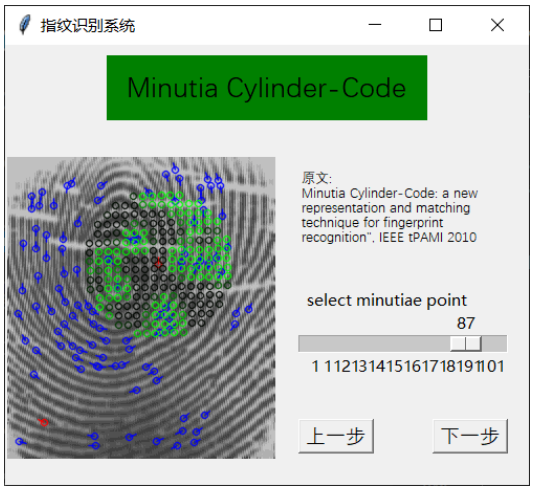


图 3-1-9

特征匹配

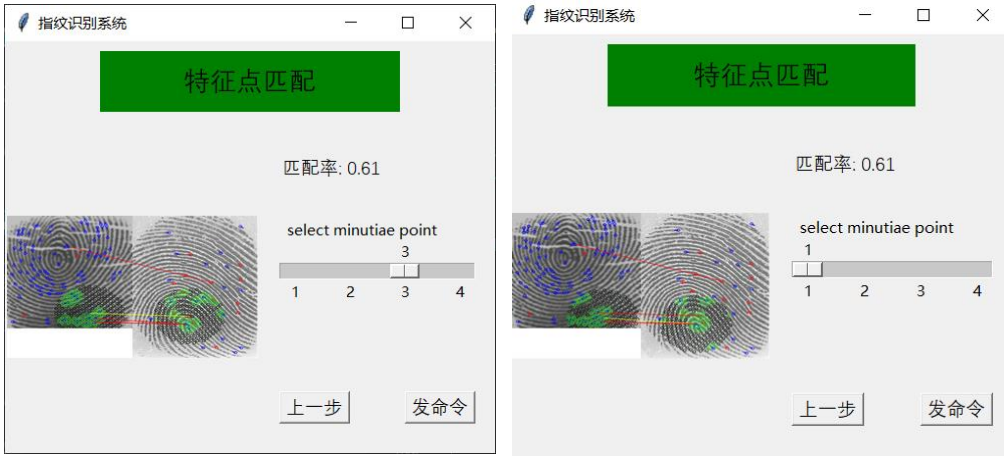


图 3-1-10

3.2 基础场景测试

3.2.1 录入指纹

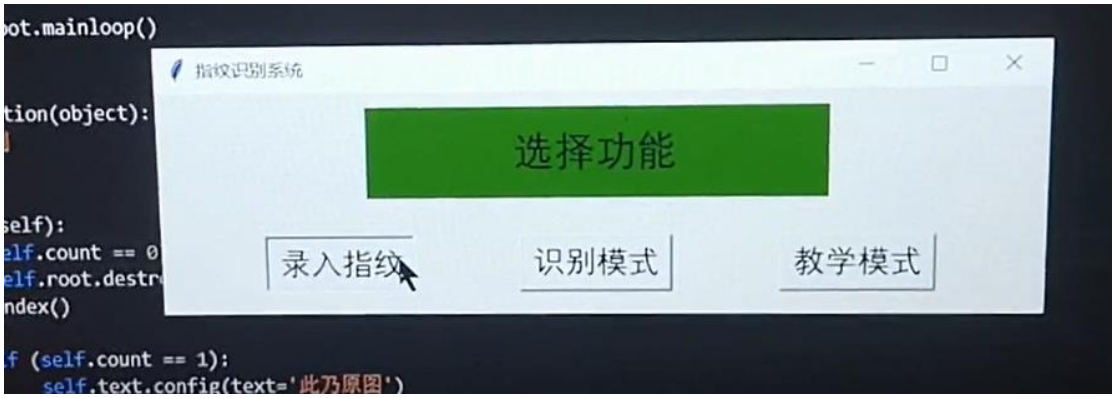


图 3-2-1 录入指纹-界面

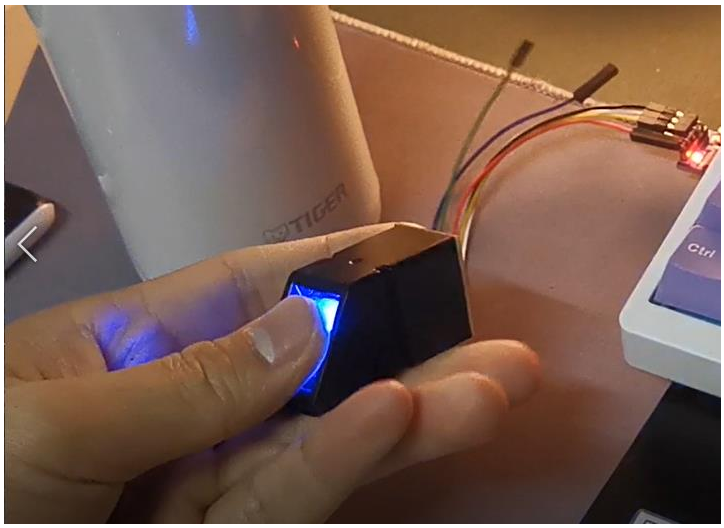


图 3-2-2 录入指纹-操作

3.2.2 指纹识别

图 3-2-3 指纹识别-界面

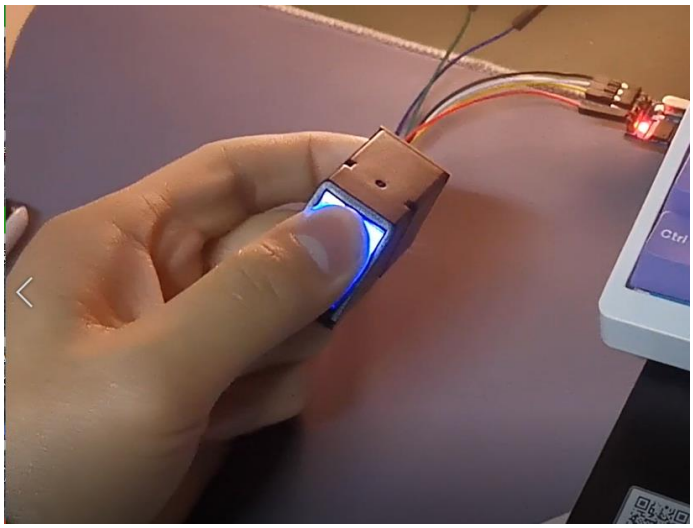


图 3-2-4 指纹识别-操作

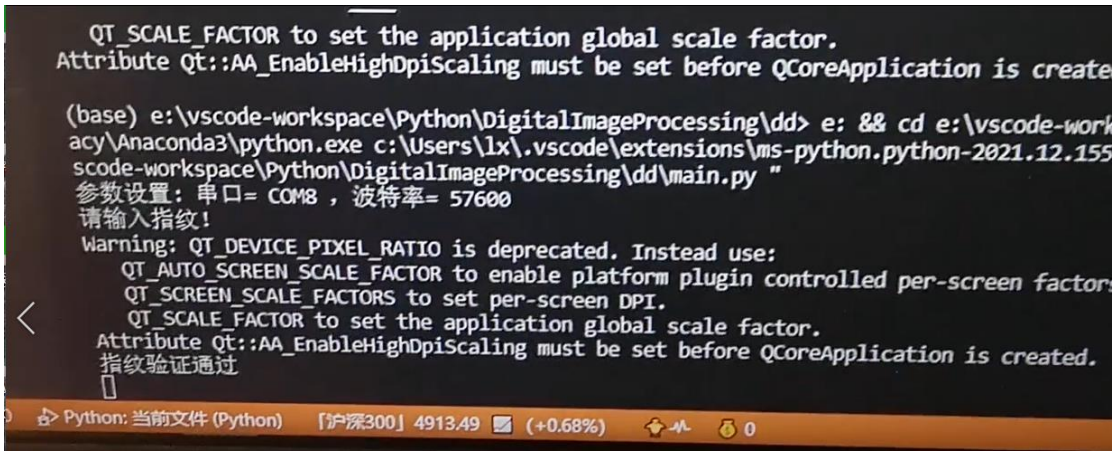


图 3-2-5 指纹识别-结果

六、参考文献

[1]付翔,毛紫微,刘重晋,封举富.构建细节点柱形结构的指纹匹配算法[J].计算机科学与探索,2012,6(07):586-592.

[2] R. Cappelli, M. Ferrara, D. Maltoni. Minutia Cylinder-Code: a new representation and matching technique for fingerprint recognition [J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 32(12): 2128-2141, 2010.

[3] Raffaele Cappelli, Hands on Fingerprint Recognition with OpenCV and Python - IAPR/IEEE Winter School on Biometrics 2021 (hkbu.edu.hk)