基于指纹特征点方向梯度编码和匈牙利算法的指纹匹配分类模型

摘要

本文根据指纹学基本知识和机理建立了基于指纹核心点和细节特征点信息的编码，建立了二分图模式的匹配分类模型，利用图像特征点信息和图像方向场梯度，通过匈牙利算法给出了上述问题结果。

针对问题一，该问题需要设计一种能反映指纹基本特征的编码方式。我们首先进行原始图像的预处理，**通过归一化、分割、Gabor 滤波增强、二值化等方法**，将原始图像转换为便于提取特征信息的清晰图像。考虑到**指纹采集过程发生的旋转现象**，我们建立了**图像旋转校正模型**，通过 OpenCV 自带函数 cvFitellipse2 和 cvBoxPoints 找到图像的最小外椭圆和外接矩形，将图像转换成竖直方向的状态，接着我们应用 **Poincare Index 索引法**找出指纹的中心点，并基于中心点建立二维坐标系。编码方式建立在中心点与细节点的关系上，**分别以细节点到中心点的横轴相对距离与纵轴相对距离，细节点的类型和经过细节点的指纹脊线斜率作为四个维度设计编码**，编码的顺序取决于中心点到细节点的几何距离，**每张图片共选取 20 个特征点，所有编码均控制为 140 个字节大小。**

针对问题二，该问题需要比较不同指纹之间的**相似程度**。根据指纹学知识，指纹的特征取决于**细节点与中心点的相对位置，类型以及其所在脊线切线的方向**。在第一问编码的基础上，通过**特征点和与其最近邻的一个特征点**构建匹配收益矩阵。使用**匈牙利算法**求解二部图之间的最大匹配情况，然后利用匹配结果计算总收益，归一化后指纹图像间的匹配度。1 代表完全匹配，0 代表完全不匹配。**在前六张图像中，其中 2 和 6 匹配度较高为 0.454979，而图像 2 和图像 1 的匹配度较低为 0.112406，其余匹配度在附录中给出，结果符合原始信息**。对该模型进行**灵敏度分析**，发现将节点数由 20 变为 15 或 10 的情况下，结果的平均误差和最大误差变化不大，**说明该结果具有稳定性**。

针对问题三，根据指纹学知识，**中心点和三角点**对于指纹的分类而言是一个重要依据，同时**指纹中心附近的方向场和梯度能准确的反映我们指纹曲线的趋势和方向**，能便于我们更好的进行分类，从以上两个方面，我们利用 **Sobel 算子和高斯滤波**的获取了指纹图像方向场梯度，而**指纹中心点和三角点则就是方向场梯度骤变的交叉点**，然后在用 **poincare 验证和提取**出对应坐标，借助前面的编码模型基础和匹配算法从而对 16 个指纹图像进行了准确分类，**最后结果为两个中心点的当中 7，8 为螺旋型，14 号为斗型，其余单中心点的图像中 1，4，9，15 为弓型，2，3，5，6，10，11，12，13，16 为箕型，并求出了指纹的旋转方向梯度。**

本文的优点在于：1. 将指纹的特征点和相似度匹配分类转换为了二分图之间的最大匹配模型的问题。2. 在指纹分类和匹配上既考虑了特征点的信息还考虑了图像的方向梯度问题。

关键字： poincare 索引　二分图匹配　匈牙利算法　方向场梯度

目录

一、 问题重述

1.1 问题背景

指纹是指人的手指第一节手掌面皮肤上的乳突线花纹，由于指纹具有唯一性、稳定性和可采集性等特点，指纹也被称为人体身份证。指纹识别技术是目前最为可靠的身份认证方法之一，被广泛用于边境检查，贵重物品保管，重要部门安保，网络安全防护等领域中，指纹是应用最普遍，识别率最高，最容易被接受的个人身份认定方法。

目前大部分指纹自动识别方法都是基于细节特征的，即提取细节点作为特征来表征指纹图像，通过这些特征进行识别，指纹的特征可分为三类：全局特征，局部特征和细微特征，且对任一指纹的识别要满足至少两类特征，这种方法具有信息储存量小，识别速度快等优点。

1.2 待解决的问题

1. 给出一种不超过 200 字节的，用来描述相应指纹的基本特征的指纹密码

2. 给出每一幅指纹图像的指纹密码，基于指纹密码的表示给出一种能够比较不同指纹间异同和相似度的方法。

3. 对附件中 16 个指纹进行对比和归类，并给出相应的依据和结果。

二、 模型假设

1. 指纹相对完整，指纹区域内至少包含一个中心点。
2. 假设所给图像是噪声较小，对于图像特征点的提取编码不会出现较大的影响。
3. 尽管由于按压深浅或其他原因导致的图像清晰程度有所不同，但经过图像增强或者修复后，靠近中心点的核心区域细节点缺失较少。

三、 符号说明

符号	含义
$I(i,j)$	像素点 (i,j) 的灰度值
M_i,V_i	像素点灰度值的平均值及方差
$E(i,j)$	滤波后图像的灰度值
T	二值化方法的阈值
$\theta(k)$	封闭曲线上第 k 个点的方向,k 取 1,2,...
$\partial x(i,j)$,$\partial y(i,j)$	像素点在 X,Y 方向的梯度
W_ϕ	滤波器的尺寸

注：表中未说明的符号以首次出现处为准

四、 问题一模型的建立与求解

4.1 问题分析

该问题要根据一般问题，给出指纹识别方法的一般性步骤。针对该问题，首先要对指纹学的基本概念进行了解，例如指纹的 3 大类型和 9 种形态，这种分类方法也被称为二级分类法。

由于指纹的唯一性特性，在设计指纹密码时应以指纹的特异性特征为依据，指纹的特异性特征包括指纹的种类，特征点的数量及类型，脊线的走势及长度等等。为了对指纹的特征点进行编码处理，我们需把提取出的特征向量放在一个字符串中，并根据特征向量的数量和类型给出编码格式，并作出数学解释。

需要注意的是，由于题目提供的指纹图像的大小，深浅，方向不一致，我们首先需要对各个图像进行一致化处理，由于现实中指纹采集的角度和方向不可能完全一致，我们提出的编码方式必须不受图像旋转的影响。

4.2 前期准备

4.2.1 相关概念解释

指纹学是研究掌面肤纹生理特征、纹理结构及其收集、显现、储存、分类与识别的原理和方法的科学. 在介绍我们的方法之前需要对指纹学相关概念作出解释。

指纹的基本特征指纹是指收支正面皮肤凹凸不平的纹路，纹路中隆起的部分是手指真皮向表皮乳突形成的褶皱，又被称为**指纹脊线 (ridge)**，指纹脊线之间的凹陷部分被称为**指纹谷线 (furrow)**, 不同指纹间纹线的数目和走势是存在差异的，且纹线数目，走势不会随年纪的增长而发生显著变化 [6]。

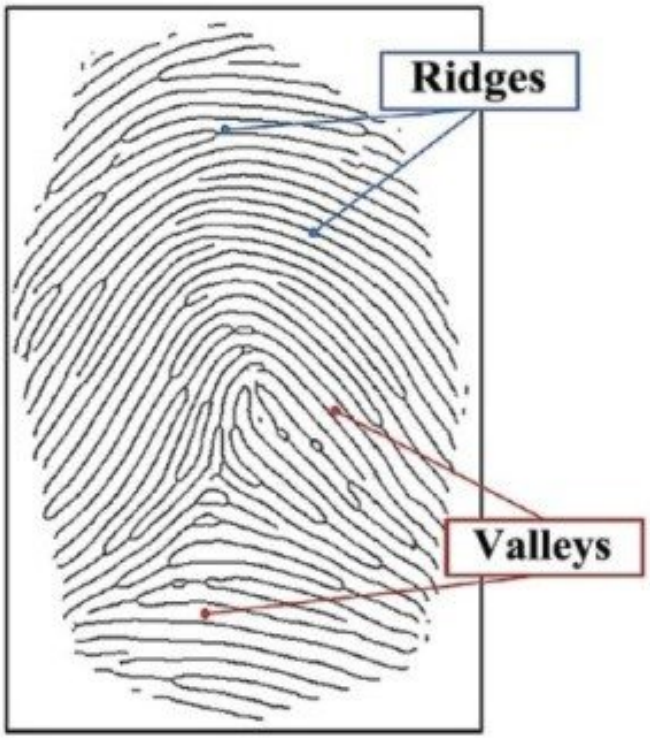


图 **1**　脊线和谷线

指纹的全局特征指纹的全局特征包括人眼直接可以观察到的特征，比如指纹纹型，模式区，中心区，三角区及纹线数目等。最基本的分类方法是按照纹路纹型的分类法，目前比较常见的方法是依据指纹的中心模式区和三角模式区划分的，一般依据指纹形状的不同将指纹分为 3 大类 (**弓形纹 (Arch)**，**箕型纹 (Loop)** 和**螺旋型纹 (Whorl)**),9 种形态 (弧形，帐形，左簸箕形，游簸箕形，环形，螺形，囊形，双簸箕形，杂形）。

指纹的局部特征脊线上有一些能够是指纹图像获得更高区分度和可信度的点，这些点夹杂在脊线与谷线之间，被称作细节点。我们在下图给出了常见的细节点类型，并作出解释。

端点 (ridge ending): 一条脊线的终结点.

分叉点 (bifurcation): 一条指纹纹线在此点分成两条或者更多条的指纹纹路.

湖 (lake): 存在两条或者两条以上的连接两个分叉点的纹线.

独立脊线 (independent ridge): 一条较短但不足以成为一点的纹线.

孤岛 (dot): 一条较短的纹线，可以看做是一点.

毛刺 (hook): 一条指纹文献在此处分出一条较短但不至于成为脊线的纹线.

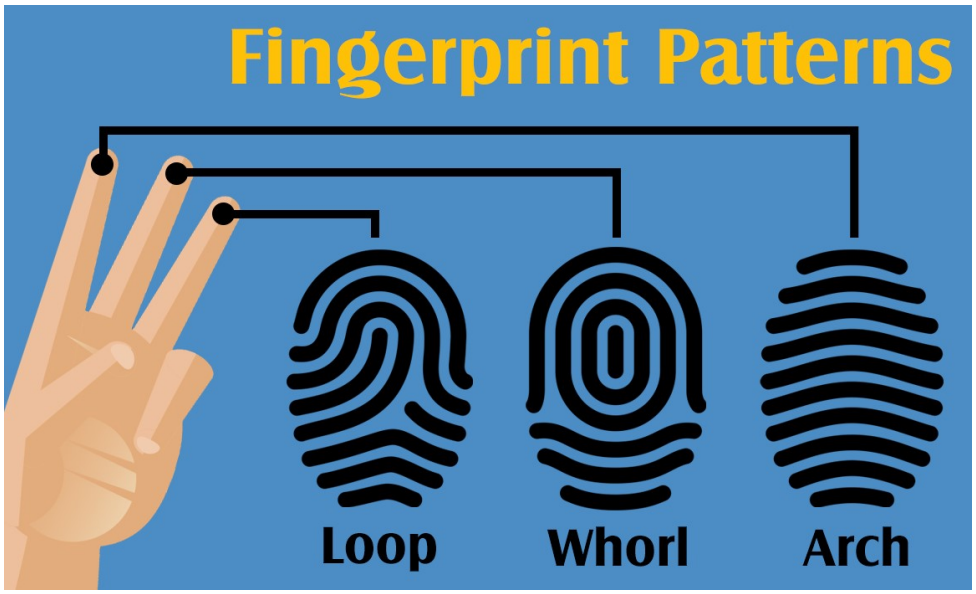


图 **2**　纹型示意图

交叉点 (ridge crossing): 四条纹线汇聚在此点.

需要注意的是，人的指纹中还有很多特殊的细节特征，但是大部分特征是难以提取或者不能作为匹配特征的，在上述细节点中，端点和分叉点的概率是所有局部特征的 80% 以上，所以本文主要关注这两种局部特征.

当采用高分辨率传感器的时候，我们可以提取脊线上更细微的特征，但这样的手段一般适用于及其重要或者安全等级最高的防护设施中，因此，本文对细微特征不做考虑。

4.2.2 图像预处理

在图像的形成和运输过程中，由于受到诸如系统噪声，曝光不足等的影响，获取到的图像可能与原始图像之间存在差异，为了更好的提取指纹图像中的信息，我们需要对原始图像进行预处理。指纹图像处理的主要流程如图4:

归一化和切割

由于指纹按压形成的过程中的手指压力和强度不同，不同图像区域之间的清晰度会有明显差异，归一化调整的是指纹灰度均值和其方差，调整到标准状态并屏蔽不必要的噪声。

1. 我们将指纹分成 W*H 个小块，W,H 分别表示将指纹图像的宽度和长度切成的份数。设图像中的像素点的弧度制为 $I(i,j)$, 归一化后的图像灰度值用 $G(i,j)$ 来表示，灰度平均值和方差分别用 M_i 和 V_i 来表示，则归一化的算法如下：

$$M_i = \frac{1}{WH} \sum_{i=0}^{H-1} \sum_{j=0}^{W-1} I(i,j) \tag{1}$$

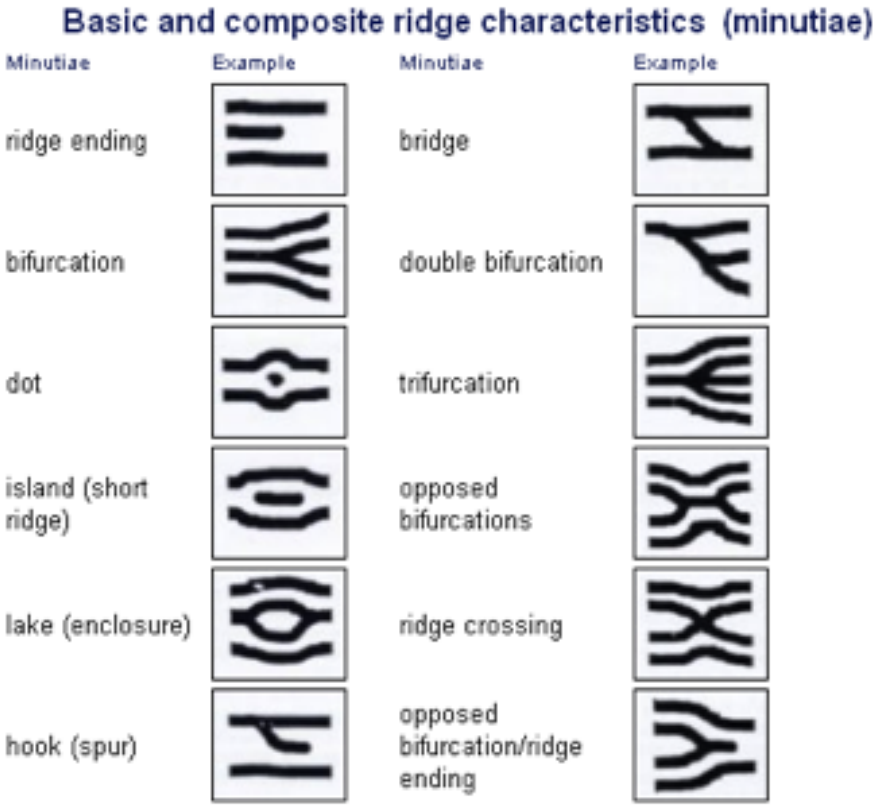


图 **3**　细节点类型示意图

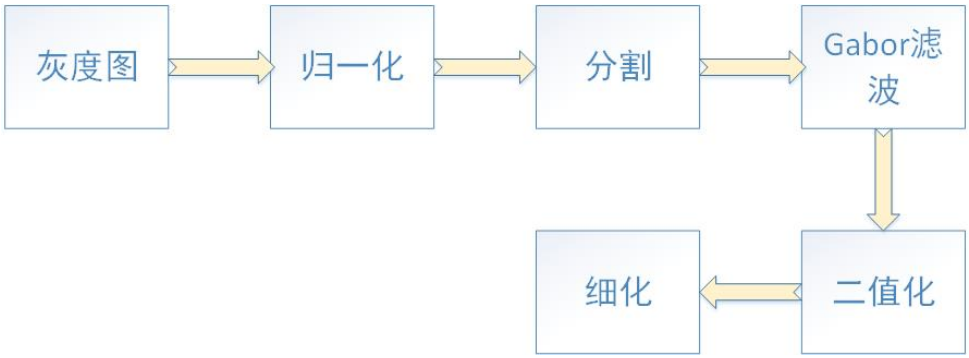


图 **4**　预处理流程图

$$V_i = \frac{1}{WH}\sum_{i=0}^{H-1}\sum_{j=0}^{W-1}(I(i,j)-M_i)^2 \tag{2}$$

2. 指订期望的图像方差和平均值后，算出归一化后的图像灰度值 $G(i,j)$

$$G(i,j)=\begin{cases} M_0+\sqrt{\frac{V_0(I(i,j)-M_i)^2}{V_i}}, & I(i,j)>M_i \\ M_0-\sqrt{\frac{V_0(I(i,j)-M_i)^2}{V_i}}, & I(i,j)<M_i \end{cases} \tag{3}$$

其中，M_0，V_0 为期望的平均值和方差。

分割

分割图像基于块特征的指纹图像分割，本文采用均值方差法，该算法基于背景区灰度方差小，而指纹区方差大的思想，将指纹图像分成块，计算每一块的方差，如果该房差小于阈值则为背景，否则为前景. 具体步骤分为以下三步:

1. 将图像分为 $N\times N$ 的方块。

2. 计算出每一块的均值和方差。

3. 将该块方差与阈值进行比较。

Gabor 滤波增强

Gabor 滤波器作为带通滤波器，因其空域内具有良好的方向选择性及频域内良好的频率选择性而在计算机视觉领域尤其是问你分析方面得到了广泛的应用 [5]。在指纹图像中，对于灰度指纹图像，脊线和谷线在局部的小领域中可被认为是正选波形状，具有一定的频率和方向。

利用 Gaborl 滤波器良好的对方向和频率的选择性滤波是图像增强的一个很好地方案，实验表明，以 Gabor 函数的偶分量实部为模板，脊线与谷线形成的近似正弦波的频率为频波器的频率，以指纹的局部方向构建的滤波器，去噪效果非常好，滤波后的指纹图像的灰度直方图呈现明显的双峰形势。

Gabor 滤波器的形式如下:

$$h(x,y:\theta,f)=\exp\frac{-1}{2}[\frac{x'^2}{\delta_x^2}+\frac{y'^2}{\delta_y^2]}\cos(2\pi fx) \tag{4}$$

其中:

$$x'=x\sin\theta+y\cos\theta$$

$$y'=x\cos\theta-y\sin\theta$$

滤波后图像的灰度值为

$$E(i,j)=\sum_{x=\frac{W}{2}}^{\frac{W}{2}}\sum_{y=\frac{W}{2}}^{\frac{W}{2}}h(x,y:\varphi(i,j),f(i,j))g(i-x,j-y) \tag{5}$$

Gabor 滤波增强后的效果前后对比如图**??**所示：

图 **5**　滤波增强前

图 **6**　滤波增强后

图 **7**　滤波增强前后对比

图像二值化处理

指纹图像二值化处理作为指纹预处理过程的一部分，是进行指纹图像细化处理的基础，二值化后的图像其几何性质只与 0 和 1 的位置有关，不在设计像素的灰度值，使得处理变得简单，这给存储和处理带来了方便。二值化处理后的图像有几个基本要求:

1. 脊线中不出现空白
2. 二值化后的脊线基本保持原来的指纹的特征
3. 指纹的纹线不应有太多的间断和相连
4. 指纹纹线间的间距应大致相同

我们使用自适应阈值二值化的思想，对每块指纹图像，选取的阈值应尽量使该块图像内大于该阈值的像素点数小于该阈值的像素点数。一般灰度图像二值化的变换函数用下列公式表示:

$$f(x)=\begin{cases}1 & \text{x>=T}\\ 0 & \text{x<T}\end{cases} \tag{6}$$

T 为阈值

自适应阈值算法是利用固定阈值算法的思想，根据图像的每一部分的敏感度来调整阈值。本文首先把图像分为若干个方块，每一块根据自己的阈值进行二值化，这种算法充分利用了指纹图中脊线和谷线宽度大致相同的特点，即二值化后黑白像素的个数也大致相同，首先利用固定阈值算法的特点对指纹图像中的每块确定一个大致的阈值，然后再利用自适应的思想对阈值进行准确地调整，即阈值的取值合适时图像时最光滑的，

0-1 之间的转换次数最少。

$$T=\frac{1}{w_2}\sum_{u=i-\frac{2}{w}}^{i+\frac{2}{w}}\sum_{v=j-\frac{2}{w}}^{i-\frac{2}{w}}\times G(i,j)$$

4.3 图像旋转校正特征点编码模型

4.3.1 图像旋转校正

实际上, 在采集指纹图像时，手指与采集仪器之间往往并不是平行接触的，这就导致得到的指纹图像发生旋转倾斜的情况，图像的大小比例也会发生失调，并且无法找到图像的对称轴。yin 和 ye 等提出了一种图像旋转矫正的方法 [1]，利用 OpenCV 自带函数 $cvFitellipse2$ 和 $cvBoxPoints$ 找到最小的外椭圆和矩形。当矩形存在一定的倾斜角度时，无需判断椭圆即可确定指纹发生了偏转；当矩形未偏转时，若椭圆是倾斜的，且其长径远远大于短径，则说明指纹需要进行旋转校正。指纹分类算法流程图为：

图 **8**　旋转流程图

首先我们将指纹图像的最大外接正矩形按照左下，右下，左上，右上四个方向分为 1,2,3,4 标注的 4 个区域，若发生了偏转，则必然会存在某两个对称区域存在相对较多的背景点。我们可以利用这一特点验证椭圆校正方法是否判断正确。另外，若在实际应用中遇到指纹图像几乎充满了整个所在矩形，而只有一个方向存在很多背景点，则无论椭圆是否有偏转都不需要校正图像。

得到了指纹的偏转角度后，调用 OpenCV 库中的 $cvWAaRPAffine$ 函数对整个指纹区域进行仿射变换，进而完成指纹图像的旋转校正，再次基础上计算指纹的方向场，角度范围在 [0 π] 之间，算法流程图如图9：

4.3.2 中心点的提取

在指纹图像中，奇异点是一种不同于细节点的宏观特征，其数目和相对文职可以帮助判断指纹的类别，根据奇异点领域内的脊线分布情况，可分为中心点和三角点 [2]。

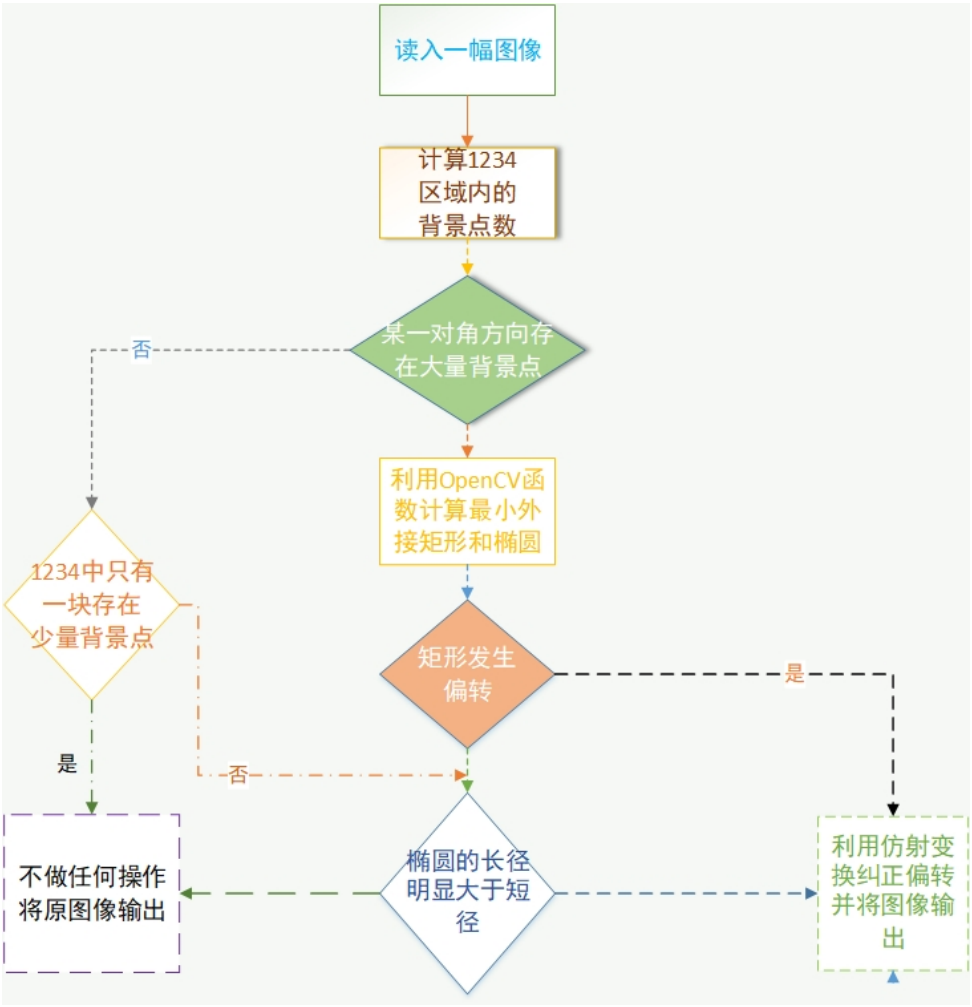


图 **9**　校正偏角算法流程图

奇异点提取是指纹图像处理中的一个重要部分，它可以作为指纹分类和识别过程中的定位基准点，根据其分布特性可以对指纹图像做粗分类，故提取的准确性直接影响到指纹分类和细节匹配的准确率.

本文采用主流的 Poincare 索引法，该方法不会受到图像旋转的影响，即检测过程补语像素点在图像中的绝对位置和方向相关.Poincare 所以发是根据 Poincare 索引值对像素点的性质进行区分，如图10所示，C 是方向图 D 中围绕像素点 (x,y) 的一条封闭曲线，V_1 到 V_9 表示曲线 C 上的像素点的方向矢量. $Poincare$ 索引值的计算公式为:

$$Poincare(x,y)=\frac{1}{2\pi}\sum_{k=0}^{N}\Delta(k)$$

其中:

$$\Delta(k)=\begin{cases}\delta(k), & |\delta(k)|\leq\frac{\pi}{2}\\ \delta(k)+\pi, & \delta(k)\leq\frac{-\pi}{2}\\ \delta(k)-\pi, & otherwise\end{cases}\tag{7}$$

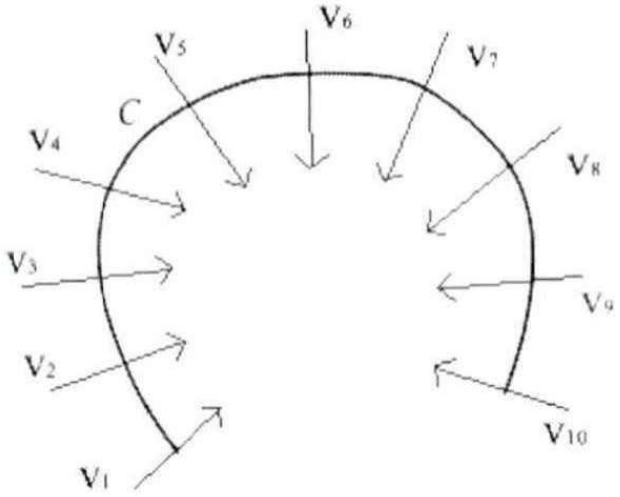


图 **10** **Poincare** 索引法原理图

$$\delta(k) = \theta(k') - \theta(k)$$

$$k' = (k+1) mod N$$

式中，$\theta(k)$ 的意义为编号为 k 的点的方向值，N 的意义为组成曲线 C 的点的数量

本文选取了 3×3 的邻域模板用于检测所有奇异点。我们发现采用 Poincare 索引法求得的所有奇异点中，存在大量靠近图像边缘的伪奇异点，为此我们给每张图像划取了一个大致范围，并去掉范围外的奇异点，这种方法的可取性在于：在获取指纹图像时，由于按压深浅或者采集传导过程中的随机因素，图像边缘会出现大量端点，去除这些奇异点有助于使处理后的图像更加可靠。

4.3.3 编码方法

本题中我们得到的图像为分辨率 640×480 的大小为 300kb 的图像，我们默认以后获得的图像也是同样大小及分辨率的图像或者可以被转化为这种样式的图像。在对整个 16 幅图像进行了图像旋转校正之后，我们得到了近似于正向，即在指纹录取时得到的最规范的图像。在前期的工作中我们已经得到各原始指纹图像二值化后的新图像，同时我们找到了各图像的中心点以及细节点。由此我们提出了一种指纹密码的编码方式：

第一步：以中心点为原点 (用 O_i 表示，i=1,2,...16), 借助与旋转后图像边缘平行的直线，建立二维平面直角坐标系。

第二步：在所有细节点中，选取离中心点最近的 20 个细节点，并画出这些点的大致范围。

第三步：求出所有细节点与中心点在横坐标轴和纵坐标轴上的相对距离，距离为正表示该细节点在中心点的指向正半轴的方向上，距离为负表示该细节点在中心点的指向

负半轴的方向上，由此我们获得了各细节点相对于中心点的距离，也就是它们的相对坐标, 分别用 δx 和 δy 表示

第四步：将细节点分类，端点记为 0，分叉点记为 1.

第五步：依据建立的二维直角坐标系，计算细节点沿脊向方向的切线的斜率，由于图像已被二值化，我们通过以下方法近似处理：对于端点，我们选择同一条脊线上距离该端点最近的两个像素点，以此三点确定一条直线的斜率，通过最小二乘法进行拟合；对于分叉点，我们选择的是两条分叉路径之中偏离中心点的一条直线，求解斜率方法与端点相同。

最小二乘法确定了线性回归方程为：

$$\hat{y} = a + bx$$

其中:

$$\hat{b} = \frac{\sum_{j=1}^{n} x_j y_j - n\bar{x}\bar{y}}{\sum_{j=1}^{n} x_j^2 - n\bar{x}^2} \quad \hat{a} = \bar{y} - \hat{b}\bar{x} \tag{8}$$

推到过程不在此赘述。我们的编码设计有 4 个维度，分别是与中心点在横坐标上的相对距离 x, 与中心点在纵坐标上的相对距离 y, 细节点的类型 t 及经过细节点的脊线切线的斜率。需要说明的是编码的顺序按照距离中心点距离，由小至大的顺序排列，该距离为二维空间几何距离。此方法表示的特征点信息的编码格式:

$$Code = \left(\begin{array}{cccccccccccc} \delta x_1 & \delta y_1 & t_1 & k_1, & \delta x_2 & \delta y_2 & t_2 & k_2, & \ldots & t_{20} & k_{20} \end{array} \right) \tag{9}$$

其中 δx_1 表示存放细节点与中心点在横轴上的相对距离，占用 2 个字节，其中 δy_1 表示存放细节点与中心点在纵轴上的相对距离，占用 2 个字节，t_1 表示存放细节点的类型，占用一个字节，k_1 表示经过细节点的脊线切线斜率，占用 2 个字节。任意图像的指纹密码共保存 20 个特征点，占用 $20\times(1+2+2+2) = 140$ 个字节。

4.4 模型求解

4.4.1 图像腐蚀前后背景分离

录取的指纹在边缘部分常常是模糊不清的，图像同时伴随大量噪声，我们首先对原指纹图像进行了腐蚀与膨胀，其目的在于消除噪声，寻找图像中明显的极大值或极小值区域。

(a) 图像膨胀前　　**(b)** 图像膨胀后

图 **11**　图像膨胀前后对比

4.4.2 图像二值化与特征点提取

接下来我们对图像进行进一步处理，划分了图像包含了大量细节点的核心区域。我们使用 Poincare index 索引法求出了图像的中心点 (以黄色点表示)，并利用二值化后图像的性质，找到了指纹的端点和分叉点 (分别以蓝色点和红色点表示), 下图给出了图片一经过处理后核心区域的图像：

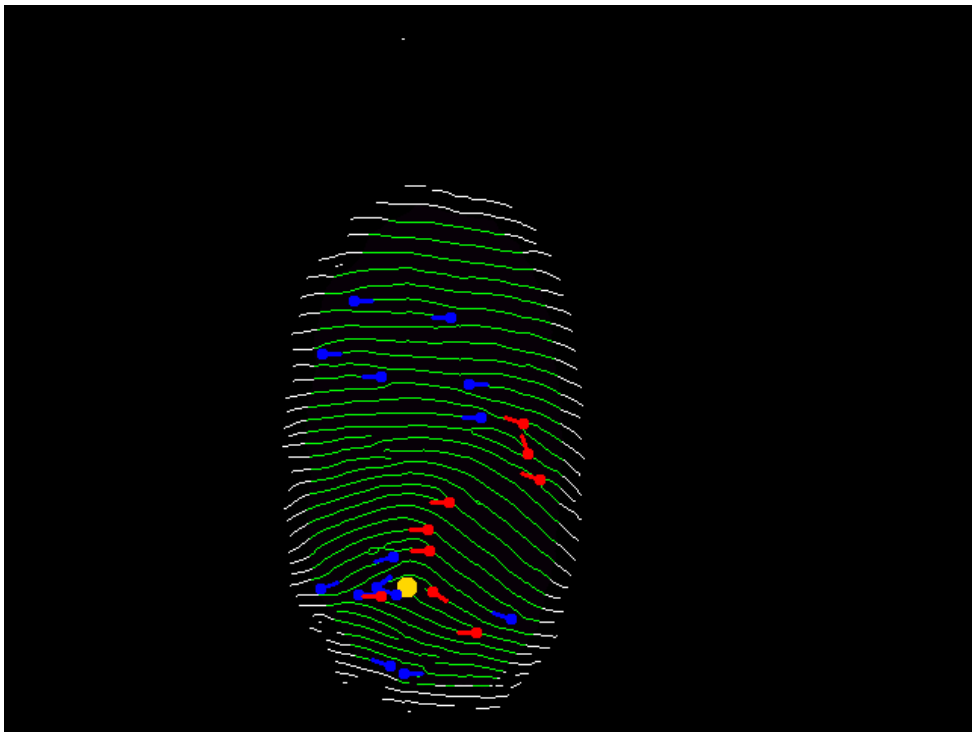


图 **12**　图片一核心区域示意图

接下来利用中心点定位编码方法，我们计算出图像一中各细节点到中心点在横轴与纵轴上的相对位置，和通过细节点的与脊线相切的直线的斜率，细节点的编号按照到中心点的距离从小到大进行排列.

中心点与细节点之间的距离采用欧几里得距离公式

$$D_i = \sqrt{(\delta x)^2 + (\delta y)^2}. \tag{10}$$

表1列出了图一中距离中心点最近的 20 个细节点的特征

表 **1**　第一个图像的特征点相对位置、角度和类别

	row	**col**	**angle**	**class**
0	389	259	-2.81984	0
1	387	283	0.588003	1
2	390	249	3.141593	1
3	384	246	-0.588	0
4	364	257	2.819842	0
5	360	281	3.141593	1
6	389	234	0	0
7	346	280	3.141593	1
8	436	255	-2.81984	0
9	414	312	3.141593	1
10	385	209	-0.32175	0
11	441	264	0	0
12	328	294	3.141593	1
13	405	335	-2.81984	0
14	313	354	-2.81984	1
15	296	346	-1.89255	1
16	272	315	3.141593	0
17	276	343	-2.81984	1
18	245	249	3.141593	0
19	250	307	0	0
20	230	210	0	0
21	206	295	3.141593	0
22	195	231	0	0

4.5 编码结果展示

第一张图片的选取 20 个特征点时的 16 进制编码如下，编码共 140 字节：完整的 16

表 **2**　第一张图片的 **16** 进制编码

16 进制编码
0x05 0x00 0xf9 0xff 0xa1 0x00 0x03 0x00 0x11 0x00 0xdf 0xff 0xe8 0xff 0x0f 0x00 0x4c
0xff 0xda 0xff 0x0e 0x00 0x4c 0xff 0x10 0x00 0xd2 0xff 0x00 0x00 0x1e 0x00 0x2e 0x00
0x4c 0xff 0x39 0x00 0xfe 0xff 0x00 0x00 0xc8 0xff 0x1c 0x00 0x4c 0xff 0x15 0x00 0x45 0x00
0xa1 0x00 0xb9 0xff 0x58 0x00 0xa1 0x00 0x00 0x01 0x01 0x01 0x00 0x01 0x00 0x01 0x00 0x01

张图片编码在附录中给出。

五、 问题二模型的建立与求解

5.1 问题分析

该问题要我们比较题目所提供的 16 幅图像的异同及相似程度。基于问题一所给的方法我们得到了 16 幅图的对应指纹密码，这些指纹密码是不同的特征向量组合而成的字符串。我们可以依据表示两指纹图像中同一指纹特征的两特征向量的异同，以及对图像进行处理时得到的其他特征进行图像的比较。

5.2 特征编码点

若要比较两指纹间的相似程度，则要先匹配两个指纹图像的特征点，然后通过最佳匹配的“契合度”来描述两个指纹的相似性。对于两个特征点本身，其相对与中心点的位移以及所在脊线切线的角度越接近，则越有可能配对。我们定义给定两个指纹中任意两个特征点的配对收益 $profitMatrix$，它又两部分加和构成，$geometryProfitMatrix$ 衡量两点本身的几何位置和所在脊线切线的角度的接近程度，$neighborProfitMatrix$ 衡量两点最近邻居之间的几何位置和所在脊线切线的角度的接近程度。

$$profitMatrix_{i,j} = sameClass_{i,j}(geometryProfitMatrix_{i,j} + neighborProfitMatrix_{i,j}) \tag{11}$$

其中 $sameClass_{i,j}$ 为一个布尔变量，当特征点 i, j 为同类别（同为端点或同为分叉点）时取 1，否则取 0。

$$geometryProfitMatrix_{i,j} = exp((x_i - x_j)^2 + (y_i - y_j)^2 + (angle_i - angle_j)^2) \tag{12}$$

$$neighborProfitMatrix_{i,j} = exp((x_p - x_q)^2 + (y_p - y_q)^2 + (angle_p - angle_q)^2) \quad (13)$$

式13中的 p, q 分别为距离特征点 i, j 的最近的特征点编号，即

$$p = argmin_p((x_i - x_p)^2 + (y_i - y_p)^2)$$

$$q = argmin_p((x_j - x_q)^2 + (y_j - y_q)^2)$$

$profitMatrix$ 综合考虑特征点本身的参数和最近邻居的参数，衡量出两个指纹任意一对特征点之间的匹配相似度。求解最佳匹配可以转化成一共二部图最大匹配模型。

定义 1 二部图：$G =< V, E >$, $V = V_1 \cup V_2$ 且 $V_1 \cap V_2 = \phi, G$ 中任一条边的两个端点一个属于 V_1, 另一个属于 V_2, 则称 G 为二部图, 记为 $G =< V_1, V_2, E >$。在二部图 $G =< V_1, V_2, E >$ 中，若 $|V_1| = m, |V_2| = n$, 且. $\forall u \in V_1, v \in V_2$ 均有 $[u, v] \in E$, 称 G 为完全二部图 , 记为 K_{mn}。

若两个指纹的特征点分别记为V_1,V_2,则由于他们两两间的匹配度均有 $profitMatrix$ 定义，所以 $profitMatrix$ 其实描述了一个完全二部图

定义 2 最大匹配：设二部图 $G =< V_1, V_2, E >, M \subseteq E$, 若 M 中任意两条边都不相邻, 称 M 为二部图 G 的一个匹配。若在 M 中再加入任何条边就都不是匹配了，称 M 为极大匹配。边数最多的极大匹配称为二部图 G 的最大匹配。

定义 3 完全匹配：设二部图 $G =< V_1, V_2, E >, M$ 是 G 中一个最大匹配若 $|M| = min\{|V_I|, V_2\}$, 称 M 为 G 中的一个完备匹配。若 $|V_1| = |V_2|, M$ 为 G 的完全匹配。

由上述定义可知" 最大匹配和完全匹配一定是极大匹配" 但反之不一定成立。因此" 只要求出所有极大匹配" 则所有最大匹配和完全匹配也就求出来了。

5.3 匈牙利算法求解二部图最大匹配

匈牙利算法，也被称作 $Kuhn - Munkres$ 算法，是一种用于进行二分图完全匹配的算法，而二部图很容易用邻接矩阵表示 [3]。其目的在于解决在一个完整的二部图中，进行最大权重的匹配. 在上文模型建立时我们已经得到了细节点匹配的损失矩阵，只需要使用 $Kuhn - Munkres$ 算法求解。

对于一个 n×n 的矩阵，匈牙利方法的步骤如下：

1. 从该行的所有其他条目中减去每行中的最小条目，这将使得改行的最小条目等于 0.

2. 从列中的所有其他条目中减去每列中的最小条目。这将使列中的最小条目现在等于 0.

3. 绘制具有 0 项的行和列的线，一遍绘制尽可能少的线.

4. 如果有 n 条被画出的直线，则可以实现零的最佳分配，算法完成，如果线的数目小于 n，则尚未达到最佳的零数，转到下一步.

5 查找没有任何行覆盖的最小条目，从没有划掉的每一行中减去该条目，将其添加到被划掉的每一列中，然后返回步骤 3.

5.4 模型求解

我们提取指纹核心区域的特征点时,统一选取保留距离指纹中心点最近的 $numFeatures$ 个特征点。在细节点去伪后，16 张指纹剩下的特征点个数从 23 到 67 不等，本节我们选取 $numFeatures = 20$

16 张图像之间两两之间的总匹配度见表3。任意两张指纹图像的匹配度为 0-1 之间

表 **3**　前 **6** 张图片两两之间的匹配度

	0	**1**	**2**	**3**	**4**	**5**	**6**
0	1	0.303675	0.275236	0.373354	0.280451	0.316537	0.284933
1	0.310738	1	0.122825	0.217637	0.383135	0.368417	0.220953
2	0.250855	0.112406	1	0.407693	0.163804	0.194754	0.40802
3	0.378363	0.211481	0.298624	1	0.268001	0.269794	0.375182
4	0.259674	0.316364	0.202442	0.303484	1	0.364401	0.248107
5	0.300937	0.334146	0.221745	0.281311	0.407993	1	0.322138
6	0.351364	0.204295	0.454979	0.40011	0.292844	0.329055	1

的一个数值，数值越接近 1，代表匹配度越高，为 1 则代表相同的指纹图像实现了完美匹配。

$similarityMat$ 矩阵的热力图表示见图13。

5.4.1 灵敏度分析

在使用匈牙利算法求解二部图最大匹配时，我们人物给定了每个指纹选取的特征点的个数 $numFeatures$。本节我们分析 $numFeatures$ 的取值是否对指纹的匹配度又重大影响。我们比较 $numFeatures = 10, 15, 20$ 时，相似度矩阵的平均误差绝对和最大绝对误差。

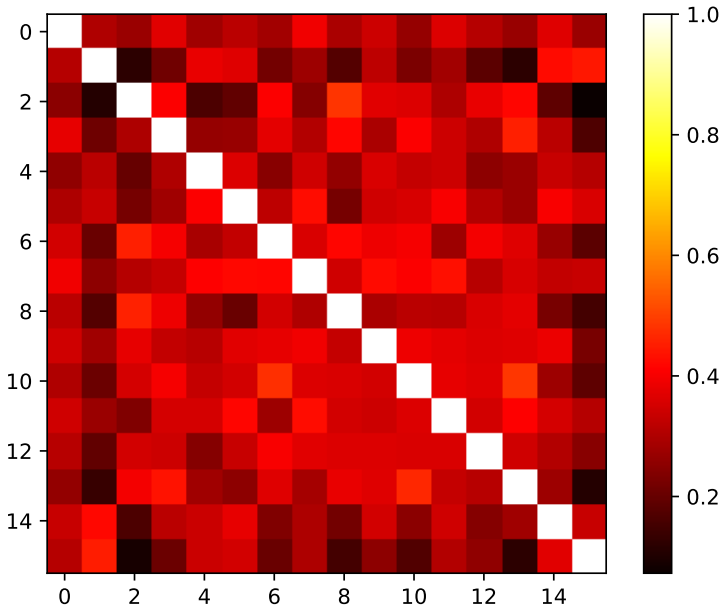
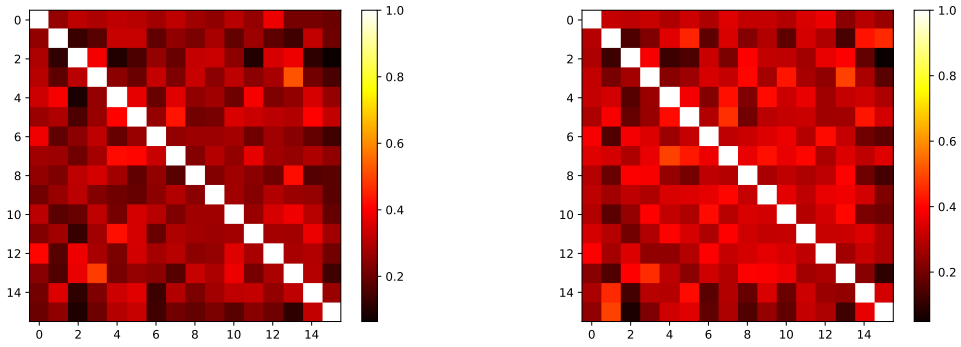


图 **13**　指纹匹配度矩阵



(a) 取 **10** 个特征点的相似度矩阵　　**(b)** 取 **15** 个特征点的相似度矩阵

图 **14**　$numFeatures = 10, 15$ 的对比

表 **4**　平均误差

	0	**1**	**2**
0	0	0.046907	0.06007
1	0.046907	0	0.031546
2	0.06007	0.031546	0

表 **5**　最大误差

	0	**1**	**2**
0	0	0.249391	0.238563
1	0.249391	0	0.108495
2	0.238563	0.108495	0

可以看出改变特征点的个数 $numFeatures$ 对指纹匹配度的影响不大，平均误差约为 6%，而最大误差也不超过 25%。

六、 问题三模型的建立与求解

6.1 问题分析

该问题要我们对 16 幅指纹图像进行分类，通过参考指纹学对指纹的分类的判断依据和本题设计的编码方式，我们可以从指纹学上建立一个基于指纹学奇异点特征和指纹梯度方向与骨架抽象的方式，对指纹的总体特征和方向进行有效的分类，同时利用第二问中编码的结果对该模型进行修正以及同一类别不同指纹之间的对比相似度结果进行修正，并给出最终结果。

6.2 奇异点特征与指纹方向场梯度分类模型

6.2.1 方向场梯度表示

方向场代表了指纹图像的本质特征，经过预处理的指纹图像已得到精确的方向场，本文算法基于梯度求方向场。具体步骤如下:

(1) 计算指纹图像每一点在 X,Y 方向的梯度 $\partial x(i,j)$ 和 $\partial y(i,j)$, 梯度计算使用 $Sobel$ 梯度算子。接着，使用下列公式计算每一点的方向，其中 W 常取常数 8.

$$V_x(i,j) = \sum_{u=i-w}^{i+w} \sum_{v=j-w}^{j+w} (2\partial_x(u,v)\partial_y(u,v)) \tag{14}$$

$$V_y(i,j) = \sum_{u=i-w}^{i+w} \sum_{v=j-w}^{j+w} \left(\partial_x^2(u,v) - \partial_y^2(u,v)\right) \tag{15}$$

$$\theta(x,y) = \frac{1}{2}\arctan\left(\frac{V_y(i,j)}{V_x(i,j)}\right) \tag{16}$$

(2) 由步骤 (1) 得到的 $q(x,y)$ 是局部脊线方向的最小平方估计，需要通过一个低滤波器进行修正，具体的修成方案如下: 首先将方向场转化到一个连续的向量场中，再对其滤波，滤波后的向量场可以用下式表示:

$$\phi_x(i,j) = \cos(2\theta(i,j)) \tag{17}$$

$$\phi_y(i,j) = \sin(2\theta(i,j)) \tag{18}$$

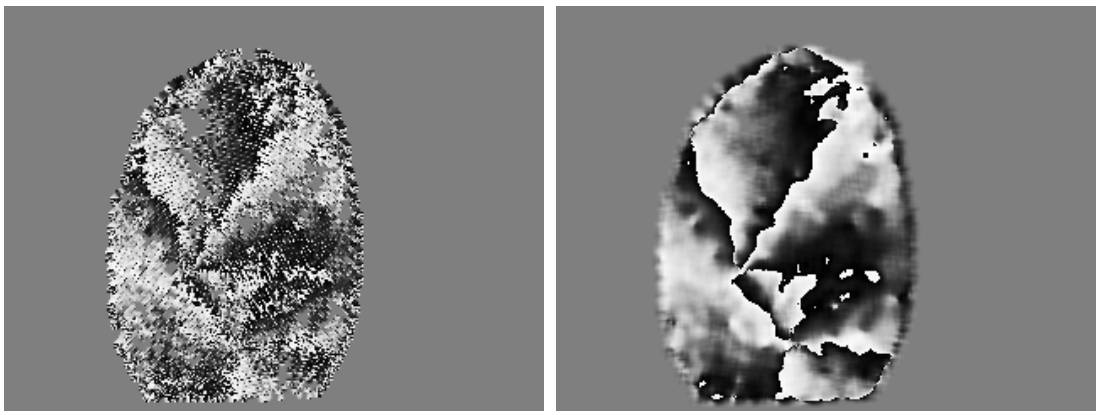
$$\phi'_x(x,y) = \sum_{u=-w_\phi/2}^{w_\phi/2} \sum_{v=-w_\phi/2}^{w_\phi/2} ((h(u,v)\phi_x(i-uw, j-vw)) \tag{19}$$

$$\phi'_y(x,y) = \sum_{u=-w_\phi/2}^{w_\phi/2} \sum_{v=-w_\phi/2}^{w_\phi/2} ((h(u,v)\phi_y(i-uw, j-vw)) \tag{20}$$

$$O(i,j) = \frac{1}{2}\operatorname{atan}2(\frac{\phi'_y(i,j)}{\phi'_x(i,j)} \tag{21}$$

$$O'(i,j) = O(i,j) + \frac{\pi}{2} \tag{22}$$

式中 $\phi'_x(x,y)$,$\phi'_y(x,y)$ 是向量场的 X,Y 分量，中间两式子中 $h(u,v)$ 是一个二维低通滤波器，积分为 1，滤波器的大小为 $W_\phi \times W_\phi$. 将滤波前后的方向场图像进行对比如图15所示:



(a) 取 **10** 个特征点的相似度矩阵　　**(b)** 取 **15** 个特征点的相似度矩阵

图 **15**　$numFeatures = 10, 15$ 的对比

6.2.2 指纹奇异点特征提取

奇异点提取是指纹图像处理中的一个重要部分，它可以作为指纹分类和识别过程中的定位基准点，根据其分布特性可以对指纹图像做粗分类，指纹的奇异点包括有指纹的中心点和三角点，根据指纹学的基础知识，中心点一般分为两种情况，有两个中心点或者一个中心点，而三角点并不是对于每个指纹都有的，不过也可以作为一个分类的依

据和重要特征，在本文中，根据指纹方向场梯度得出的结果并提炼出其骨架可以从而将整个图像的指纹场的方向通过灰度梯度图来体现，根据方向梯度图，将方向场归域化 $\Omega_{(0,4/\pi)}, \Omega_{(\pi/4,\pi/2)}, \Omega_{(\pi/3,3\pi/4)}, \Omega_{(3\pi/4,\pi)}$，在方向图的交界处会出现很多的曲线交叉，这意味着这些特殊点周围区域方向变化十分强烈，这与奇异点周围方向变化强烈这一理论相符合, 同时我们以 3×3 的 8 领域阵列，在特殊点交叉区域进行检验和去伪，从而得出了我们的奇异点。

$$|Poincare(i,j)+\pi| < 0.01\times\pi \tag{23}$$

在满足该条件下的且是方向场剧烈变化的交叉点则可确定为奇异点，包括中心点和三角点都可以识别。

根据指纹学相关知识和方向场的理论我们可以对图片的类型和方向进行分类，首先基于中心点我们分为两个中心点和一个中心点，同时在根据方向场的梯度和方向进行位置关系的判断，从而可以分辨出该指纹的类型，同时根据资料查找和题中所给的图片我们可以知道，一般带有三角点的图片可以认为是箕形，两个中心点的类型根据对应方向场的梯度方向大致可以判断出，螺旋型和斗型。

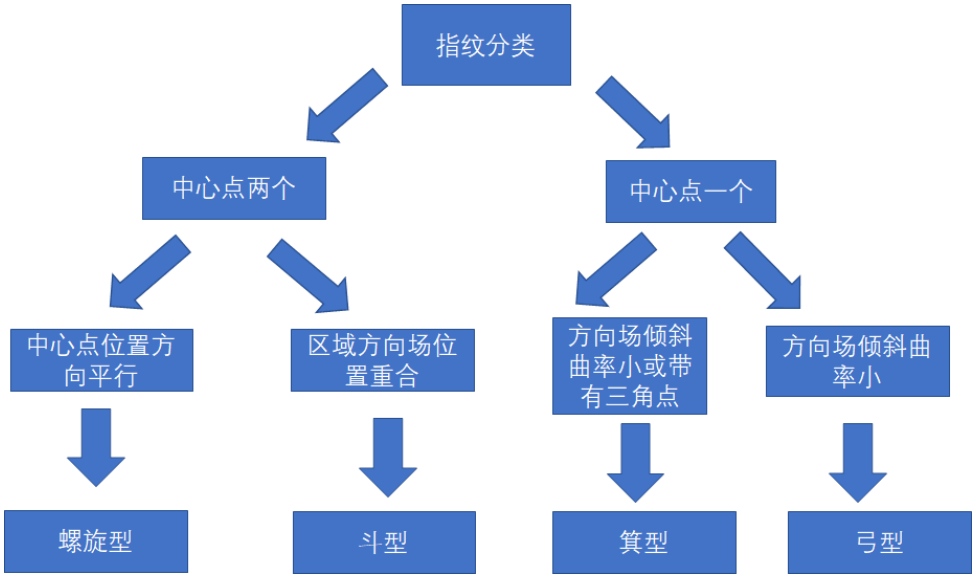


图 **16**　指纹分类流程

6.3 模型求解

6.3.1 sobel 算子与高斯滤波梯度提取

在求解梯度的过程当中，为了减少计算量，我们将图片的规模缩小为原来的二分之一，然后利用 python 和 opencv 求解出 sobel 算子对图像在 x 和 y 方向上的梯度，然后

我们利用低通滤波和高斯滤波的过滤对于图片中的方法进行了优化和提升。

$$Sobel_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} Sobel_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 1 & 0 \\ -1 & -2 & -1 \end{bmatrix} \tag{24}$$

记原图像的像素矩阵为 A，那么通过 sobel 算子可以很快求出图像在 x 和 y 方向上的梯度，G_x 和 G_y, 然后通过合成计算出当前像素的梯度 G'

$$G_x = Sobel_x * A \qquad G_y = Sobel_y * A \qquad G' = \sqrt{G_x^2 + G_y^2} \tag{25}$$

然后我们在上述方向场梯度模型的基础上加入高斯滤波的方式，从而最后得出了下面的结果 最后我们能通过梯度的计算将该点斜率的倾斜角换算出来，然后对每一行和每一

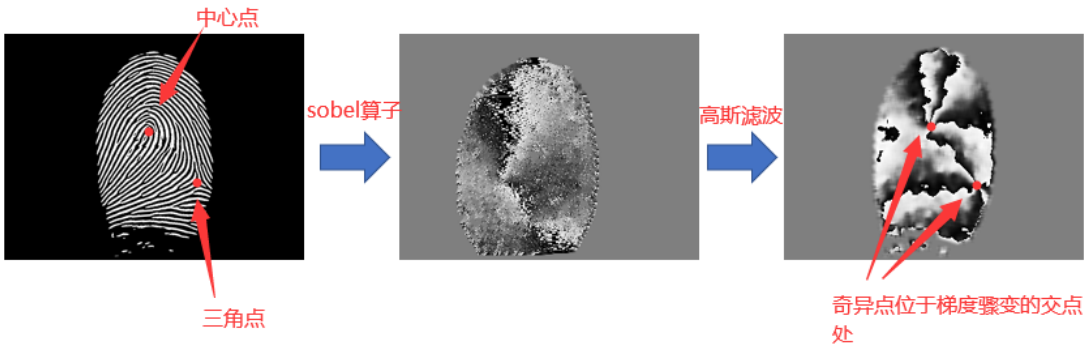


图 **17**　滤波梯度提取

列进行倾斜角均值的计算，从而勾勒出该点的方向指纹方向。

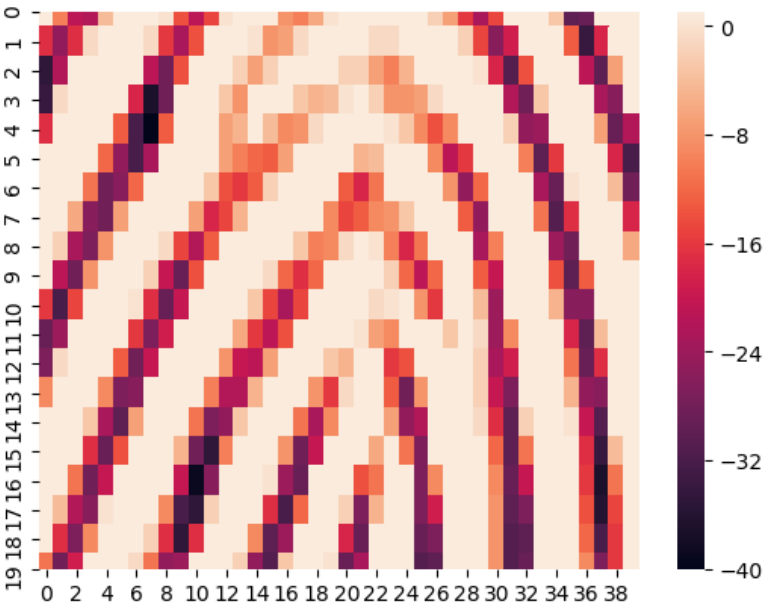


图 **18**　中心点附近图像梯度热力图

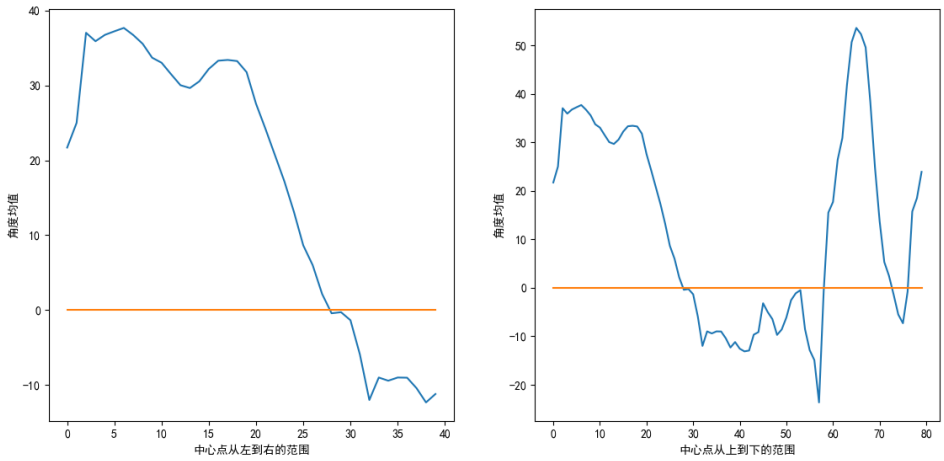


图 **19**　中心点附近图像勾勒

从图中我们就能看出来改中心点附件，从左往右他的角度确实呈现出整体向左的趋势，而从上往下也能体现出，当前位置下，呈现出先向上然后持平到最后下降的趋势，从而可以很好的反映中心点附件的朝向和趋势。

6.3.2 分水岭算法与 poincare 索引提取边界获取奇异点

骨架提取与分水岭算法也属于形态学处理范畴，骨架提取，也叫二值图像细化。这种算法能将一个连通区域细化成一个像素的宽度，用于特征提取和目标拓扑表示，在我们第一步的操作中，我们滤波后的梯度图像作为输入图像，然后设置阈值 $T = 200$ 将其转化成二值图像，然后在对其骨架进行提取。步骤：1. 读取滤波后的图像 2. 设置阈值，提取梯度骤变的边界处 3. 分水岭算法提取提取边界骨架

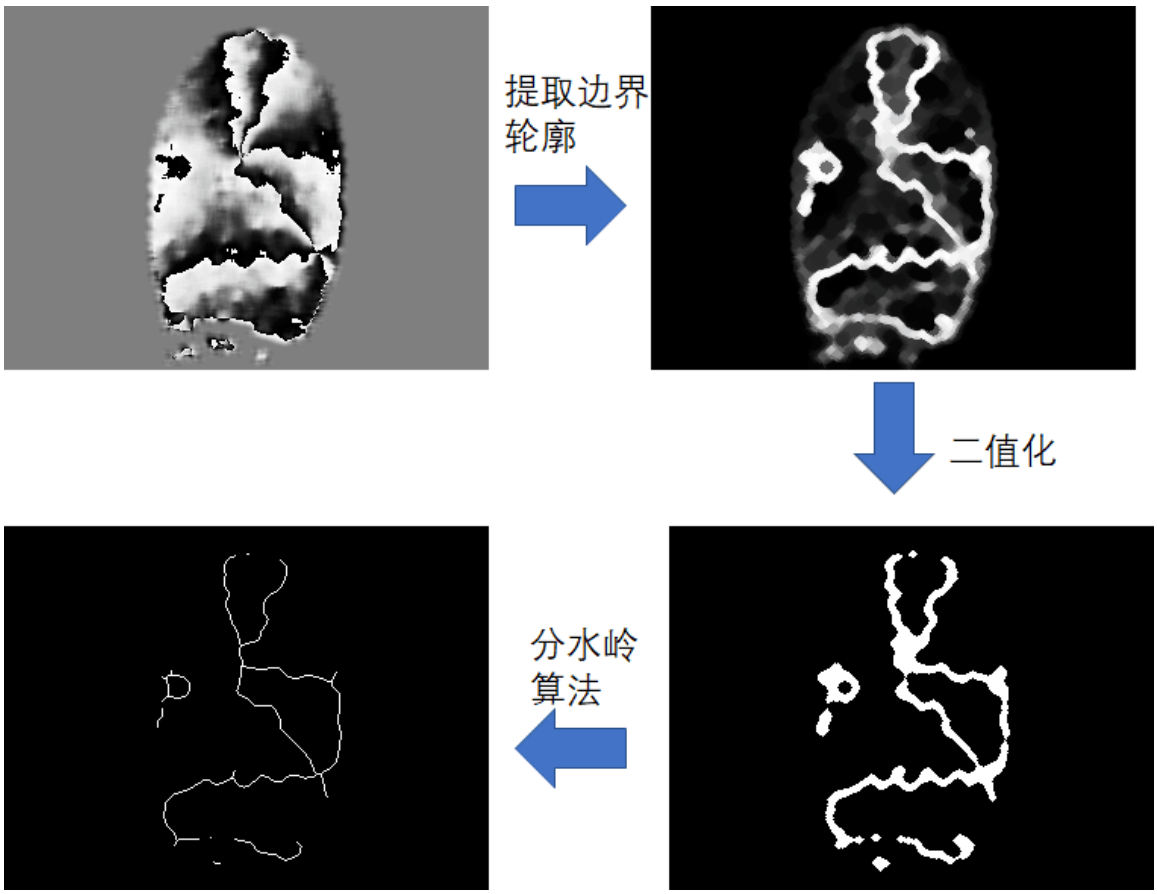


图 **20**　分水岭细化边界骨架

6.4 结果分析

6.4.1 结果展示

在获得骨架以后我们利用前期工作中提到的 poincareinedx 算法对梯度骤变的交界处的点进行奇异点的判断，然后并剔除了一些伪奇异点，从而准确得出了奇异点的所有坐标，为后面的分类提供支撑依据做准备。

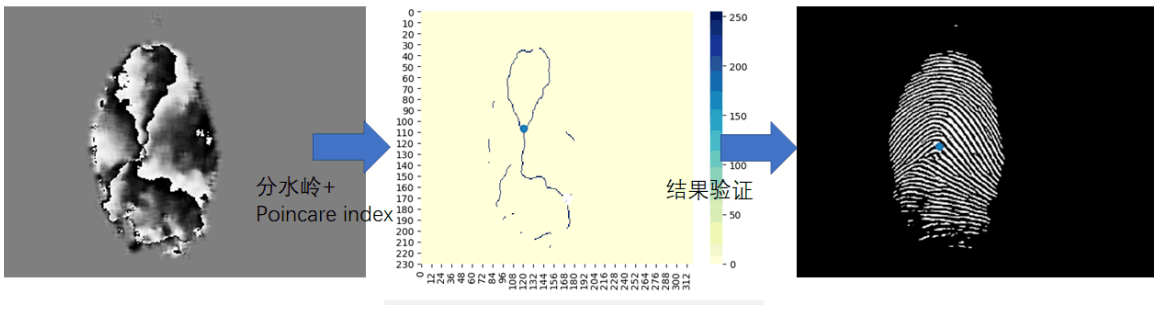


图 **21**　奇异点搜索效果

表 **6**　分类结果

$Name$	$coreCount$	$deltaCount$	$center_x$	$center_y$	$delta_x$	$delta_y$	$class$
01.tif	1	0	266	384			弓型
02.tif	1	1	308	192	416	322	箕型
03.tif	1	1	168	303	316	386	箕型
04.tif	1	0	250	248			弓型
05.tif	1	1	226	312	262	386	箕型
06.tif	1	1	249	256	306	340	箕型
07.tif	2	0	310,350	164,254			螺旋型
08.tif	2	1	366,304	300,268			螺旋型
09.tif	1	0	219	232			弓型
10.tif	1	1	282	240	362	384	箕型
12.tif	1	0	304	77			箕型
13.tif	1	0	336	278			箕型
14.tif	2	0	313,282	191,172			斗型
15.tif	1	0	305	328			弓型
16.tif	1	0	315	248			箕型

6.4.2 稳定性分析

为了保证我们模型的准确性，我们对图片添加了噪声进行干扰，观察其是否会对我们模型的奇异点识别和梯度造成很大的影响从而影响分类的效果。我们对图片添加了高斯噪声，可以看出噪声前后中心点的差距只有 5%, 而前后的趋势线也基本一致。

表 **7**　噪声前后结果对比

$Name$	$coreCount$	$deltaCount$	$center_x$	$center_y$	$delta_x$	$delta_y$	$class$
01.tif	1	0	266	384			弓型
01.tif 噪声后	1	0	262	395			弓型

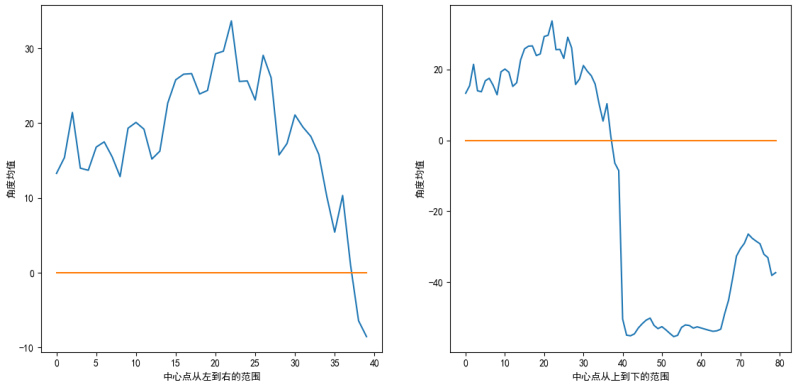


图 **22**　噪声前中心点趋势图

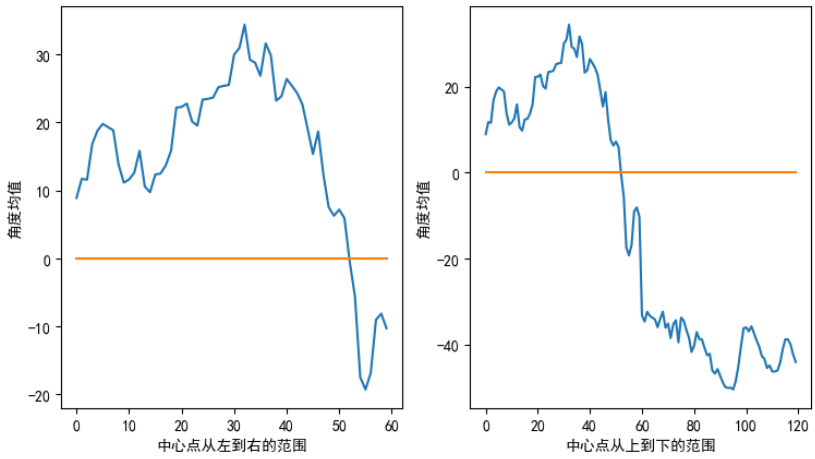


图 **23**　噪声后中心点趋势图

七、 模型评价

7.1 模型的优点

1. 模型的建立考虑了指纹识别的整体流程，对因采集导致的清晰度下降和指纹因旋转而导致不一致的问题进行了深入讨论，同时我们以指纹细节点的相对位置作为编码的信息，使得模型不会受到图像平移的影响，模型使用范围具有普遍性。

2. 模型二将题目转换成了一个二分图的图论模型，通过基于特征点的编码来进行分类操作，并利用匈牙利图论算法进行最大匹配，从而得出匹配的相似度匹配运算速度相比于其他智能算法的搜索要快。

3. 第三问从图片纹理的方向梯度和指纹学的基本知识利用编码模型进行了合理的计算从而得出了该模型的准确分类。

7.2 模型的缺点

1. 传统的 Poincare index 方法需要遍历全部像素点，使得模型算法变慢，降低了检测效率; 且由于 index 值的不稳定性，容易产生伪中心点。

2. 模型的二三问需要计算的梯度，和方向在操作层面上要计算的东西太多，需要继续整理成一个统一化的指标来进行预测会更好。

7.3 改进与展望

本文的模型是基于指纹学的相关知识和基本的编码思路来进行的操作，参考了许多关于指纹的模型来进行的模型建立，目前该模型对于指纹分类和指纹的相似度计算都给出了计算的方法和公式，要计算的指标和类型目前还有待进一步的提炼和统一，需要继续研究并给出统一的计算方法会更好。

1. 更加高效的求解奇异点和特征点的方法

在目前的主流方法下还有很多可以借鉴的方法能够更加快速的来求解奇异点和特征点，如基于变维数优化模型的指纹奇异点提取，基于方向角度变化的指纹奇异点提取的快速算法，这些算法的核心大多是利用方向角的旋转变化和维度的搜索来准确定位奇异点和特征点，通过这些算法我们可以更快速的获取到指纹图中的重要特征点。

2. 采用 SVM 等机器学习方法的指纹识别技术

如果在数据集达到一定规模程度之后，我们可以在当前特征点的基础上，利用支持向量机等机器学习的算法对指纹图像的全局特征和局部特征都能进行合适的分析和提取从而使得识别的效率更高，识别的准确度也相比少数量的指纹有更大的提升。

参考文献

[1] 尹婉琳, 叶桦, and 仰燕兰. 指纹图像的旋转校正与分类. *计算机科学与应用*, 4(5):9, 2014.

[2] 张祖泷, 杨永明, 韩凤玲, and 林坤明. 指纹奇异点精确定位新方法. *计算机工程与应用*, 48(32):203–207, 2012.

[3] 李昌华, 李智杰, and 高阳. 图谱和 kuhn-munkres 算法在图匹配中的应用研究. 2017.

[4] 王和平, 李媛, 张保成, and 康景利. 数学形态法在指纹特征提取中的研究与应用. *中北大学学报 (自然科学版)*, 25(5):330–332, 2004.

[5] 贺颖 and 蒲晓蓉. 应用 gabor 滤波的指纹识别算法的研究和实现. *计算机工程与应用*, (12):172–175, 2010.

[6] 赵应丁, 戴仕明, and 刘金刚. 基于灰度指纹图像的指纹特征提取算法研究. *系统仿真学报*, 018(002):319–322,326, 2006.

附录 A　16 个指纹的编码

```
0x05 0x00 0xf9 0xff 0xfe 0xff 0x03 0x00 0x11 0x00 0x00 0x00 0x06 0x00 0xef 0xff 0x03 0x00 0x00
    0x00 0xec 0xff 0x00 0x00 0xec 0xff 0xf7 0xff 0x02 0x00 0xe8 0xff 0x0f 0x00 0x03 0x00 0x05
    0x00 0xe0 0xff 0x00 0x00 0xda 0xff 0x0e 0x00 0x03 0x00 0x34 0x00 0xf5 0xff 0xfe 0xff 0x1e
    0x00 0x2e 0x00 0x03 0x00 0x01 0x00 0xc7 0xff 0x00 0x00 0x39 0x00 0xfe 0xff 0x00 0x00 0xc8
    0xff 0x1c 0x00 0x03 0x00 0x15 0x00 0x45 0x00 0xfe 0xff 0xb9 0xff 0x58 0x00 0xfe 0xff 0xa8
    0xff 0x50 0x00 0xff 0xff 0x90 0xff 0x31 0x00 0x03 0x00 0x94 0xff 0x4d 0x00 0xfe 0xff 0x75
    0xff 0xef 0xff 0x03 0x00 0x7a 0xff 0x29 0x00 0x00 0x00 0x66 0xff 0xc8 0xff 0x00 0x00 0x4e
    0xff 0x1d 0x00 0x03 0x00 0x43 0xff 0xdd 0xff 0x00 0x00 0x00 0x01 0x01 0x00 0x00 0x01 0x00
    0x01 0x00 0x01 0x00 0x00 0x01 0x00 0x01 0x01 0x00 0x01 0x00 0x00 0x00 0x00 0x00

0x04 0x00 0x00 0x00 0xfe 0xff 0xed 0xff 0x05 0x00 0x02 0x00 0x17 0x00 0xfa 0xff 0xff 0xff 0x16
    0x00 0x1c 0x00 0xff 0xff 0x1d 0x00 0xeb 0xff 0xff 0xff 0x25 0x00 0xf4 0xff 0x02 0x00 0x1e
    0x00 0xe7 0xff 0x00 0x00 0x1c 0x00 0x1e 0x00 0xff 0xff 0x19 0x00 0xdf 0xff 0x02 0x00 0xd6
    0xff 0xf6 0xff 0x02 0x00 0xf9 0xff 0x36 0x00 0xff 0xff 0xda 0xff 0xd4 0xff 0x00 0x00 0x32
    0x00 0xde 0xff 0xff 0xff 0x16 0x00 0x47 0x00 0xff 0xff 0x4f 0x00 0xe4 0xff 0x02 0x00 0x3a
    0x00 0x3d 0x00 0xff 0xff 0x46 0x00 0xd0 0xff 0x02 0x00 0x4e 0x00 0x22 0x00 0xff 0xff 0x36
    0x00 0x47 0x00 0xff 0xff 0xf6 0xff 0x5e 0x00 0x00 0x00 0x9d 0xff 0xd5 0xff 0x00 0x00 0x94
    0xff 0x00 0x00 0x00 0x00 0x90 0xff 0xeb 0xff 0x02 0x00 0x2c 0x00 0x6a 0x00 0x01 0x00 0x6f
    0x00 0xd4 0xff 0x00 0x00 0x79 0x00 0xf8 0xff 0x00 0x00 0x87 0xff 0x0a 0x00 0x00 0x00 0x94
    0xff 0x3a 0x00 0xfe 0xff 0x55 0x00 0x71 0x00 0x00 0x00 0x79 0x00 0xae 0xff 0xff 0xff 0x88
    0x00 0x42 0x00 0x00 0x00 0x97 0x00 0xf3 0xff 0x00 0x00 0x97 0x00 0x17 0x00 0x00 0x00 0x90
    0x00 0xc1 0xff 0x00 0x00 0x7b 0x00 0x6d 0x00 0x02 0x00 0x91 0x00 0x5c 0x00 0x03 0x00 0xb0
    0x00 0xf9 0xff 0x00 0x00 0xb0 0x00 0x31 0x00 0x03 0x00 0xae 0x00 0x42 0x00 0x00 0x00 0xbc
    0x00 0xdc 0xff 0x00 0x00 0xc6 0x00 0x08 0x00 0x00 0x00 0xd8 0x00 0xc6 0xff 0x00 0x00 0xd2
    0x00 0x59 0x00 0x00 0x00 0xe9 0x00 0xd4 0xff 0x03 0x00 0x00 0x01 0x01 0x01 0x01 0x01 0x00
    0x00 0x01 0x01 0x01 0x00 0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x00 0x00 0x00 0x00 0x00 0x00
    0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x01 0x01 0x00 0x01 0x00 0x00 0x00 0x00 0x00
    0x00

0xf1 0xff 0x12 0x00 0x00 0x00 0x14 0x00 0xf2 0xff 0x00 0x00 0x09 0x00 0xe9 0xff 0x00 0x00 0x11
    0x00 0x12 0x00 0x01 0x00 0x1d 0x00 0xe7 0xff 0x01 0x00 0x0e 0x00 0xd2 0xff 0x00 0x00 0x1b
    0x00 0x29 0x00 0xff 0xff 0x30 0x00 0xf0 0xff 0x02 0x00 0xd0 0xff 0x19 0x00 0x00 0x00 0x30
    0x00 0xe5 0xff 0x00 0x00 0xe5 0xff 0xcd 0xff 0x00 0x00 0xfa 0xff 0x3a 0x00 0xfe 0xff 0xf8
    0xff 0x42 0x00 0xfe 0xff 0x08 0x00 0x4a 0x00 0x00 0x00 0xfb 0xff 0x4d 0x00 0xfe 0xff 0x45
    0x00 0x24 0x00 0x01 0x00 0x4d 0x00 0xe3 0xff 0x02 0x00 0x57 0x00 0x06 0x00 0x02 0x00 0x07
    0x00 0x5b 0x00 0x00 0x00 0x4f 0x00 0x33 0x00 0xff 0xff 0x31 0x00 0x5b 0x00 0x00 0x00 0xc1
    0xff 0x55 0x00 0xfe 0xff 0x33 0x00 0x69 0x00 0x00 0x00 0x33 0x00 0x72 0x00 0x00 0x00 0x89
    0xff 0x39 0x00 0x00 0x00 0x7e 0xff 0x1d 0x00 0xfe 0xff 0x32 0x00 0x7d 0x00 0x00 0x00 0x78
    0xff 0x3f 0x00 0x00 0x00 0x76 0xff 0x3b 0x00 0xfe 0xff 0xdd 0xff 0x97 0x00 0x00 0x00 0x24
    0x00 0x99 0x00 0xfe 0xff 0x64 0xff 0x1a 0x00 0x00 0x00 0x65 0xff 0x20 0x00 0xfe 0xff 0x7a
    0x00 0x68 0x00 0x02 0x00 0x60 0x00 0x89 0x00 0xfe 0xff 0xb2 0xff 0x99 0x00 0x00 0x00 0x5d
    0x00 0x95 0x00 0xfe 0xff 0x20 0x00 0xaf 0x00 0xfe 0xff 0x9a 0xff 0x95 0x00 0x00 0x00 0x5f
    0x00 0x9a 0x00 0x00 0x00 0x40 0x00 0xab 0x00 0xfe 0xff 0x52 0x00 0xa8 0x00 0x03 0x00 0x75
    0x00 0x93 0x00 0xfe 0xff 0xf9 0xff 0xbf 0x00 0xfe 0xff 0x83 0x00 0x8e 0x00 0x03 0x00 0x3b
```

```
    0xff 0x16 0x00 0x03 0x00 0x34 0xff 0x26 0x00 0x00 0x00 0x49 0xff 0x65 0x00 0xfe 0xff 0x2e
    0xff 0x0f 0x00 0x03 0x00 0x87 0xff 0xaf 0x00 0xfe 0xff 0x89 0xff 0xb4 0x00 0x00 0x00 0x36
    0xff 0x58 0x00 0xfe 0xff 0x4b 0xff 0x7f 0x00 0x00 0x00 0x42 0xff 0x72 0x00 0x00 0x00 0x24
    0xff 0x1d 0x00 0x03 0x00 0x6f 0xff 0xae 0x00 0x00 0x00 0x36 0xff 0x69 0x00 0xfe 0xff 0x1e
    0xff 0x2b 0x00 0x00 0x00 0x39 0xff 0x8d 0x00 0xfe 0xff 0x1d 0xff 0x75 0x00 0xfe 0xff 0x19
    0xff 0x7a 0x00 0xfe 0xff 0x1e 0xff 0x85 0x00 0x00 0x00 0x22 0xff 0x97 0x00 0x00 0x00 0x15
    0xff 0x82 0x00 0xfe 0xff 0x1b 0xff 0x8f 0x00 0x00 0x00 0x13 0xff 0x8d 0x00 0x03 0x00 0x0d
    0xff 0x8d 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
    0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
    0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
    0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00

0xfd 0xff 0x02 0x00 0xfe 0xff 0xf8 0xff 0xf4 0xff 0x02 0x00 0x12 0x00 0xf6 0xff 0x00 0x00 0xeb
    0xff 0x08 0x00 0xfe 0xff 0x19 0x00 0xf8 0xff 0xfe 0xff 0xfc 0xff 0xde 0xff 0x02 0x00 0x1c
    0x00 0xdc 0xff 0x00 0x00 0xc5 0xff 0x10 0x00 0x03 0x00 0x40 0x00 0x04 0x00 0x03 0x00 0x1f
    0x00 0xc4 0xff 0x00 0x00 0x39 0x00 0xd4 0xff 0xfe 0xff 0xb1 0xff 0xf1 0xff 0x02 0x00 0x51
    0x00 0xfb 0xff 0xfe 0xff 0x37 0x00 0x40 0x00 0x03 0x00 0xa9 0xff 0xe0 0xff 0x02 0x00 0xe1
    0xff 0x59 0x00 0x00 0x00 0xf3 0xff 0x63 0x00 0xff 0xff 0xa3 0xff 0x27 0x00 0xfe 0xff 0x32
    0x00 0x5f 0x00 0xfe 0xff 0x6c 0x00 0xe6 0xff 0x00 0x00 0x69 0x00 0xd6 0xff 0x03 0x00 0x73
    0x00 0xe6 0xff 0x00 0x00 0x71 0x00 0xda 0xff 0x03 0x00 0x84 0xff 0xec 0xff 0x02 0x00 0x89
    0xff 0x29 0x00 0x00 0x00 0x89 0x00 0xe1 0xff 0x00 0x00 0x92 0x00 0x00 0x00 0x03 0x00 0x63
    0xff 0x1b 0x00 0xfe 0xff 0x9e 0x00 0x26 0x00 0x00 0x00 0x9a 0x00 0x3b 0x00 0x00 0x00 0x01
    0x00 0x00 0x00 0x00 0x01 0x00 0x00 0x01 0x00 0x00 0x00 0x01 0x01 0x00 0x00 0x00 0x01 0x00
    0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00

0xff 0xff 0xfc 0xff 0xfe 0xff 0x07 0x00 0x00 0x00 0x01 0x00 0xf2 0xff 0x10 0x00 0xfe 0xff 0x10
    0x00 0x13 0x00 0xff 0xff 0xfa 0xff 0xd7 0xff 0x02 0x00 0xcf 0xff 0xf5 0xff 0x03 0x00 0x2a
    0x00 0x21 0x00 0x01 0x00 0xc5 0xff 0x0e 0x00 0x02 0x00 0x3d 0x00 0xf6 0xff 0x00 0x00 0x2a
    0x00 0x2e 0x00 0x01 0x00 0x38 0x00 0x27 0x00 0xff 0xff 0xb7 0xff 0x09 0x00 0x03 0x00 0x2a
    0x00 0x42 0x00 0x01 0x00 0xd2 0xff 0xc0 0xff 0x00 0x00 0x46 0x00 0x2c 0x00 0x01 0x00 0xc3
    0xff 0xbf 0xff 0x00 0x00 0xb4 0xff 0x3a 0x00 0x00 0x00 0x43 0x00 0xb7 0xff 0x00 0x00 0xbc
    0xff 0x57 0x00 0x03 0x00 0x2c 0x00 0x6c 0x00 0x00 0x00 0x62 0x00 0x40 0x00 0xfe 0xff 0x89
    0xff 0x0b 0x00 0x03 0x00 0x91 0xff 0x35 0x00 0x00 0x00 0x2a 0x00 0x75 0x00 0x00 0x00 0x73
    0x00 0x36 0x00 0x00 0x00 0x61 0x00 0x54 0x00 0x03 0x00 0x80 0x00 0xd7 0xff 0x00 0x00 0x4e
    0x00 0x71 0x00 0x02 0x00 0x5c 0x00 0x77 0x00 0x02 0x00 0x41 0x00 0x8e 0x00 0xfe 0xff 0xf8
    0xff 0x9d 0x00 0xfe 0xff 0x7a 0x00 0x6b 0x00 0xfe 0xff 0x5f 0x00 0x8e 0x00 0x03 0x00 0x75
    0xff 0x7f 0x00 0xfe 0xff 0x78 0x00 0x95 0x00 0x02 0x00 0x3f 0xff 0xe2 0xff 0x00 0x00 0x4b
    0xff 0x5b 0x00 0x02 0x00 0x34 0xff 0xe7 0xff 0xfe 0xff 0x65 0xff 0x93 0x00 0x02 0x00 0x24
    0xff 0x04 0x00 0x02 0x00 0x17 0xff 0x19 0x00 0x03 0x00 0x1e 0xff 0x52 0x00 0xfe 0xff 0x14
    0xff 0x31 0x00 0xfe 0xff 0x1e 0xff 0x5b 0x00 0x03 0x00 0x12 0xff 0x3a 0x00 0x03 0x00 0x01
    0x00 0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x00 0x01 0x01 0x00 0x00 0x00 0x01 0x00 0x01 0x01
    0x00 0x01 0x00 0x00 0x00 0x00 0x00 0x00 0x01 0x01 0x00 0x01 0x01 0x01 0x01 0x00 0x00 0x01
    0x01 0x01 0x01 0x00 0x01 0x01 0x00 0x00

0x06 0x00 0xfd 0xff 0x01 0x00 0x0d 0x00 0x10 0x00 0xff 0xff 0xee 0xff 0x0b 0x00 0xfe 0xff 0x10
    0x00 0x25 0x00 0xff 0xff 0xcd 0xff 0xf0 0xff 0x02 0x00 0x3d 0x00 0xf5 0xff 0x00 0x00 0xb3
    0xff 0x04 0x00 0x03 0x00 0xd4 0xff 0xbd 0xff 0x00 0x00 0x42 0x00 0x3e 0x00 0x01 0x00 0xc5
```

```
    0xff 0xbb 0xff 0x00 0x00 0x0f 0x00 0x5c 0x00 0xfe 0xff 0x5b 0x00 0x1c 0x00 0x02 0x00 0xe9
    0xff 0x9f 0xff 0xff 0xff 0x46 0x00 0xb7 0xff 0xff 0xff 0x61 0x00 0x31 0x00 0x02 0x00 0x57
    0x00 0x49 0x00 0x03 0x00 0x10 0x00 0x73 0x00 0xfe 0xff 0xd5 0xff 0x6d 0x00 0x00 0x00 0x70
    0x00 0x37 0x00 0x00 0x00 0x7e 0x00 0x06 0x00 0x00 0x00 0x88 0xff 0x2e 0x00 0x00 0x00 0x43
    0x00 0x6e 0x00 0xfe 0xff 0x62 0x00 0x57 0x00 0x02 0x00 0xb2 0xff 0x6d 0x00 0xfe 0xff 0x52
    0x00 0x73 0x00 0xfe 0xff 0xa5 0xff 0x70 0x00 0xfe 0xff 0x8b 0x00 0xd7 0xff 0x00 0x00 0xb0
    0xff 0x81 0x00 0x00 0x00 0xf0 0xff 0x97 0x00 0x00 0x00 0x98 0x00 0xfd 0xff 0x00 0x00 0x98
    0x00 0xe7 0xff 0x00 0x00 0x8e 0x00 0xbd 0xff 0x00 0x00 0xef 0xff 0x9d 0x00 0xfe 0xff 0x0d
    0x00 0x9e 0x00 0xfe 0xff 0x9b 0x00 0x24 0x00 0x02 0x00 0x95 0x00 0xc8 0xff 0xfe 0xff 0x01
    0x00 0xa0 0x00 0xff 0xff 0xd5 0xff 0x9c 0x00 0xfe 0xff 0x58 0x00 0x8e 0x00 0x02 0x00 0x9d
    0xff 0x88 0x00 0xfe 0xff 0x9d 0xff 0x8d 0x00 0x03 0x00 0x53 0xff 0x31 0x00 0x00 0x00 0xb2
    0x00 0x21 0x00 0x00 0x00 0x51 0xff 0x50 0x00 0x03 0x00 0x50 0xff 0x55 0x00 0x00 0x00 0x3a
    0xff 0xd1 0xff 0x02 0x00 0x3f 0xff 0x4e 0x00 0x02 0x00 0x53 0xff 0x77 0x00 0xfe 0xff 0x2b
    0xff 0x06 0x00 0xfe 0xff 0x3b 0xff 0x5d 0x00 0xfe 0xff 0x3c 0xff 0x61 0x00 0x00 0x00 0x41
    0xff 0x6d 0x00 0xfe 0xff 0x22 0xff 0x22 0x00 0xfe 0xff 0x20 0xff 0x28 0x00 0x03 0x00 0x24
    0xff 0x3b 0x00 0x03 0x00 0x00 0x01 0x01 0x00 0x01 0x01 0x01 0x00 0x00 0x01 0x01 0x01 0x00
    0x01 0x01 0x00 0x00 0x01 0x00 0x00 0x00 0x00 0x01 0x01 0x00 0x01 0x00 0x00 0x00 0x00 0x00
    0x00 0x00 0x00 0x01 0x00 0x00 0x00 0x01 0x01 0x00 0x00 0x00 0x00 0x00 0x00 0x01 0x00 0x01
    0x00 0x00 0x00 0x01 0x00 0x00

0x00 0x00 0xff 0xff 0x01 0x00 0xf0 0xff 0x06 0x00 0xff 0xff 0xe9 0xff 0x07 0x00 0x01 0x00 0xf4
    0xff 0x1f 0x00 0xff 0xff 0x16 0x00 0xe5 0xff 0x01 0x00 0x00 0x00 0x28 0x00 0xff 0xff 0x13
    0x00 0x25 0x00 0xff 0xff 0x2e 0x00 0x18 0x00 0x00 0x00 0x2c 0x00 0x29 0x00 0x03 0x00 0xc4
    0xff 0xee 0xff 0xff 0xff 0x4b 0x00 0x03 0x00 0xfe 0xff 0xbe 0xff 0xd7 0xff 0x01 0x00 0xf7
    0xff 0xb0 0xff 0x01 0x00 0xb8 0xff 0xd9 0xff 0x00 0x00 0xaf 0xff 0xdc 0xff 0x00 0x00 0x1a
    0x00 0x5f 0x00 0x01 0x00 0x0d 0x00 0x6a 0x00 0x01 0x00 0xc6 0xff 0x5a 0x00 0x01 0x00 0x72
    0x00 0x0c 0x00 0xfe 0xff 0x66 0x00 0xca 0xff 0xff 0xff 0x41 0x00 0xa0 0xff 0x01 0x00 0x74
    0x00 0x10 0x00 0x00 0x00 0xc3 0xff 0x9b 0xff 0x01 0x00 0x60 0x00 0x45 0x00 0xfe 0xff 0x77
    0x00 0xfb 0xff 0x03 0x00 0x81 0x00 0x0d 0x00 0x02 0x00 0x88 0x00 0xeb 0xff 0x02 0x00 0x82
    0x00 0xca 0xff 0xfe 0xff 0x6f 0xff 0x29 0x00 0x03 0x00 0x6d 0xff 0xcb 0xff 0x02 0x00 0x92
    0x00 0xc7 0xff 0x03 0x00 0x61 0xff 0x24 0x00 0x03 0x00 0x93 0x00 0xb6 0xff 0xfe 0xff 0x9b
    0x00 0xbb 0xff 0x00 0x00 0xa1 0x00 0x37 0x00 0x02 0x00 0x98 0x00 0xb0 0xff 0xfe 0xff 0xb1
    0x00 0xfe 0xff 0x00 0x00 0xb6 0x00 0x25 0x00 0x02 0x00 0xb1 0x00 0x3b 0x00 0x00 0x00 0xba
    0x00 0xe6 0xff 0x03 0x00 0xc1 0x00 0xe5 0xff 0x03 0x00 0x00 0x01 0x00 0x00 0x00 0x00 0x00
    0x00 0x01 0x01 0x01 0x00 0x00 0x00 0x01 0x00 0x00 0x00 0x00 0x01 0x01 0x00 0x00 0x00 0x01
    0x00 0x01 0x00 0x01 0x01 0x00 0x01 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00

0x05 0x00 0xfe 0xff 0x00 0x00 0x0c 0x00 0xef 0xff 0x02 0x00 0xed 0xff 0xf2 0xff 0x02 0x00 0xee
    0xff 0x12 0x00 0x03 0x00 0x02 0x00 0x20 0x00 0xff 0xff 0x0b 0x00 0x25 0x00 0xff 0xff 0x27
    0x00 0x12 0x00 0xfe 0xff 0x22 0x00 0x1e 0x00 0x00 0x00 0x15 0x00 0x2a 0x00 0x01 0x00 0x1d
    0x00 0x25 0x00 0x00 0x00 0xdd 0xff 0xd7 0xff 0x02 0x00 0xfa 0xff 0x42 0x00 0xff 0xff 0x41
    0x00 0x20 0x00 0x00 0x00 0xbb 0xff 0x17 0x00 0x03 0x00 0x20 0x00 0x4b 0x00 0x01 0x00 0xc5
    0xff 0xc7 0xff 0x00 0x00 0x0d 0x00 0xaf 0xff 0x01 0x00 0x5c 0x00 0x0b 0x00 0xfe 0xff 0x5b
    0x00 0x15 0x00 0x00 0x00 0xa2 0xff 0xe3 0xff 0x00 0x00 0xfa 0xff 0x9d 0xff 0xff 0xff 0xc0
    0xff 0x4e 0x00 0x00 0x00 0xb7 0xff 0x4a 0x00 0xfe 0xff 0x99 0xff 0xe3 0xff 0xfe 0xff 0x90
    0xff 0xf7 0xff 0x00 0x00 0x6f 0x00 0xd7 0xff 0xfe 0xff 0xc3 0xff 0x8f 0xff 0x00 0x00 0x6f
    0x00 0xba 0xff 0xfe 0xff 0x5b 0x00 0x9d 0xff 0xff 0xff 0x59 0x00 0x97 0xff 0x00 0x00 0x82
```

```
    0x00 0xce 0xff 0x00 0x00 0x82 0xff 0xbe 0xff 0x00 0x00 0x7b 0xff 0xca 0xff 0x02 0x00 0x94
    0x00 0xd1 0xff 0x03 0x00 0x8a 0x00 0xaa 0xff 0xfe 0xff 0x8a 0x00 0xa4 0xff 0xfe 0xff 0x00
    0x00 0x01 0x01 0x01 0x00 0x01 0x00 0x01 0x01 0x00 0x01 0x00 0x01 0x00 0x01 0x00 0x00 0x00
    0x00 0x01 0x00 0x00 0x01 0x00 0x00 0x01 0x00 0x01 0x00 0x00 0x00 0x01 0x01 0x00 0x01

0x0a 0x00 0x13 0x00 0xfe 0xff 0x1b 0x00 0xfb 0xff 0x02 0x00 0xe9 0xff 0x15 0x00 0xfe 0xff 0x1f
    0x00 0xe6 0xff 0x02 0x00 0x34 0x00 0xfd 0xff 0x00 0x00 0x3c 0x00 0xfa 0xff 0xfe 0xff 0xbe
    0xff 0x26 0x00 0x03 0x00 0x41 0x00 0xcc 0xff 0x00 0x00 0xcf 0xff 0x4b 0x00 0x00 0x00 0xfb
    0xff 0x5a 0x00 0x00 0x00 0x61 0x00 0xfe 0xff 0x03 0x00 0x5c 0x00 0xde 0xff 0xfe 0xff 0x5a
    0x00 0x29 0x00 0xfe 0xff 0x9a 0xff 0xf2 0xff 0x03 0x00 0xa0 0xff 0x29 0x00 0x00 0x00 0xf7
    0xff 0x71 0x00 0xfe 0xff 0x71 0x00 0xd4 0xff 0x00 0x00 0x16 0x00 0x7a 0x00 0xfe 0xff 0x7e
    0x00 0x01 0x00 0x03 0x00 0x63 0x00 0x52 0x00 0xfe 0xff 0x50 0x00 0x6a 0x00 0x00 0x00 0x58
    0x00 0x64 0x00 0x00 0x00 0x7b 0x00 0x3d 0x00 0xfe 0xff 0x75 0xff 0x21 0x00 0x03 0x00 0x87
    0x00 0x3d 0x00 0x03 0x00 0x97 0x00 0xf7 0xff 0x00 0x00 0x67 0x00 0x78 0x00 0x03 0x00 0x9c
    0x00 0xe1 0xff 0x03 0x00 0x83 0x00 0x5e 0x00 0x03 0x00 0x6a 0x00 0x7c 0x00 0x00 0x00 0x85
    0x00 0x62 0x00 0x00 0x00 0xa6 0x00 0xef 0xff 0x00 0x00 0xa7 0x00 0x59 0x00 0x00 0x00 0xb9
    0x00 0x3e 0x00 0x03 0x00 0xb9 0x00 0x46 0x00 0x00 0x00 0xc5 0x00 0x35 0x00 0x00 0x00 0x01
    0x00 0x00 0x01 0x00 0x00 0x01 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
    0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00

0xf7 0xff 0x03 0x00 0x01 0x00 0xe8 0xff 0x0e 0x00 0xff 0xff 0x1c 0x00 0x03 0x00 0x00 0x00 0xdf
    0xff 0xef 0xff 0x02 0x00 0x27 0x00 0x00 0x00 0x02 0x00 0x0b 0x00 0x2b 0x00 0x01 0x00 0x2a
    0x00 0x21 0x00 0x01 0x00 0x3c 0x00 0xfd 0xff 0x00 0x00 0x27 0x00 0x35 0x00 0xff 0xff 0x22
    0x00 0x41 0x00 0xfe 0xff 0xba 0xff 0x16 0x00 0x03 0x00 0x07 0x00 0xb4 0xff 0x01 0x00 0x50
    0x00 0x1c 0x00 0x02 0x00 0x4f 0x00 0xde 0xff 0x00 0x00 0xa8 0xff 0xe0 0xff 0x03 0x00 0xae
    0xff 0x33 0x00 0xfe 0xff 0x60 0x00 0x17 0x00 0x00 0x00 0xbb 0xff 0xb5 0xff 0x00 0x00 0x7e
    0x00 0xdb 0xff 0x00 0x00 0x77 0x00 0x39 0x00 0x02 0x00 0x75 0xff 0x0a 0x00 0x00 0x00 0x2d
    0x00 0x78 0xff 0x02 0x00 0x8e 0x00 0x28 0x00 0x02 0x00 0x8f 0x00 0xc5 0xff 0x00 0x00 0x9e
    0x00 0xe4 0xff 0x00 0x00 0xaa 0x00 0x27 0x00 0x00 0x00 0xb5 0x00 0xc5 0xff 0x00 0x00 0xbf
    0x00 0xff 0xff 0x00 0x00 0xc7 0x00 0xfa 0xff 0x03 0x00 0xbf 0x00 0xae 0xff 0x00 0x00 0xd7
    0x00 0xcd 0xff 0x00 0x00 0x00 0x01 0x00 0x01 0x01 0x00 0x00 0x00 0x01 0x01 0x01 0x01 0x00
    0x00 0x00 0x01 0x00 0x01 0x00 0x00 0x00 0x00 0x01 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00

0xfb 0xff 0x0b 0x00 0xff 0xff 0xf5 0xff 0xf7 0xff 0x02 0x00 0xe8 0xff 0xef 0xff 0x00 0x00 0xdd
    0xff 0x02 0x00 0x03 0x00 0xd6 0xff 0x00 0x00 0x00 0x00 0xec 0xff 0x2f 0x00 0x01 0x00 0xc4
    0xff 0xfb 0xff 0x00 0x00 0x3b 0x00 0x15 0x00 0xff 0xff 0x15 0x00 0x3d 0x00 0x01 0x00 0xcb
    0xff 0xd9 0xff 0x00 0x00 0x1a 0x00 0xc3 0xff 0x00 0x00 0x30 0x00 0x33 0x00 0xff 0xff 0xb8
    0xff 0xe1 0xff 0x02 0x00 0x51 0x00 0x13 0x00 0x02 0x00 0xe3 0xff 0x55 0x00 0x00 0x00 0x67
    0x00 0x16 0x00 0x00 0x00 0x4a 0x00 0x4f 0x00 0x01 0x00 0xc7 0xff 0x67 0x00 0x00 0x00 0x87
    0xff 0x06 0x00 0xfe 0xff 0x49 0x00 0x64 0x00 0x01 0x00 0x88 0xff 0xda 0xff 0x00 0x00 0x7d
    0x00 0xee 0xff 0x00 0x00 0x7c 0xff 0xed 0xff 0x02 0x00 0x82 0xff 0x3e 0x00 0xfe 0xff 0x87
    0x00 0x2a 0x00 0x00 0x00 0x70 0xff 0xed 0xff 0x00 0x00 0x8d 0x00 0x28 0x00 0x02 0x00 0x68
    0xff 0xff 0xff 0x02 0x00 0x99 0x00 0xeb 0xff 0x00 0x00 0x9b 0x00 0xdc 0xff 0x02 0x00 0xa3
    0x00 0xe0 0xff 0x02 0x00 0x90 0x00 0x53 0x00 0x00 0x00 0xa7 0x00 0x36 0x00 0x00 0x00 0xb6
    0x00 0x0e 0x00 0x00 0x00 0x01 0x01 0x00 0x00 0x00 0x00 0x00 0x01 0x00 0x00 0x00 0x01 0x00
    0x00 0x00 0x00 0x00 0x00 0x01 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
    0x00 0x00 0x00
```

```
0xf8 0xff 0x05 0x00 0x01 0x00 0xf6 0xff 0x01 0x00 0x02 0x00 0xfc 0xff 0xf0 0xff 0x01 0x00 0xf3
    0xff 0xf2 0xff 0x01 0x00 0x1f 0x00 0x0b 0x00 0xff 0xff 0xfe 0xff 0x2e 0x00 0xff 0xff 0xd3
    0xff 0xf4 0xff 0x00 0x00 0x2b 0x00 0xd1 0xff 0xff 0xff 0x3b 0x00 0xe2 0xff 0xff 0xff 0xbb
    0xff 0x3b 0x00 0xfe 0xff 0x55 0x00 0x35 0x00 0x01 0x00 0x60 0x00 0x41 0x00 0xff 0xff 0x44
    0x00 0xa2 0xff 0x01 0x00 0x74 0x00 0x04 0x00 0x02 0x00 0x63 0x00 0x47 0x00 0xff 0xff 0x41
    0x00 0x96 0xff 0x00 0x00 0x80 0x00 0xea 0xff 0x01 0x00 0x74 0x00 0x43 0x00 0x01 0x00 0x8b
    0x00 0xf6 0xff 0x00 0x00 0x94 0x00 0x28 0x00 0xff 0xff 0x9f 0x00 0xe9 0xff 0x00 0x00 0x97
    0x00 0xb5 0xff 0xff 0xff 0xc8 0x00 0xf0 0xff 0x00 0x00 0x00 0x00 0x01 0x01 0x01 0x01 0x00
    0x01 0x01 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x01 0x00 0x00 0x00 0x00 0x00 0x00

0x04 0x00 0x00 0x00 0x01 0x00 0xfe 0xff 0xef 0xff 0x01 0x00 0xe6 0xff 0x15 0x00 0xfe 0xff 0x26
    0x00 0x01 0x00 0x01 0x00 0xdb 0xff 0x0c 0x00 0xfe 0xff 0xd9 0xff 0xf8 0xff 0x00 0x00 0xcd
    0xff 0x0a 0x00 0x03 0x00 0xd1 0xff 0xe6 0xff 0x02 0x00 0x3a 0x00 0x12 0x00 0x01 0x00 0xc2
    0xff 0xde 0xff 0x00 0x00 0x41 0x00 0xdb 0xff 0x01 0x00 0xc5 0xff 0x32 0x00 0xfe 0xff 0x3a
    0x00 0xc3 0xff 0x01 0x00 0xa1 0xff 0xea 0xff 0x00 0x00 0x60 0x00 0x1b 0x00 0xff 0xff 0x65
    0x00 0x12 0x00 0xff 0xff 0x6b 0x00 0x06 0x00 0xff 0xff 0xbb 0xff 0xa5 0xff 0xff 0xff 0x75
    0x00 0xf4 0xff 0x01 0x00 0x7b 0x00 0xdf 0xff 0x00 0x00 0x7f 0xff 0xfe 0xff 0x03 0x00 0x7b
    0x00 0xd8 0xff 0x00 0x00 0x81 0x00 0xd7 0xff 0x02 0x00 0x89 0x00 0xe4 0xff 0x00 0x00 0x80
    0xff 0xc1 0xff 0x00 0x00 0x94 0x00 0xd1 0xff 0x00 0x00 0x98 0x00 0xd7 0xff 0x00 0x00 0x61
    0xff 0xf7 0xff 0x00 0x00 0x9b 0x00 0xcf 0xff 0x02 0x00 0x00 0x01 0x00 0x00 0x00 0x00 0x01
    0x01 0x00 0x00 0x00 0x01 0x01 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
    0x00 0x00 0x00 0x00

0x04 0x00 0x01 0x00 0xfe 0xff 0x05 0x00 0xff 0xff 0x01 0x00 0x10 0x00 0x04 0x00 0x01 0x00 0xe6
    0xff 0x16 0x00 0xfe 0xff 0x0b 0x00 0xd0 0xff 0x01 0x00 0x0e 0x00 0x30 0x00 0x00 0x00 0x3d
    0x00 0xf5 0xff 0x00 0x00 0xbd 0xff 0x1b 0x00 0x00 0x00 0x21 0x00 0x41 0x00 0x00 0x00 0x45
    0x00 0xdf 0xff 0x00 0x00 0xef 0xff 0xb3 0xff 0x02 0x00 0xd0 0xff 0xbc 0xff 0xfe 0xff 0x53
    0x00 0x11 0x00 0x02 0x00 0x55 0x00 0x03 0x00 0xfe 0xff 0x2c 0x00 0xb7 0xff 0xff 0xff 0xb5
    0xff 0xd4 0xff 0x02 0x00 0x5b 0x00 0x25 0x00 0x02 0x00 0x64 0x00 0xf0 0xff 0xfe 0xff 0x4a
    0x00 0xb9 0xff 0xff 0xff 0x6b 0x00 0x11 0x00 0x00 0x00 0x61 0x00 0x32 0x00 0x01 0x00 0x5f
    0x00 0x36 0x00 0xff 0xff 0x67 0x00 0xab 0xff 0xff 0xff 0x89 0x00 0xf6 0xff 0xfe 0xff 0x7e
    0x00 0x46 0x00 0xff 0xff 0x96 0x00 0x17 0x00 0x00 0x00 0x8f 0x00 0x34 0x00 0x00 0x00 0xb5
    0x00 0x0d 0x00 0x00 0x00 0xaa 0x00 0x48 0x00 0x00 0x00 0xb2 0x00 0xcc 0xff 0x03 0x00 0xb2
    0x00 0x34 0x00 0x00 0x00 0xbd 0x00 0xff 0xff 0x02 0x00 0xbb 0x00 0x22 0x00 0x02 0x00 0xbf
    0x00 0x30 0x00 0x02 0x00 0xc8 0x00 0x0e 0x00 0x00 0x00 0xc0 0x00 0xb0 0xff 0xfe 0xff 0xd4
    0x00 0x17 0x00 0x00 0x00 0xd8 0x00 0xf7 0xff 0x03 0x00 0xe3 0x00 0xd4 0xff 0x00 0x00 0xe9
    0x00 0x10 0x00 0x02 0x00 0xea 0x00 0xfb 0xff 0x00 0x00 0xec 0x00 0xe0 0xff 0x00 0x00 0xf5
    0x00 0xdf 0xff 0x02 0x00 0x01 0x01 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x01 0x00
    0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
    0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00

0xfd 0xff 0xed 0xff 0x02 0x00 0xec 0xff 0x13 0x00 0xff 0xff 0xdf 0xff 0xfd 0xff 0x00 0x00 0xd5
    0xff 0xeb 0xff 0x02 0x00 0x32 0x00 0x0d 0x00 0xff 0xff 0xe1 0xff 0x2d 0x00 0xff 0xff 0xcd
    0xff 0xe2 0xff 0x02 0x00 0x11 0x00 0xbc 0xff 0x01 0x00 0xbb 0xff 0x2a 0x00 0xfe 0xff 0x0d
    0x00 0x53 0x00 0xff 0xff 0x1e 0x00 0xb0 0xff 0x00 0x00 0x56 0x00 0xdc 0xff 0x00 0x00 0xe5
    0xff 0x60 0x00 0xff 0xff 0xb0 0xff 0x4d 0x00 0xfe 0xff 0x53 0x00 0xb3 0xff 0x00 0x00 0xb2
```

```
    0xff 0xa6 0xff 0x02 0x00 0xb1 0xff 0x5a 0x00 0x00 0x00 0xb2 0xff 0xa4 0xff 0x02 0x00 0x03
    0x00 0x87 0xff 0x02 0x00 0xf5 0xff 0x86 0xff 0x00 0x00 0x86 0xff 0x11 0x00 0x02 0x00 0x7b
    0xff 0xda 0xff 0x00 0x00 0x6d 0xff 0x3d 0x00 0x00 0x00 0x61 0xff 0x0e 0x00 0x03 0x00 0x6b
    0xff 0x4a 0x00 0xfe 0xff 0x58 0xff 0xd9 0xff 0x00 0x00 0x55 0xff 0x4e 0x00 0xfe 0xff 0x4a
    0xff 0xce 0xff 0x00 0x00 0x2a 0xff 0x11 0x00 0x00 0x00 0x22 0xff 0x15 0x00 0x02 0x00 0x21
    0xff 0xf0 0xff 0x02 0x00 0x16 0xff 0xf4 0xff 0x00 0x00 0x0b 0xff 0x1d 0x00 0x02 0x00 0x06
    0xff 0x02 0x00 0x02 0x00 0x04 0xff 0xb0 0xff 0x00 0x00 0xf9 0xfe 0xcb 0xff 0x00 0x00 0x01
    0x01 0x00 0x01 0x01 0x00 0x01 0x01 0x01 0x01 0x00 0x01 0x01 0x01 0x00 0x01 0x00 0x01 0x00
    0x00 0x00 0x00 0x00 0x01 0x01 0x01 0x01 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00

0x0a 0x00 0x00 0x00 0x01 0x00 0xec 0xff 0xef 0xff 0x02 0x00 0xec 0xff 0x12 0x00 0x00 0x00 0xe9
    0xff 0xe4 0xff 0x02 0x00 0xd6 0xff 0xf5 0xff 0x00 0x00 0xdd 0xff 0xde 0xff 0x01 0x00 0xd0
    0xff 0x0a 0x00 0x03 0x00 0xc9 0xff 0x15 0x00 0x03 0x00 0xc9 0xff 0xe0 0xff 0x02 0x00 0x3e
    0x00 0xec 0xff 0x01 0x00 0x23 0x00 0xc8 0xff 0xff 0xff 0x40 0x00 0x14 0x00 0xff 0xff 0xd7
    0xff 0x38 0x00 0xff 0xff 0x3b 0x00 0xd6 0xff 0xff 0xff 0x21 0x00 0xbb 0xff 0x01 0x00 0x3b
    0x00 0xbe 0xff 0x01 0x00 0x2a 0x00 0x54 0x00 0xff 0xff 0xc3 0xff 0x48 0x00 0xfe 0xff 0xc8
    0xff 0x4f 0x00 0xfe 0xff 0x9d 0xff 0xf2 0xff 0x00 0x00 0xbe 0xff 0xa7 0xff 0x01 0x00 0x78
    0x00 0xf4 0xff 0xff 0xff 0x36 0x00 0x94 0xff 0x02 0x00 0x8a 0xff 0x2c 0x00 0x00 0x00 0x82
    0xff 0x06 0x00 0x03 0x00 0x99 0xff 0x58 0x00 0xfe 0xff 0x7b 0xff 0xcc 0xff 0x00 0x00 0x7d
    0xff 0x43 0x00 0x03 0x00 0x73 0xff 0x31 0x00 0x03 0x00 0x8c 0x00 0x3f 0x00 0xff 0xff 0x5e
    0x00 0x86 0xff 0x00 0x00 0x72 0x00 0x97 0xff 0x00 0x00 0x63 0xff 0x01 0x00 0x03 0x00 0x99
    0x00 0x26 0x00 0x01 0x00 0x87 0x00 0x9f 0xff 0x00 0x00 0xa1 0x00 0xcd 0xff 0x00 0x00 0xad
    0x00 0xe0 0xff 0x00 0x00 0xaf 0x00 0xdc 0xff 0x02 0x00 0xa4 0x00 0xb5 0xff 0x00 0x00 0x00
    0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x00
    0x00 0x01 0x01 0x00 0x00 0x00 0x01 0x00 0x01 0x01 0x00 0x00 0x00 0x01 0x00 0x00 0x00 0x00
    0x00 0x00


进程已结束，退出代码 0
```

附录 B　python 程序

2.1　main.py

```
# -*- coding: utf-8 -*-
"""
Created on Sun Jan 28 18:13:36 2018

@author: Utkarsh
"""



import cv2
import numpy as np
import pandas as pd
```

```
import skimage.morphology as sm
from skimage import filters
import skimage
import os
import warnings
warnings.simplefilter("ignore")

from Feature_Extraction.src.getTerminationBifurcation import getTerminationBifurcation
from Feature_Extraction.src.removeSpuriousMinutiae import removeSpuriousMinutiae
from Feature_Extraction.src.getGradient import getGradient
from Feature_Extraction.src.CommonFunctions import *
from Feature_Extraction.src.extractMinutiaeFeatures import extractMinutiaeFeatures
from Feature_Extraction.src.matchMinutiae import getSimilarity


def getFeatures(img_id, featureNum=20):
    try:
        Features = pd.read_excel(os.path.join(FEATURE_DIR, "{:02d}.xlsx".format(img_id)))
        return Features[:featureNum]
    except FileNotFoundError:
        pass
    img_name = "{:02d}.tif".format(img_id)
    centerdf = pd.read_excel(os.path.join(DATA_DIR, "new_center.xlsx")).astype(int)
    center = tuple(centerdf.loc[img_id-1])
    # print("center :{}".format(center))
    img = cv2.imread('../rotated/'+img_name, 0)
    # 获取二值化图像
    img_01 = np.uint8(img > 128)
    # skeletonize细化图像，细化用于减少二值图像中的每个连通分量，到一个单像素宽的骨架。
    skel = sm.skeletonize(img_01)
    skel = np.uint8(skel) * 255
    # 未经细化的二值图像
    mask = img_01 * 255
    mask = sm.dilation(mask, sm.disk(1))
    cv2.imwrite(os.path.join(RESULT_DIR, "{:02d}mask.png".format(img_id)), mask)

    foregroundArea = sm.closing(mask, sm.disk(15))
    edge_filter = sm.erosion(foregroundArea, sm.disk(15))

    cv2.imwrite(os.path.join(RESULT_DIR, "{:02d}edge_filter.png".format(img_id)), edge_filter)
    # gradientX, gradientY = getGradient(mask)

    # 原始端点和分叉点
    (minutiaeTerm, minutiaeBif) = getTerminationBifurcation(skel, mask)

    Features = extractMinutiaeFeatures(skel, minutiaeTerm, minutiaeBif, center)
    Features = removeSpuriousMinutiae(Features, edge_filter, 5)#[:NUM_POINTS]
```

```
    # print(len(Features))
    Features.to_excel(os.path.join(FEATURE_DIR, "{:02d}.xlsx".format(img_id)))
    # print("Features:\n {}".format(Features))
    ShowResults(skel, Features, edge_filter, center, img_id, False)
    return Features[:featureNum]

def print_all_Features_len():
    for i in range(1, NUM_PIC+1):
        Features = getFeatures(i, 200)
        print(len(Features))

def print_all_Bytes():
    for i in range(1, NUM_PIC+1):
        Features = getFeatures(i, 20)
        prettyBytesStr = getPrettyFeaturesBytes(Features)
        print(prettyBytesStr)
        print()

def get_match_result(featureNum=20):
    similarityMat = np.zeros((NUM_PIC, NUM_PIC))
    for i in range(NUM_PIC):
        for j in range(NUM_PIC):
            Features1 = getFeatures(i+1, featureNum)
            Features2 = getFeatures(j+1, featureNum)
            total_potential = getSimilarity(Features1, Features2)
            similarityMat[i, j] = total_potential
    plot_mat(similarityMat, name="similarityMat{}".format(featureNum))
    pd.DataFrame(similarityMat).to_excel(os.path.join(FEATURE_DIR,
        "similarityMat{}.xlsx".format(featureNum)))
    return similarityMat

def getMatError(mat1, mat2):
    errMat = np.abs(mat1-mat2)
    avgErr = np.mean(errMat)
    maxErr = np.max(errMat)
    return avgErr, maxErr

def compare_match_result(featureNumLst=[10, 15, 20]):
    lstLen = len(featureNumLst)
    avgErrMat = np.zeros((lstLen, lstLen))
    maxErrMat = np.zeros((lstLen, lstLen))
    similarityMatLit = []
    for featureNum in featureNumLst:
        similarityMatLit.append(get_match_result(featureNum))
    for i in range(lstLen):
        for j in range(lstLen):
            avgErr, maxErr = getMatError(similarityMatLit[i], similarityMatLit[j])
```

```
            avgErrMat[i, j] = avgErr
            maxErrMat[i, j] = maxErr
    pd.DataFrame(avgErrMat).to_excel(os.path.join(FEATURE_DIR, "avgErrMat.xlsx"))
    pd.DataFrame(maxErrMat).to_excel(os.path.join(FEATURE_DIR, "maxErrMat.xlsx"))

if __name__ == '__main__':
    # print_all_Features_len()
    print_all_Bytes()
    compare_match_result()
```

2.2 CommonFunctions.py

```
import numpy as np
import pandas as pd
import skimage
import os
import cv2
import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib.backends.backend_pdf import PdfPages
import skimage
import imutils


PROCESS_DIR = "../process"
RESULT_DIR = "../result"
FEATURE_DIR = "../featuresData"
DATA_DIR = '../../../data'
IMG_SRC_DIR = '../../../data/img'

NUM_PIC = 16
NUM_POINTS = 10

def plt_show(img,name=None, ax=None):
    if ax == None:
        fig=plt.figure()
        ax = fig.add_subplot(111)
    if len(img.shape) == 3:
        ax.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
        #opencv读取的格式默认是BGR
    else:
        ax.imshow(img, cmap='gray')
    ax.axis("off")
    ax.set_title(name)
    plt.show()
```

```
def plot_mat(mat, save=True, name="mat"):
    plt.figure()
    plt.imshow(mat, cmap=plt.cm.hot)
    plt.colorbar()
    if save:
        pdf = PdfPages("../{}.pdf".format(name))
        pdf.savefig()
        pdf.close()
    else:
        plt.show()
    plt.close()

def External_ellipse(edge_filter, mask):
    contours = cv2.findContours(edge_filter, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_NONE)
    cnts = contours[0] if imutils.is_cv2() else contours[1] # 用imutils来判断是opencv是2还是2+
    cnt = cnts[0]
    """ellipse为三元组(center, axes, angle)
    center: 椭圆中心点坐标
    axes: 椭圆尺寸（即长短轴）
    angle: 旋转角度（顺时针方向）
    """
    ellipse = cv2.fitEllipse(cnt)
    # print(ellipse)

    return ellipse


def ShowResults(skel, Features, edge_filter, center, img_id, show=False):
    # Features["angle"] /= (180/np.pi)
    arrowsLen = 12
    arrowsWidth = 2
    radius = 4
    (rows, cols) = skel.shape
    DispImg = np.zeros((rows, cols, 3), np.uint8)
    DispImg[:, :, 0] = skel + .03 * edge_filter
    DispImg[:, :, 1] = skel
    DispImg[:, :, 2] = skel + .03 * edge_filter

    (rr, cc) = skimage.draw.circle(center[1], center[0], 7)
    skimage.draw.set_color(DispImg, (rr, cc), (0, 215, 255))

    for i, minutiae in Features.loc[Features["class"] == 0].iterrows():
        c = (255, 0, 0)
        (row, col) = int(minutiae["row"]), int(minutiae["col"])
        (row_, col_) = int(minutiae["row"]+arrowsLen*np.sin(minutiae["angle"])), \
```

```
                    int(minutiae["col"]+arrowsLen*np.cos(minutiae["angle"]))
        (rr, cc) = skimage.draw.circle(row, col, radius)
        skimage.draw.set_color(DispImg, (rr, cc), c)
        cv2.line(DispImg, (col, row), (col_, row_), c, arrowsWidth)

    for i, minutiae in Features.loc[Features["class"] == 1].iterrows():
        c = (0, 0, 255)
        (row, col) = int(minutiae["row"]), int(minutiae["col"])
        (row_, col_) = int(minutiae["row"]+arrowsLen*np.sin(minutiae["angle"])), \
                    int(minutiae["col"]+arrowsLen*np.cos(minutiae["angle"]))
        (rr, cc) = skimage.draw.circle(row, col, radius)
        skimage.draw.set_color(DispImg, (rr, cc), c)
        cv2.line(DispImg, (col, row), (col_, row_), c, arrowsWidth)

    img_name = "{:02d}result.png".format(img_id)
    cv2.imwrite(os.path.join(RESULT_DIR, img_name), DispImg)
    if show:
        plt_show(DispImg, name=img_name)


def normalizedArr(data, newMinValue=0, newMaxValue=1):
    minVal = np.amin(data)
    maxVal = np.amax(data)
    normalized = (data-minVal)*newMaxValue/(maxVal-minVal) + newMinValue
    return normalized

def getPrettyBytesStr(bytes):
    return ''.join(['0x%02x ' % b for b in bytes])

def fromPrettyBytesStr(prettyBytesStr):
    pass

def getPrettyFeaturesBytes(Features: pd.DataFrame):
    locArr = np.array(Features[["deltay", "deltax", "angle"]])

    Features[["deltay", "deltax", "angle", "class"]].astype(int).to_excel("../Features.xlsx")
    locArr = locArr.astype(np.int16)
    locBytes = locArr.tostring()

    # print(locBytes)
    # print(len(locBytes))
    # print(type(locBytes))
    # print(np.fromstring(locBytes, dtype=np.int16).reshape((-1, 3)))

    classArr = np.array(Features["class"]).astype(np.uint8)
    classBytes = classArr.tostring()
    # print(classBytes)
```

```
    # print(len(classBytes))
    # print(type(classBytes))
    # print(np.fromstring(classBytes, dtype=np.uint8))

    prettyBytesStr = getPrettyBytesStr(locBytes+classBytes)
    return prettyBytesStr
```

2.3 extractMinutiaeFeatures.py

```
import numpy as np
import cv2
import skimage
import math
import pandas as pd

from Feature_Extraction.src.CommonFunctions import *

def computeAngle(block, minutiaeType):
    (blkRows, blkCols) = np.shape(block)
    CenterX, CenterY = (blkRows-1)/2, (blkCols-1)/2
    if(minutiaeType.lower() == 'termination'):
        sumVal = 0
        for i in range(blkRows):
            for j in range(blkCols):
                if((i == 0 or i == blkRows-1 or j == 0 or j == blkCols-1) and block[i][j] != 0):
                    angle = math.atan2((i-CenterY),(j-CenterX))
                    sumVal += 1
        if(sumVal != 1):
            return None
        return(angle)
    elif(minutiaeType.lower() == 'bifurcation'):
        (blkRows, blkCols) = np.shape(block)
        CenterX, CenterY = (blkRows - 1) / 2, (blkCols - 1) / 2
        angle = []
        sumVal = 0
        for i in range(blkRows):
            for j in range(blkCols):
                if ((i == 0 or i == blkRows - 1 or j == 0 or j == blkCols - 1) and block[i][j] !=
                    0):
                    angle.append(math.atan2(i - CenterY, j - CenterX))
                    sumVal += 1
        if(sumVal != 3):
            return None
        minid = np.argmin([np.abs(angle[2]-angle[1]), np.abs(angle[0]-angle[2]),
            np.abs(angle[1]-angle[0])])
```

```
        return angle[minid]


def extractMinutiaeFeatures(skel, minutiaeTerm, minutiaeBif, center):
    minutiaeTerm = skimage.measure.label(minutiaeTerm, connectivity=2)
    RP = skimage.measure.regionprops(minutiaeTerm)
    WindowSize = 3

    FeaturesTerm = pd.DataFrame(columns=["row", "col", "angle", "class", "distance", "deltay",
        "deltax"])
    for i, Term in enumerate(RP):
        (row, col) = np.int16(np.round(Term['Centroid']))
        block = skel[row-WindowSize:row+WindowSize+1, col-WindowSize:col+WindowSize+1]
        angle = computeAngle(block, 'termination')
        deltay = row-center[1]
        deltax = col-center[0]
        distance = (deltax)**2 + (deltay)**2
        if angle is not None:
            FeaturesTerm.loc[i] = np.array([row, col, angle, 0, distance, deltay, deltax])

    minutiaeBif = skimage.measure.label(minutiaeBif, connectivity=2)
    RP = skimage.measure.regionprops(minutiaeBif)
    FeaturesBif = pd.DataFrame(columns=["row", "col", "angle", "class", "distance", "deltay",
        "deltax"])
    for i, Bif in enumerate(RP):
        (row, col) = np.int16(np.round(Bif['Centroid']))
        block = skel[row-WindowSize:row+WindowSize+1, col-WindowSize:col+WindowSize+1]
        angle = computeAngle(block, 'Bifurcation')
        deltay = row-center[1]
        deltax = col-center[0]
        distance = (deltax)**2 + (deltay)**2
        if angle is not None:
            FeaturesBif.loc[i] = np.array([row, col, angle, 1, distance, deltay, deltax])
    Features = FeaturesTerm.append(FeaturesBif)
    return Features.reset_index(drop=True)
```

2.4 getGradient.py

```
import cv2
import numpy as np
import pandas as pd
import skimage.morphology as sm
from skimage import filters
import skimage
```

```
import os
import warnings

from Feature_Extraction.src.CommonFunctions import *
from Feature_Extraction.src.getTerminationBifurcation import getTerminationBifurcation


def getGradient(img):
    SCALE = 1/2
    SOBEL_KSIZE = 5
    GUASSIAN_KSIZE = (7, 7)
    img = cv2.resize(img, (0, 0), fx=SCALE, fy=SCALE)
    cv2.imwrite(os.path.join(PROCESS_DIR, "0img.png"), img)
    gradX = cv2.Sobel(img/255., cv2.CV_16S, 1, 0, ksize=SOBEL_KSIZE)
    gradY = cv2.Sobel(img/255., cv2.CV_16S, 0, 1, ksize=SOBEL_KSIZE)

    # gradX = normalizedArr(gradX, -1,1)
    # gradY = normalizedArr(gradY, -1,1)

    dxdy = 2 * gradX*gradY + 1e-3
    dx2_dy2 = gradX**2 - gradY**2
    theta = np.arctan(dx2_dy2/dxdy)
    cv2.imwrite(os.path.join(PROCESS_DIR, "1theta.png"), normalizedArr(theta, 0,
        255).astype(np.uint8))

    phix = np.cos(2*theta)
    phiy = np.sin(2*theta)

    O = .5*np.arctan(2*(phiy/phix))
    cv2.imwrite(os.path.join(PROCESS_DIR, "2imgO_before_Gaussian.png"),
              normalizedArr(O, 0, 255).astype(np.uint8))

    phix = cv2.GaussianBlur(phix, GUASSIAN_KSIZE, 15)
    phiy = cv2.GaussianBlur(phiy, GUASSIAN_KSIZE, 15)

    O = .5*np.arctan(2*(phiy/phix))
    imgO = normalizedArr(O, 0, 255).astype(np.uint8)
    cv2.imwrite(os.path.join(PROCESS_DIR, "3imgO_after_Gaussian.png"), imgO)

    imgOEdge = normalizedArr(filters.sobel(imgO), 0, 255)
    cv2.imwrite(os.path.join(PROCESS_DIR, "4imgOEdge.png"), imgOEdge)

    imgOEdge = sm.dilation(imgOEdge, sm.disk(2))
    imgOEdge = sm.closing(imgOEdge, sm.disk(4))
    cv2.imwrite(os.path.join(PROCESS_DIR, "5imgOEdge_closing.png"), imgOEdge)

    THRESHOLD = 200
```

```
    imgOEdge = (imgOEdge > THRESHOLD)*255
    cv2.imwrite(os.path.join(PROCESS_DIR, "6imgOEdge_01.png"), imgOEdge)
    # plt_show(imgOEdge>THRESHOLD)
    imgOSkel = sm.skeletonize(imgOEdge > THRESHOLD)*255
    cv2.imwrite(os.path.join(PROCESS_DIR, "7imgOSkel.png"), imgOSkel)

    return gradX, gradY
```

2.5 getTerminationBifurcation.py

```
# -*- coding: utf-8 -*-
"""
Created on Sun Jan 28 18:12:44 2018

@author: Utkarsh
"""

import numpy as np
from skimage.morphology import convex_hull_image, erosion
from skimage.morphology import square


def getTerminationBifurcation(img, mask):
    img = img == 255
    (rows, cols) = img.shape
    minutiaeTerm = np.zeros(img.shape)
    minutiaeBif = np.zeros(img.shape)

    for i in range(1, rows - 1):
        for j in range(1, cols - 1):
            if (img[i][j] == 1):
                block = img[i - 1:i + 2, j - 1:j + 2]
                block_val = np.sum(block)
                if (block_val == 2):
                    minutiaeTerm[i, j] = 1
                elif (block_val == 4):
                    minutiaeBif[i, j] = 1

    mask = convex_hull_image(mask > 0)
    mask = erosion(mask, square(5)) # Structuing element for mask erosion = square(5)
    minutiaeTerm = np.uint8(mask) * minutiaeTerm
    return (minutiaeTerm, minutiaeBif)
```

2.6 hungarian.py

```
#!/usr/bin/python
"""
Implementation of the Hungarian (Munkres) Algorithm using Python and NumPy
References: http://www.ams.jhu.edu/~castello/362/Handouts/hungarian.pdf
       http://weber.ucsd.edu/~vcrawfor/hungar.pdf
       http://en.wikipedia.org/wiki/Hungarian_algorithm
       http://www.public.iastate.edu/~ddoty/HungarianAlgorithm.html
       http://www.clapper.org/software/python/munkres/
"""

# Module Information.
__version__ = "1.1.1"
__author__ = "Thom Dedecko"
__url__ = "http://github.com/tdedecko/hungarian-algorithm"
__copyright__ = "(c) 2010 Thom Dedecko"
__license__ = "MIT License"


class HungarianError(Exception):
    pass

# Import numpy. Error if fails
try:
    import numpy as np
except ImportError:
    raise HungarianError("NumPy is not installed.")


class Hungarian:
    """
    Implementation of the Hungarian (Munkres) Algorithm using np.

    Usage:
        hungarian = Hungarian(cost_matrix)
        hungarian.calculate()
    or
        hungarian = Hungarian()
        hungarian.calculate(cost_matrix)

    Handle Profit matrix:
        hungarian = Hungarian(profit_matrix, is_profit_matrix=True)
    or
        cost_matrix = Hungarian.make_cost_matrix(profit_matrix)

    The matrix will be automatically padded if it is not square.
```

```
For that numpy's resize function is used, which automatically adds 0's to any row/column
    that is added

Get results and total potential after calculation:
   hungarian.get_results()
   hungarian.get_total_potential()
"""

def __init__(self, input_matrix=None, is_profit_matrix=False):
   """
   input_matrix is a List of Lists.
   input_matrix is assumed to be a cost matrix unless is_profit_matrix is True.
   """
   if input_matrix is not None:
      # Save input
      my_matrix = np.array(input_matrix)
      self._input_matrix = np.array(input_matrix)
      self._maxColumn = my_matrix.shape[1]
      self._maxRow = my_matrix.shape[0]

      # Adds 0s if any columns/rows are added. Otherwise stays unaltered
      matrix_size = max(self._maxColumn, self._maxRow)
      pad_columns = matrix_size - self._maxRow
      pad_rows = matrix_size - self._maxColumn
      my_matrix = np.pad(my_matrix, ((0,pad_columns),(0,pad_rows)), 'constant',
          constant_values=(0))

      # Convert matrix to profit matrix if necessary
      if is_profit_matrix:
         my_matrix = self.make_cost_matrix(my_matrix)

      self._cost_matrix = my_matrix
      self._size = len(my_matrix)
      self._shape = my_matrix.shape

      # Results from algorithm.
      self._results = []
      self._totalPotential = 0
   else:
      self._cost_matrix = None

def get_results(self):
   """Get results after calculation."""
   return self._results

def get_total_potential(self):
   """Returns expected value after calculation."""
```

```
        return self._totalPotential

    def calculate(self, input_matrix=None, is_profit_matrix=False):
        """
        Implementation of the Hungarian (Munkres) Algorithm.

        input_matrix is a List of Lists.
        input_matrix is assumed to be a cost matrix unless is_profit_matrix is True.
        """
        # Handle invalid and new matrix inputs.
        if input_matrix is None and self._cost_matrix is None:
            raise HungarianError("Invalid input")
        elif input_matrix is not None:
            self.__init__(input_matrix, is_profit_matrix)

        result_matrix = self._cost_matrix.copy()

        # Step 1: Subtract row mins from each row.
        for index, row in enumerate(result_matrix):
            result_matrix[index] -= row.min()

        # Step 2: Subtract column mins from each column.
        for index, column in enumerate(result_matrix.T):
            result_matrix[:, index] -= column.min()

        # Step 3: Use minimum number of lines to cover all zeros in the matrix.
        # If the total covered rows+columns is not equal to the matrix size then adjust matrix
            and repeat.
        total_covered = 0
        while total_covered < self._size:
            # Find minimum number of lines to cover all zeros in the matrix and find total
                covered rows and columns.
            cover_zeros = CoverZeros(result_matrix)
            covered_rows = cover_zeros.get_covered_rows()
            covered_columns = cover_zeros.get_covered_columns()
            total_covered = len(covered_rows) + len(covered_columns)

            # if the total covered rows+columns is not equal to the matrix size then adjust it
                by min uncovered num (m).
            if total_covered < self._size:
                result_matrix = self._adjust_matrix_by_min_uncovered_num(result_matrix,
                    covered_rows, covered_columns)

        # Step 4: Starting with the top row, work your way downwards as you make assignments.
        # Find single zeros in rows or columns.
        # Add them to final result and remove them and their associated row/column from the
            matrix.
```

```
    expected_results = min(self._maxColumn, self._maxRow)
    zero_locations = (result_matrix == 0)
    while len(self._results) != expected_results:

        # If number of zeros in the matrix is zero before finding all the results then an
            error has occurred.
        if not zero_locations.any():
            raise HungarianError("Unable to find results. Algorithm has failed.")

        # Find results and mark rows and columns for deletion
        matched_rows, matched_columns = self.__find_matches(zero_locations)

        # Make arbitrary selection
        total_matched = len(matched_rows) + len(matched_columns)
        if total_matched == 0:
            matched_rows, matched_columns = self.select_arbitrary_match(zero_locations)

        # Delete rows and columns
        for row in matched_rows:
            zero_locations[row] = False
        for column in matched_columns:
            zero_locations[:, column] = False

        # Save Results
        self.__set_results(zip(matched_rows, matched_columns))

    # Calculate total potential
    value = 0
    for row, column in self._results:
        value += self._input_matrix[row, column]
    self._totalPotential = value

@staticmethod
def make_cost_matrix(profit_matrix):
    """
    Converts a profit matrix into a cost matrix.
    Expects NumPy objects as input.
    """
    # subtract profit matrix from a matrix made of the max value of the profit matrix
    matrix_shape = profit_matrix.shape
    offset_matrix = np.ones(matrix_shape, dtype=int) * profit_matrix.max()
    cost_matrix = offset_matrix - profit_matrix
    return cost_matrix

def _adjust_matrix_by_min_uncovered_num(self, result_matrix, covered_rows, covered_columns):
    """Subtract m from every uncovered number and add m to every element covered with two
        lines."""
```

```
    # Calculate minimum uncovered number (m)
    elements = []
    for row_index, row in enumerate(result_matrix):
        if row_index not in covered_rows:
            for index, element in enumerate(row):
                if index not in covered_columns:
                    elements.append(element)
    min_uncovered_num = min(elements)

    # Add m to every covered element
    adjusted_matrix = result_matrix
    for row in covered_rows:
        adjusted_matrix[row] += min_uncovered_num
    for column in covered_columns:
        adjusted_matrix[:, column] += min_uncovered_num

    # Subtract m from every element
    m_matrix = np.ones(self._shape, dtype=int) * min_uncovered_num
    adjusted_matrix -= m_matrix

    return adjusted_matrix

def __find_matches(self, zero_locations):
    """Returns rows and columns with matches in them."""
    marked_rows = np.array([], dtype=int)
    marked_columns = np.array([], dtype=int)

    # Mark rows and columns with matches
    # Iterate over rows
    for index, row in enumerate(zero_locations):
        row_index = np.array([index])
        if np.sum(row) == 1:
            column_index, = np.where(row)
            marked_rows, marked_columns = self.__mark_rows_and_columns(marked_rows,
                marked_columns, row_index,
                                                                       column_index)

    # Iterate over columns
    for index, column in enumerate(zero_locations.T):
        column_index = np.array([index])
        if np.sum(column) == 1:
            row_index, = np.where(column)
            marked_rows, marked_columns = self.__mark_rows_and_columns(marked_rows,
                marked_columns, row_index,
                                                                       column_index)

    return marked_rows, marked_columns
```

```
    @staticmethod
    def __mark_rows_and_columns(marked_rows, marked_columns, row_index, column_index):
        """Check if column or row is marked. If not marked then mark it."""
        new_marked_rows = marked_rows
        new_marked_columns = marked_columns
        if not (marked_rows == row_index).any() and not (marked_columns == column_index).any():
            new_marked_rows = np.insert(marked_rows, len(marked_rows), row_index)
            new_marked_columns = np.insert(marked_columns, len(marked_columns), column_index)
        return new_marked_rows, new_marked_columns

    @staticmethod
    def select_arbitrary_match(zero_locations):
        """Selects row column combination with minimum number of zeros in it."""
        # Count number of zeros in row and column combinations
        rows, columns = np.where(zero_locations)
        zero_count = []
        for index, row in enumerate(rows):
            total_zeros = np.sum(zero_locations[row]) + np.sum(zero_locations[:, columns[index]])
            zero_count.append(total_zeros)

        # Get the row column combination with the minimum number of zeros.
        indices = zero_count.index(min(zero_count))
        row = np.array([rows[indices]])
        column = np.array([columns[indices]])

        return row, column

    def __set_results(self, result_lists):
        """Set results during calculation."""
        # Check if results values are out of bound from input matrix (because of matrix being
            padded).
        # Add results to results list.
        for result in result_lists:
            row, column = result
            if row < self._maxRow and column < self._maxColumn:
                new_result = (int(row), int(column))
                self._results.append(new_result)


class CoverZeros:
    """
    Use minimum number of lines to cover all zeros in the matrix.
    Algorithm based on: http://weber.ucsd.edu/~vcrawfor/hungar.pdf
    """

    def __init__(self, matrix):
```

```
        """
        Input a matrix and save it as a boolean matrix to designate zero locations.
        Run calculation procedure to generate results.
        """
        # Find zeros in matrix
        self._zero_locations = (matrix == 0)
        self._shape = matrix.shape

        # Choices starts without any choices made.
        self._choices = np.zeros(self._shape, dtype=bool)

        self._marked_rows = []
        self._marked_columns = []

        # marks rows and columns
        self.__calculate()

        # Draw lines through all unmarked rows and all marked columns.
        self._covered_rows = list(set(range(self._shape[0])) - set(self._marked_rows))
        self._covered_columns = self._marked_columns

    def get_covered_rows(self):
        """Return list of covered rows."""
        return self._covered_rows

    def get_covered_columns(self):
        """Return list of covered columns."""
        return self._covered_columns

    def __calculate(self):
        """
        Calculates minimum number of lines necessary to cover all zeros in a matrix.
        Algorithm based on: http://weber.ucsd.edu/~vcrawfor/hungar.pdf
        """
        while True:
            # Erase all marks.
            self._marked_rows = []
            self._marked_columns = []

            # Mark all rows in which no choice has been made.
            for index, row in enumerate(self._choices):
                if not row.any():
                    self._marked_rows.append(index)

            # If no marked rows then finish.
            if not self._marked_rows:
                return True
```

```
        # Mark all columns not already marked which have zeros in marked rows.
        num_marked_columns = self.__mark_new_columns_with_zeros_in_marked_rows()

        # If no new marked columns then finish.
        if num_marked_columns == 0:
            return True

        # While there is some choice in every marked column.
        while self.__choice_in_all_marked_columns():
            # Some Choice in every marked column.

            # Mark all rows not already marked which have choices in marked columns.
            num_marked_rows = self.__mark_new_rows_with_choices_in_marked_columns()

            # If no new marks then Finish.
            if num_marked_rows == 0:
                return True

            # Mark all columns not already marked which have zeros in marked rows.
            num_marked_columns = self.__mark_new_columns_with_zeros_in_marked_rows()

            # If no new marked columns then finish.
            if num_marked_columns == 0:
                return True

        # No choice in one or more marked columns.
        # Find a marked column that does not have a choice.
        choice_column_index = self.__find_marked_column_without_choice()

        while choice_column_index is not None:
            # Find a zero in the column indexed that does not have a row with a choice.
            choice_row_index = self.__find_row_without_choice(choice_column_index)

            # Check if an available row was found.
            new_choice_column_index = None
            if choice_row_index is None:
                # Find a good row to accomodate swap. Find its column pair.
                choice_row_index, new_choice_column_index = \
                    self.__find_best_choice_row_and_new_column(choice_column_index)

                # Delete old choice.
                self._choices[choice_row_index, new_choice_column_index] = False

            # Set zero to choice.
            self._choices[choice_row_index, choice_column_index] = True
```

```
            # Loop again if choice is added to a row with a choice already in it.
            choice_column_index = new_choice_column_index

    def __mark_new_columns_with_zeros_in_marked_rows(self):
        """Mark all columns not already marked which have zeros in marked rows."""
        num_marked_columns = 0
        for index, column in enumerate(self._zero_locations.T):
            if index not in self._marked_columns:
                if column.any():
                    row_indices, = np.where(column)
                    zeros_in_marked_rows = (set(self._marked_rows) & set(row_indices)) != set([])
                    if zeros_in_marked_rows:
                        self._marked_columns.append(index)
                        num_marked_columns += 1
        return num_marked_columns

    def __mark_new_rows_with_choices_in_marked_columns(self):
        """Mark all rows not already marked which have choices in marked columns."""
        num_marked_rows = 0
        for index, row in enumerate(self._choices):
            if index not in self._marked_rows:
                if row.any():
                    column_index, = np.where(row)
                    if column_index in self._marked_columns:
                        self._marked_rows.append(index)
                        num_marked_rows += 1
        return num_marked_rows

    def __choice_in_all_marked_columns(self):
        """Return Boolean True if there is a choice in all marked columns. Returns boolean False
            otherwise."""
        for column_index in self._marked_columns:
            if not self._choices[:, column_index].any():
                return False
        return True

    def __find_marked_column_without_choice(self):
        """Find a marked column that does not have a choice."""
        for column_index in self._marked_columns:
            if not self._choices[:, column_index].any():
                return column_index

        raise HungarianError(
            "Could not find a column without a choice. Failed to cover matrix zeros. Algorithm
                has failed.")

    def __find_row_without_choice(self, choice_column_index):
```

```
        """Find a row without a choice in it for the column indexed. If a row does not exist
            then return None."""
        row_indices, = np.where(self._zero_locations[:, choice_column_index])
        for row_index in row_indices:
            if not self._choices[row_index].any():
                return row_index

        # All rows have choices. Return None.
        return None

    def __find_best_choice_row_and_new_column(self, choice_column_index):
        """
        Find a row index to use for the choice so that the column that needs to be changed is
            optimal.
        Return a random row and column if unable to find an optimal selection.
        """
        row_indices, = np.where(self._zero_locations[:, choice_column_index])
        for row_index in row_indices:
            column_indices, = np.where(self._choices[row_index])
            column_index = column_indices[0]
            if self.__find_row_without_choice(column_index) is not None:
                return row_index, column_index

        # Cannot find optimal row and column. Return a random row and column.
        from random import shuffle

        shuffle(row_indices)
        column_index, = np.where(self._choices[row_indices[0]])
        return row_indices[0], column_index[0]


if __name__ == '__main__':
    profit_matrix = [
        [62, 75, 80, 93, 95, 97],
        [75, 80, 82, 85, 71, 97],
        [80, 75, 81, 98, 90, 97],
        [78, 82, 84, 80, 50, 98],
        [90, 85, 85, 80, 85, 99],
        [65, 75, 80, 75, 68, 96]]

    hungarian = Hungarian(profit_matrix, is_profit_matrix=True)
    hungarian.calculate()
    print("Expected value:\t\t543")
    print("Calculated value:\t", hungarian.get_total_potential()) # = 543
    print("Expected results:\n\t[(0, 4), (2, 3), (5, 5), (4, 0), (1, 1), (3, 2)]")
    print("Results:\n\t", hungarian.get_results())
    print("-" * 80)
```

```
    cost_matrix = [
        [4, 2, 8],
        [4, 3, 7],
        [3, 1, 6]]
    hungarian = Hungarian(cost_matrix)
    print('calculating...')
    hungarian.calculate()
    print("Expected value:\t\t12")
    print("Calculated value:\t", hungarian.get_total_potential()) # = 12
    print("Expected results:\n\t[(0, 1), (1, 0), (2, 2)]")
    print("Results:\n\t", hungarian.get_results())
    print("-" * 80)

    profit_matrix = [
        [62, 75, 80, 93, 0, 97],
        [75, 0, 82, 85, 71, 97],
        [80, 75, 81, 0, 90, 97],
        [78, 82, 0, 80, 50, 98],
        [0, 85, 85, 80, 85, 99],
        [65, 75, 80, 75, 68, 0]]
    hungarian = Hungarian()
    hungarian.calculate(profit_matrix, is_profit_matrix=True)
    print("Expected value:\t\t523")
    print("Calculated value:\t", hungarian.get_total_potential()) # = 523
    print("Expected results:\n\t[(0, 3), (2, 4), (3, 0), (5, 2), (1, 5), (4, 1)]")
    print("Results:\n\t", hungarian.get_results())
    print("-" * 80)
```

2.7 matchMinutiae.py

```
import numpy as np
import pandas as pd

from Feature_Extraction.src.hungarian import Hungarian
from sklearn.preprocessing import MinMaxScaler

from Feature_Extraction.src.CommonFunctions import *

def getSimilarity(Features1, Features2):
    featureNum = len(Features1)
    Features1Arr = np.array(Features1[["deltay", "deltax", "angle"]])
    class1Arr = np.array(Features1["class"])
    Features2Arr = np.array(Features2[["deltay", "deltax", "angle"]])
```

```
    class2Arr = np.array(Features2["class"])

    mm = MinMaxScaler((-1,1))
    # 将3个特征"deltay", "deltax", "angle"标准化到区间[0, 1]
    Features1Arr = mm.fit_transform(Features1Arr)
    Features2Arr = mm.fit_transform(Features2Arr)

    # 每两个特征点的匹配损失
    profitMatrix = np.zeros((featureNum, featureNum))
    for i in range(featureNum):
        locProfitArr = np.exp(-np.sum((Features2Arr-Features1Arr[i])**2, axis=1))
        locProfitArr *= (class1Arr[i] == class2Arr)
        neighborProfitArr = np.zeros_like(locProfitArr)
        for j in range(featureNum):
            dist1Arr = np.sum((Features1Arr-Features1Arr[i])**2, axis=1)
            neighbor1Id = np.argsort(dist1Arr)[1]
            dist2Arr = np.sum((Features2Arr-Features2Arr[i])**2, axis=1)
            neighbor2Id = np.argsort(dist2Arr)[1]
            if class1Arr[neighbor1Id] != class2Arr[neighbor2Id]:
                continue
            neighborProfitArr[j] += np.exp(
                -np.sum((Features1Arr[neighbor1Id]-Features1Arr[neighbor2Id])**2)
            )
        profitMatrix[i] = locProfitArr + neighborProfitArr
    # print(profitMatrix)
    hungarian = Hungarian()
    hungarian.calculate(profitMatrix, is_profit_matrix=True)
    result = hungarian.get_results()
    total_potential = hungarian.get_total_potential()/(2*featureNum)
    print(result)
    print("total_profit: {}".format(total_potential))
    return total_potential
```

2.8 preprocess.py

```
import cv2
import numpy as np
import skimage.morphology as sm
import skimage
import os

from Feature_Extraction.src.CommonFunctions import *

def enhance_dir(path, savepath):
```

```
    for imgfile in os.listdir(path):
        img_id = int(imgfile.rstrip(".tif"))
        print(imgfile)
        img = cv2.imread(os.path.join(path, imgfile), 0)
        # 获取二值化图像
        img_01 = np.uint8(img > 128)
        # 未经细化的二值图像
        mask = img_01 * 255
        edge_filter = skimage.morphology.closing(mask, sm.disk(20))
        edge_filter = skimage.morphology.erosion(edge_filter, sm.disk(5))
        # plt_show(foregroundArea)
        ellipse = External_ellipse(edge_filter, mask)
        center, axes, angle = ellipse

        if angle > 90:
            angle -= 180
        # print(angle)
        matRotate = cv2.getRotationMatrix2D(center, angle, 1.)
        rotated = cv2.warpAffine(img, matRotate, (mask.shape[1],mask.shape[0]))
        cv2.imwrite(os.path.join(savepath, imgfile), rotated)

        maskBGR = cv2.cvtColor(mask, cv2.COLOR_GRAY2BGR)
        rotatedBGR = cv2.cvtColor(rotated, cv2.COLOR_GRAY2BGR)
        cv2.ellipse(maskBGR, ellipse, color=(0, 0, 255), thickness=2)
        cat = np.concatenate((maskBGR, rotatedBGR), axis=1)
        cv2.imwrite(os.path.join(RESULT_DIR, "{:02d}rotate.png".format(img_id)), cat)




if __name__ == '__main__':
    enhance_dir("../enhanced", "../rotated")
```

2.9 removeSpuriousMinutiae.py

```
# -*- coding: utf-8 -*-
"""
Created on Tue Jan 30 18:44:22 2018

@author: Utkarsh
"""

import cv2
import numpy as np
import pandas as pd
```

```
import skimage.morphology
import skimage

from Feature_Extraction.src.CommonFunctions import *

def removeSpuriousMinutiae(Features, edge_filter, thresh):
    '''
    细节点去伪
    '''
    numPoints = len(Features)
    D = np.zeros((numPoints, numPoints))
    Features["valid"] = 1

    for i, minutiaei in Features.iterrows():
        (rowi, coli) = int(minutiaei["row"]), int(minutiaei["col"])
        if edge_filter[rowi, coli]==0:
            Features["valid"][i] = 0
    Features = Features.loc[Features["valid"] == 1].reset_index(drop=True)

    for i, minutiaei in Features.iterrows():
        (rowi, coli) = int(minutiaei["row"]), int(minutiaei["col"])
        for j in range(i):
            minutiaej = Features.loc[j]
            (rowj, colj) = int(minutiaej["row"]), int(minutiaej["col"])
            if np.sqrt((rowi-rowj)**2 - (coli-colj)**2) < thresh:
                Features["valid"][i] = 0
                Features["valid"][j] = 0
    Features = Features.loc[Features["valid"] == 1]
    Features = Features.sort_values(by="distance").reset_index(drop=True)
    return Features
```