随着生物识别技术的不断发展，人们发现每个人的指纹具有唯一性和不变性。因此指纹识别技术逐步发展为一种新的身份识别方式，并且凭借其良好的安全可靠性，大有取代传统身份识别方式的趋势。

本文简要介绍了指纹识别的基本步骤，分别是指纹图像预处理、指纹特征提取、指纹匹配。在图像预处理中，依次介绍了规格化处理、图像增强、二值化处理和细化处理的方法。预处理后将得到一幅宽度为一个像素的细化二值图像，然后通过特定的端点和交叉点的特征进行指纹匹配。论文中采用 MATLAB 编程实现全部算法。

关键词：指纹识别，图像处理，特征提取，特征匹配

Abstract

With the continuous development of Biometric Identification Technology,People found that each person's fingerprint has uniqueness and invariant.Therefore Fingerprint Identification Technology gradually developed as a new identity recognition mode，and with its good safety and reliability，it has replaced the traditional identification way trends.

This paper briefly introduces the basic step of Fingerprint Identification，they are Fingerprint image preprocessing, Fingerprint characteristic extraction,Fingerprint matching.In the Fingerprint image preprocessing,in turn introduced the normalized processing, Image enhancement, Binary treatment and Refining processing method.After pretreatment will get a picture for a pixel width of twenty-first-century binary image,then through the particular endpoint and intersection on the characteristics of the Fingerprint matching.This paper using Matlab programs all algorithm.

Key words Fingerprint identification Image processing Feature extraction Feature matching

目　　录

第 1 章 绪论

1.1 研究背景及意义

现代门禁系统是一种随着电子技术和计算机技术的发展而迅速发展起来的安防系统，一套现代化的、功能齐全的门禁系统，不仅可用于进出口控制，而且有助于单位内部的有序化管理。门禁系统是新型现代化公共安全管理系统，它集微机自动识别技术和现代安全管理措施为一体，涉及电子、机械、光学、计算机技术、通信技术和生物技术等诸多新技术，是重要部门出入口实现安全防范管理的有效保障[1]。

目前有很多的生物测定技术可用于身份认证，包括虹膜识别技术、视网膜识别技术、面部识别、签名识别、声音识别技术、指纹识别等，具有安全、可靠的特点，其中自动指纹识别系统是目前研究最多、最有应用前景的生物识别系统。指纹识别技术的应用十分广泛，指纹因具有终生不变性及稳定性,而且不同人指纹相同的概率几乎为零,因此指纹自动识别系统被广泛应用于案例分析、商业活动中的身份鉴别等领域。指纹识别技术的发展得益于现代电子集成制造技术的进步和快速可靠的算法的研究。

指纹门禁系统通过将用户的指纹特征与指纹特征数据库中的数据进行对比实现用户身份的鉴别，并不直接保存和使用用户的指纹图像信息，不会侵犯到用户的隐私信息，是当前技术最先进、应用最广泛的门禁系统。对生物识别(指纹识别)技术来说，被广泛应用意味着它能在影响亿万人的日常生活的各个地方使用。通过取代个人识别码和口令，生物识别（指纹识别）技术可以阻止非授权的 “访问”，可以防止盗用 ATM、蜂窝电话、智能卡、桌面 PC、工作站及其计算机网络；在通过电话、网络进行的金融交易时进行身份认证；在建筑物或工作场所生物识别技术（指纹识别）可以取代钥匙、证件、图章等。生物识别(指纹识别)技术的飞速发展及其广泛应用将开创个人身份鉴别的新时代。指纹所具有的唯一性、不变性、易于获取、分类存储有规律等特性使其成为生物鉴定学中最为成熟的方式[2]。

1.2 国内外研究现状及分析

指纹识别技术从早期的人工比对到现在采用计算机技术实现自动指纹识别，指纹对比更加准确，识别效率得到极大提高。自动指纹识别过程通常由指纹图像滤波增强、二值化、细化、特征提取以及指纹匹配等几个环节构成。指纹图像滤波增强的目的是将有噪声干扰的指纹图像变得更加清晰，使得指纹图像的脊线更黑，谷线更白，当前在实际指纹图像增强算法的应用中一般是几种滤波增强方式结合起来使用，主要的方案是基于傅里叶变换结合滤波和指纹图像点方向场的下上下滤波器；指纹图像二值化，是将指纹图像变成灰度值

只有 0 和 255 两种颜色的图像，当前，在自动指纹识别中常采用的是根据指纹图像的点方向场在指纹纹线方向和指纹纹线垂直方向上对指纹图像进行二值化处理；指纹图像细化是指删除指纹纹线的边缘像素，使之只有一个像素宽度，目前在自动指纹识别技术中常用的是 OPTA 算法的改进的图像模板细化算法；指纹特征提取，是将细化后使用计算机数字图像处理技术采集指纹图像中奇异点、端点、叉点等指纹特征数据，目前常用的特征提取算法是先对细化后的指纹图像进行初步去噪，然后提取特征点，再根据阈值去除伪特征点；指纹匹配，是指纹预留模板图像与输入样板图像中的所有特征点的匹配，目前在自动指纹识别系统中常采用可变大小的界限盒的指纹特征匹配算法。

目前指纹识别技术还有很多困难，例如当三维的指纹被指纹录入设备扫描成二维的数字图像时，会丢失一部分信息、，手指划破、割伤、弄脏、不同干湿程度以及不同的按压方式，还会导致指纹图像的变化，这给可靠的特征提取带来了相当地困难；例如传统的基于细节点的识别方法，是依靠提取指纹脊线上的细节点，然后对其位置和类型进行匹配，来识别指纹的，而噪声会影响特征提取准确度，增加错误的特征点或丢失真正的特征点。当噪声很大时，就要增加图像增强算法来改善图像的质量，但很难找到一种增强算法能够适应所用的噪声，多种增强算法又会大幅增加算法运行时间，不好的增强算法又会增加人为特征。当噪声增大时，提取了许多虚假细节点，还有可能丢失细节点，这就是传统的基于细节点识别算法的不足之处之一，因为它只利用了指纹图像中的一小部分信息（细节点位置和方向）作为特征进行匹配，丢失了蕴涵在图像中的其他丰富的结构信息。不难想象，基于这种方法的识别算法，很难全面适应指纹的变化。

1.3 设计内容与要求

⑴、熟练掌握图像处理原理与模式识别原理；熟练掌握 MATLAB 软件及该软件中的 ImageProcessingToolbox 及其编程技巧；

⑵、掌握指纹识别的概念与实现过程框图；熟练掌握指纹图像的特征、特征提取、指纹识别方法；

⑶、构建指纹识别完整模型，包括图像获取、图形预处理、特征提取、图像识别各环节的软件算法；在图像预处理过程中，充分考虑图像去噪、图像增强等有关算法；同时，设计基于指纹识别的用户管理界面；

⑷、在消化吸收国内外研究成果的基础上，探讨指纹识别模型与算法的快速性、鲁棒性。同时，针对构建的简单指纹图像数据库具有较好的识别效果，并考虑指纹门禁控制系统的实时性；

⑸、参照国内外同行取得的研究成果，不断改进算法模型。针对实际应用，探讨该模型算法的优点；讨论指纹图像数据库的大小、系统容量、训练样本随机变化、以及选择不同算法时对识别率的影响；得出具有一般性指导意义的结论。

第 2 章 指纹识别的基本理论及应用

2.1 指纹识别的原理和方法

2.1.1 指纹的特征与分类

指纹识别学是一门古老的学科，它是基于人体指纹特征的相对稳定与唯一这一统计学结果发展起来的。实际应用中，根据需求的不同，可以将人体的指纹特征分为：永久性特征、非永久性特征和生命特征。永久性特征包括细节特征（中心点、三角点、端点、叉点、桥接点等）和辅助特征（纹型、纹密度、纹曲率等元素），在人的一生中永不会改变，在手指前端的典型区域中最为明显，分布也最均匀。细节特征是实现指纹精确比对的基础，而纹形特征、纹理特征等则是指纹分类及检索的重要依据。人类指纹的纹形特征根据其形态的不同通常可以分为“弓型、箕型、斗型”三大类型，以及“孤形、帐形、正箕形、反箕形、环形、螺形、囊形、双箕形和杂形”等 9 种形态。纹理特征则是由平均纹密度、纹密度分布、平均纹曲率、纹曲率分布等纹理参数构成。纹理特征多用于计算机指纹识别算法的多维分类及检索。非永久性特征由孤立点、短线、褶皱、疤痕以及由此造成的断点、叉点等元素构成的指纹特征，这类指纹有可能产生、愈合、发展甚至消失。

指纹的生命特征与被测对象的生命存在与否密切相关。但它与人体生命现象的关系和规律仍有待进一步认识。目前它已经成为现代民用指纹识别应用中越来越受关注的热点之一。

2.1.2 指纹识别的原理和方法

指纹识别技术主要涉及四个功能：读取指纹图像、提取特征、保存数据和比对。通过指纹读取设备读取到人体指纹的图像，然后要对原始图像进行初步的处理，使之更清晰，再通过指纹辨识软件建立指纹的特征数据。软件从指纹上找到被称为“节点”（minutiae）的数据点，即指纹纹路的分叉、终止或打圈处的坐标位置，这些点同时具有七种以上的唯一性特征。通常手指上平均具有 70 个节点，所以这种方法会产生大约 490 个数据。这些数据，通常称为模板。通过计算机模糊比较的方法，把两个指纹的模板进行比较，计算出它们的相似程度，最终得到两个指纹的匹配结果。采集设备（即取像设备）分成几类：光学、半导体传感器和其他。

2.2 指纹识别技术的主要指标和测试方法

2.2.1 算法的精确度

指纹识别系统性能指标在很大程度上取决于所采用算法性能。为了便于采用量化的方法表示其性能，引入了下列两个指标：

拒识率（false rejection rate，FRR）：是指将相同的指纹误认为是不同的，而加以拒绝的出错概率。FRR=（拒识的指纹数目 / 考察的指纹总数目）×100%。

误识率（false accept rate，FAR）：是指将不同的指纹误认为是相同的指纹，而加以接收的出错概率。FAR=（错判的指纹数目 / 考察的指纹总数目）×100%。

对于一个已有的系统而言，通过设定不同的系统阈值，就可以看出这两个指标是互为相关的，FRR 与 FAR 成反比关系。这很容易理解，“把关”越严，误识的可能性就越低，但是拒识的可能性就越高。

2.2.2 误识率和拒识率的测试方法

测试这两个指标，通常采用循环测试方法。即给定一组图像，然后依次两两组合，提交进行比对，统计总的提交比对的次数以及发生错误的次数，并计算出出错的比例，就是 FRR 和 FAR。针对 FAR=0.0001%的指标，应采用不少于 1415 幅不同的指纹图像作循环测试，总测试次数为 1000405 次，如果测试中发生一次错误比对成功，则 FAR=1/1000405；针对 FRR=0.1%，应采用不少于 46 幅属于同一指纹的图像组合配对进行测试，则总提交测试的次数为 1035 次数，如果发生一次错误拒绝，则 FRR=1/1035。测试所采用的样本数越多，结果越准确。作为测试样本的指纹图像应满足可登记的条件。

2.2.3 系统参数

登率（error registration rate，ERR）：指的是指纹设备出现不能登录及处理的指纹的概率，ERR 过高将会严重影响设备的使用范围，通常要求小于 1%。

登录时间：指纹设备登录一枚指纹所需的时间，通常单次登录的时间要求不超过 2 s。

比对时间：指纹设备对两组指纹特征模版进行比对所耗费的时间，通常要求不超过 1 s。

工作温度：指纹设备正常工作时所允许的温度变化范围，一般是 0～40 ℃。

工作湿度：指纹设备正常工作时所允许的相对湿度变化范围，一般是 30%～95%。

2.3 指纹识别技术的应用

指纹识别技术已经成熟，其应用日益普遍，除了刑事侦察用之外，在民用方面已非常广泛，如指纹门禁系统、指纹考勤系统、银行指纹储蓄系统、银行指纹保管箱、指纹医疗

保险系统、计划生育指纹管理系统、幼儿接送指纹管理系统、指纹献血管理系统、证券交易指纹系统、指纹枪械管理系统、智能建筑指纹门禁管理系统、驾驶员指纹管理系统等。指纹门禁系统和指纹考勤系统是开发和使用得最早的一种出入管理系统，包括对讲指纹门禁、联机指纹门禁、脱机指纹门禁等等。在入口将个人的手指按在指纹采集器上，系统将已登录在指纹库中的指纹（称为已经注册）进行对比，如果两者相符（即匹配），则显示比对成功，门就自动打开。如不匹配，则显示“不成功”或“没有这个指纹”，门就不开。在指纹门禁系统中，可以是一对一的比对（one to one matching），也可以是一对几个比对（one to few matching）。前者可以是一个公司、部门，后者可以是一个家庭的成员、银行的营业厅、金库、财务部门、仓库等机要场所。在这些应用中，指纹识别系统将取代或者补充许多大量使用照片和 ID 系统。

把指纹识别技术同 IC 卡结合起来，是目前最有前景的一个应用之一。该技术把卡的主人的指纹（加密后）存储在 IC 卡上，并在 IC 卡的读卡机上加装指纹识别系统，当读卡机阅读卡上的信息时，一并读入持卡者的指纹，通过比对就可以确认持卡者是否是卡的真正主人，从而进行下一步的交易。指纹 IC 卡可取代现行的 ATM 卡、制造防伪证件等。ATM 卡持卡人可不用密码，避免老人和孩子记忆密码的困难。

近年来，互联网带给人们方便与利益已，也存在着安全问题。指纹特征数据可以通过电子邮件或其它传输方法在计算机网络上进行传输和验证，通过指纹识别技术，限定只有指定的人才能访问相关的信息，可以极大地提高网上信息的安全性。网上银行、网上贸易、电子商务等一系列网络商业行为就有了安全性保障。

指纹社会保险系统的应用为养老金的准确发放起了非常有效的作用。避免了他人用图章或身份证复印件代领，而发放人员无法确定该人是故世的问题，要凭本人的活体指纹，才可准确发放养老金。

2.4 指纹识别的可靠性

指纹识别技术是成熟的生物识别技术。因为每个人包括指纹在内的皮肤纹路在图案、断点和交叉点上各不相同，是唯一的，并且终生不变。通过他的指纹和预先保存的指纹进行比较，就可以验证他的真实身份。自动指纹识别是利用计算机来进行指纹识别的一种方法。它得益于现代电子集成制造技术和快速而可靠的算法理论研究。尽管指纹只是人体皮肤的一小部分，但用于识别的数据量相当大，对这些数据进行比对是需要进行大量运算的模糊匹配算法。利用现代电子集成制造技术生产的小型指纹图像读取设备和速度更快的计算机，提供了在微机上进行指纹比对运算的可能。另外，匹配算法可靠性也不断提高。因此，指纹识别技术已经非常简单实用。由于计算机处理指纹时，只是涉及了一些有限的信息，而且比对算法并不是十分精确匹配，其结果也不能保证 100%准确。

指纹识别系统的特定应用的重要衡量标志是识别率。主要包括拒识率和误识率，两者成反比关系。根据不同的用途来调整这两个值。尽管指纹识别系统存在着可靠性问题，但其安全性也比相同可靠性级别的“用户 ID+密码”方案的安全性要高得多。拒识率实际上

也是系统易用性的重要指标。在应用系统的设计中，要权衡易用性和安全性。通常用比对两个或更多的指纹来达到不损失易用性的同时，极大提高系统的安全性。

2.5 本章小结

本章详细介绍了指纹识别的基本理论，是我们对指纹识别原理及处理方法有了初步的了解。指纹识别系统性能指标在很大程度上取决于所采用算法性能。指纹识别技术已经成熟，其应用日益普遍，除了刑事侦察用之外，在民用方面已非常广泛，如指纹门禁系统、指纹考勤系统、银行指纹储蓄系统、银行指纹保管箱、指纹医疗保险系统、计划生育指纹管理系统、幼儿接送指纹管理系统、指纹献血管理系统、证券交易指纹系统、指纹枪械管理系统、智能建筑指纹门禁管理系统、驾驶员指纹管理系统等。指纹识别技术是成熟的生物识别技术，指纹识别系统的特定应用的重要衡量标志是识别率。

第 3 章 指纹图像处理及特征提取与实现

3.1 方法概述

基于细节点特征的指纹自动识别技术是目前这方面研究中的主流，这种系统的实现有以下一些步骤如图 3-1：

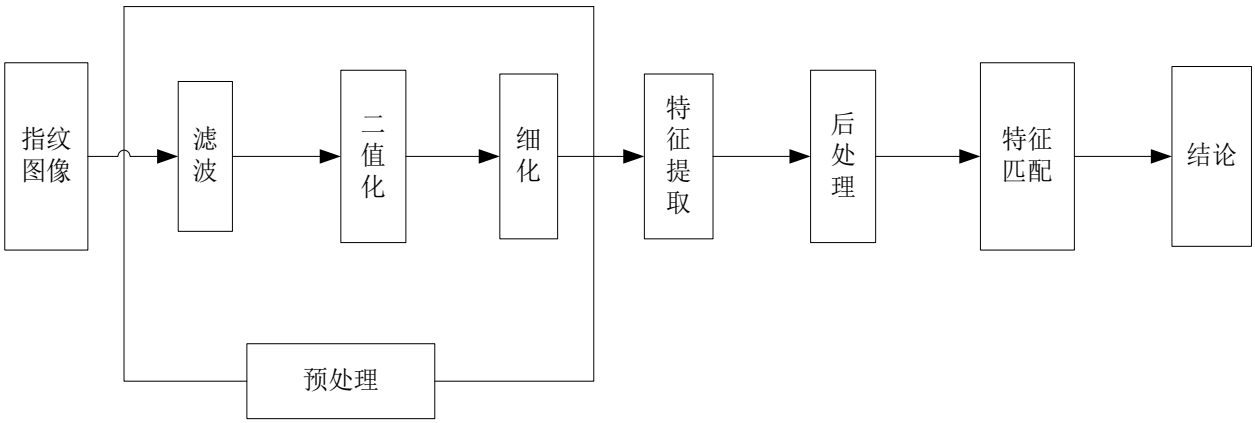


图 3-1　基于细节点特征的指纹自动识别系统

预处理是将输入的（直接采集进来的）低质量、有噪音的指纹源图象处理成已细化了的清晰的二值图像[3]。它的目的是减少低质量的图像对分类识别结果的影响，预处理中一般包括图像增强、滤波、二值化、细化等步骤。**预处理的方法通常有两种：**

方法一：先求方向图，后求频率图，最后由此得到的 Gabor 滤波器对图像进行滤波。这种方法计算量比较大，在求频率图容易产生偏差，不利于单片机的实现。

方法二：结合指纹图像自身的特点以及其源图像像素来确定该点是否为脊，直接准确地得到黑白二值的指纹脊图像。这 种方法对于从不同渠道获得的图像均有不错的效果。

在以上两种方法都要用到方向图，方向图是一种可直接从原灰度图像中得到的有用信息，在预处理、特征提取、指纹分类中有着重要意义。我们总是在准确求得方向图的基础上运用各种滤波方法或直接找脊的方法来进行预处理。

方向图描述了指纹图像中每一像素点所在脊线或谷线在该点的切线方向，也可看作是指纹源图像的一种变化表示方法，既用纹线的方向来表示该纹线。方向图分为两种：一种是点方向图，表示源指纹图像中每一点脊线的方向：另一种是块方向图，表示源指纹图像中每一块脊线的大致方向。计算方向图的基本思想是：在原灰度图像中每一点（或每一块在各个方向上的某个统计量（如灰度差、梯度等），根据这些统计量在各个方向上的差异，确定该点（块）的方向。

3.2 方向图的计算

3.2.1 求点方向图

设 $f(x,y)$ 是指纹图像中点 (x,y) 的灰度值，要计算该点的方向 $D(x,y)$，需要先求出 S_d （该点临域沿 d 方向的灰度变化）。

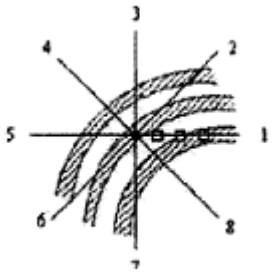


图 3-2 点方向示意图

$$S_d=\sum_{k=1}^{n}\left|f(x,y)-f_d(x_k,y_k)\right|,\quad \text{d=1，2，…，N} \tag{3.1}$$

其中：(x_k,y_k) 是方向 d 上的第 k 个点；

$f_d(x_k,y_k)$ 是该点的灰度值，N 是所取的方向数，n 为每个方向上所取的邻点数。这两个数的具体取值与图像的分辨率有关，一般取 N=16，即取 16 个方向，n=8，即一个方向上取 8 个邻点。

点 (x,y) 的方向为 S_d 取值最小的方向。对图像中的每一点求取点方向，这样便形成了

指纹点方向图。

此方向求得的方向特点：

1．方向取值不是 0～2π 中的任意值，而是有限的几个数。

2．这种方向计算出的方向范围是 0～2π，有利于求取指纹的走势。（$\pi/4$ 与 $5\pi/4$ 认为是不同方向）

3.2.2 由点方向图求块方向图的算法

把点方向图分成 $w\times w$ 大小的块，对每一块计算方向直方图（横坐标的方向取到的 N 个值，纵坐标为取这些方向的象素个数），方向直方图中的峰值所对应的方向，即该块的方向。

3.2.3 最小均方估计块方向算法

$f(i,j)$ 代表指纹图像在 (i,j) 处的灰度值步骤：

1．将图像分成大小为 $M\times M$ 的块。这里 M 的大小以包含一脊一谷（即一周期）为宜；

2．计算块中每个像素 $f(i,j)$ 在 x 轴和 y 轴上的梯度 $G_x(i,j)$ 和 $G_y(i,j)$。

这里用简单的梯度算子[4]：

$$\begin{aligned} G_x(i,j)=&\left[f(i-1,j-1)+2f(i-1,j)+f(i-1,j+1)\right]\\ &-\left[f(i+1,j-1)+2f(i+1,j)+f(i+1,j+1)\right] \end{aligned} \tag{3.2}$$

$$\begin{aligned} G_y(i,j)=&\left[f(i-1,j-1)+2f(i,j-1)+f(i+1,j-1)\right]\\ &-\left[f(i-1,j+1)+2f(i,j+1)+f(i+1,j+1)\right] \end{aligned} \tag{3.3}$$

3．用下面公式计算 $M\times M$ 块的方向 θ，

$$\theta=\frac{1}{2}\tan^{-1}\left(\frac{\sum_{i=0}^{M-1}\sum_{j=0}^{M-1}2G_x(i,j)G_y(i,j)}{\sum_{i=0}^{M-1}\sum_{j=0}^{M-1}\left(G_x^2(i,j)-G_y^2(i,j)\right)}\right)\qquad G_x\neq 0 \text{ 且 } G_y\neq 0 \tag{3.4}$$

当 $M\times M$ 的块中 G_x 或 G_y 为零的比率很大时，块方向应直接设为 0 或 π，这样 $M\times M$ 块中的每一个像素的方向都等于 θ。

（A）原始图像　　（B）改进前的结果　　（C）改进后的结果

图 3-3 指纹图像的方向图

4．在计算指纹图像方向图多采用块与块直接不重叠的方式，但这样求出的块方向仅由该块所包含的像素点所决定，没有考虑其周围像素点的影响，很容易造成方向的不连续性。于是我们采用在计算方向图时，采用块重叠的方式。这样可以使方向图精度可以大大提高。

5．方向图平滑

这里我们采用将方向图正交分解后分别进行滤波的方法对它进行平滑，实验证明这种方法比一般用的中值滤波和加权平滑的方法的效果好。

计算步骤：

将方向图转变为连续的向量场：

$$\Phi_x(i,j)=\cos(2O(i,j)) \tag{3.5}$$

$$\Phi_y(i,j)=\sin(2O(i,j)) \tag{3.6}$$

这里Φ_x，Φ_y分别是向量场 x，y 方向上的分量。

实现低通滤波：

$$\Phi'_x(i,j)=\sum_{u=-w_\phi/2}^{u=w_\phi/2}\sum_{v=-w_\phi/2}^{v=w_\phi/2}W(u,v)\cdot\Phi_x(i-uw,j-vw) \tag{3.7}$$

$$\Phi_y{}'(i,j)=\sum_{u=-w_\phi/2}^{u=w_\phi/2}\sum_{v=-w_\phi/2}^{v=w_\phi/2}W(u,v)\cdot\Phi_y(i-uw,j-vw) \tag{3.8}$$

这里 W 是一个大小为$w_\phi\times w_\phi$的二维低通滤波器[5]，一般使用5×5的均值滤波。

平滑后的方向场为：

$$O'(i,j)=\frac{1}{2}\tan^{-1}\left(\frac{\Phi_y{}'(i,j)}{\Phi_x{}'(i,j)}\right) \tag{3.9}$$

3.3 指纹图像的滤波

在指纹处理中用到滤波器，主要在于去除图像噪声，增强图像质量，即增强指纹脊与谷的对比度，修补图像——连接脊中出现的断点、去除图像中的叉连现象。

指纹图像滤波有两种方法：

方法一：利用 Gabor 滤波器的参数可利用指纹的方向性和纹理性，用 Gabor 滤波器来作为带通滤波器，去除噪音，增强脊谷结构。这种算法难点在于需要计算图像的频率图——将指纹图像看成由脊和谷组成的周期图像，在每一个局部领域内都会有一个相对固定准确的频率。这种算法的缺点在于：求频率图导致计算量比较大，而且频率容易产生偏差；

方法二：这种方法也是我们采用的滤波方法。这是一种简单但效果良好的上下文滤波器，上下文滤波法也是基于方向图的，这是一系列上下文相关的滤波器，使用时根据某一快的方向从一系列滤波器中选择一个相应的滤波器来对这一块进行滤波。其他方向的滤波器可以通过旋转得到。

一个基本的滤波器由两部分组成：平均滤波器和分离滤波器。平均滤波器的作用主要是连接边中出现的断点，而分离滤波器可以去除图像中的叉连现象。(断裂和叉连的情况；如右图 3-4 所示)。

滤波器的大小由指纹图像中脊线的周期决定，在我们的试验中，周期取为 5，因而这里以 Sx5 大小的滤波器为例。水平方向的平均滤波器的权值如右图 3-5 所示，其中系数满足：$X > Y > Z \geq 0$。

经过平均滤波器过滤的图像，其中每一点的灰度由其临近的 24 个像素的灰度值共同决定。即对于第 i 行 j 列的点的灰度值 $f(i,j)$的处理如下：

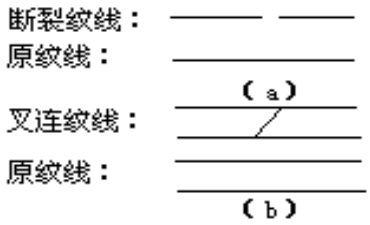


图 3-4 断裂与叉连

Z	Z	Z	Z	Z
Y	Y	Y	Y	Y
X	X	X	X	X
Y	Y	Y	Y	Y
Z	Z	Z	Z	Z

图 3-5 平均滤波器的权值

$$f(i,j)=\sum_{m=-2}^{2} Z\times f(i-2,j+m)+\sum_{m=-2}^{2} Y\times f(i-1,j+m)+\sum_{m=-2}^{2} X\times f(i,j+m)$$
$$+\sum_{m=-2}^{2} Y\times f(i+1,j+m)+\sum_{m=-2}^{2} Z\times f(i+2,j+m) \qquad (3.10)$$

若图像中出现断点，即这一点的灰度值比周围点都小得多，则经过平均滤波器的处理，它的灰度值就接近邻近点的灰度值了，所以平均滤波器有连接断点的作用。 分离滤波器的权值如图 3-6 所示，其中参数：P+2Q+2R=0。分离滤波器处理图像过程与平均滤波器相

同，图像中的叉连点是把相邻的两条脊线连接起来的点，所以叉连点的上下点灰度值较大，而其同一行上的邻点的灰度值较小，通过分离滤波器的处理，叉连点的灰度值会明显降低，所以分离滤波器有去除叉连点的作用。

一个基本滤波器要求具备上述两种功能，它的作用相当于平均滤波器加分离滤波器，所以它的权值如右图 3-7，其中参数：K=X 十 P，L=Y+Q， M=Z+R。为归一化权值，基本滤波器的每个权值都需要除以该滤波器所有权值的总和。

R R R R R
Q Q Q Q Q
P P P P P
Q Q Q Q Q
R R R R R

图 3-6 分离滤波器的权值

M M M M M
L L L L L
K K K K K
L L L L L
M M M M M

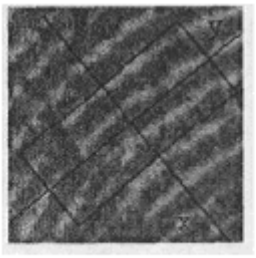
图 3-7 基本滤波器的权值

用方向滤波器进行滤波去噪时，根据块方向图中的该块的方向，选用相应的滤波器(将水平方向的滤波器旋转块方向的角度后得到)，进行滤波。

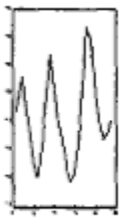
3.4 基于方向图的动态阀值指纹图像二值化方法

二值化是指纹图像预处理中必不可少的一步。常用的二值化方法有固定阀值法、自适应阀值法、局部自适应阀值法等，这些方法仅仅利用了图像的灰度信息，对指纹图像的二值化效果很不理想；现有的大部分指纹图像预处理方法都是经过滤波处理后再进行二值化[6]，这样就需要对图像进行两次扫描，不利于处理速度的提高。

将指纹图像自身的方向结构特点与源图像灰度值变化特点结合起来，确定对图像中每一像素点二值化的动态阀值。这种方法取代了一般指纹图像预处理中无效区域分割、滤波、增强、二值化等步骤，一次完成图像的二值化功能。实验结果也表明，该方法得到的二值化图像能够基本保持源图像上的特征点不丢失，确保了以后的特征提取和比对的正确性和可靠性。指纹局部图如图 3-8：



（A） 原始图像



（B）谷脊变化波形图

图 3-8 指纹谷脊变化波形图

方框 Y 方向是该块的指纹方向，X 方向是其法线方向。以法线方向上各像素点的灰度值做一曲线，可得到近似于正弦的波形图，如图 3-8(B)所示。显然，该波形图的波谷对应

指纹图像的脊线(指纹图像中暗的纹线)，而波峰则对应指纹图像的谷(指纹图像中亮的纹线)。若所考察的当前像素点恰好落在波谷上，则该点就是指纹脊线点，若所考察的当前像素点恰好落在波峰上，则该点就是指纹谷点，而谷点到脊点间像素灰度的变化几乎呈线性，正是基于指纹图像在结构上和像素灰度变化上的这些特点，本文提出了下述指纹图像二值化方法：

1．将图像分成大小为 N×N 的小块，用上述方向图改进方法计算各小块的方向 θ；

2．对图像中的每一点 (i,j)，以其所在块的方向 θ 作为该像素点的方向 $\theta(i,j)$，并以该点为中心在其法方向上取 $l\times w$（l,w 一般为奇数）的矩形窗，计算矩形窗内指纹方向每一列中像素点在法方向上的加权平均 $X[0],X[1],\cdots,X[w-1]$，具体公式为：

$$X[k]=\sum_{d=0}^{l-1} coefficent[d]\times f(i_{kd},j_{kd}) \qquad k=0,1\cdots w-1 \qquad (3.11)$$

其中 coefficent 为加权平均系数，满足 $\sum_{d=0}^{l-1} coefficent[d]=1$，$(i_{kd},j_{kd})$ 是 k 列上的第 d 个像素的位置。

3．对 $X[k](k=0\cdots w-1)$ 中极大点或极小点附近的波动做平滑处理；

4．找出 $X[k](k=0\cdots w-1)$ 极大点和极小点位置及对应的值，对极大值极小值求平均，将此平均值作为该点二值化门限，称其为动态阀值。

5．将当前像素点的加权平均灰度值（即 $X\left[\frac{l-1}{2}\right]$）与动态阀值进行比较，若小于动态阀值，同时，当前像素点在波形图极小点一个有限的邻域内，则当前像素点为脊线点，否则即为谷点，即：

$$f(i,j)=\begin{cases}0 & \left(X\left[\frac{l-1}{2}\right]<\text{动态阀值}\right)\bigcap\left(\left|\frac{l-1}{2}-\text{极小点位置}\right|\le\delta\right)\\ 255 & \text{其他}\end{cases} \qquad (3.12)$$

其中 δ 的取值与指纹读入器的分辨率有关，通常取为指纹周期的四分之一，本文取 $\delta=2$。

若 $X[k](k=0\cdots w-1)$ 的起伏很不明显，说明该区域属于无效区域或背景区域，整个区域的像素值置为 255。

本算法在具体实现中可利用同一块中所有像素有同样方向的条件，推导出快速算法，防止一些点的重复扫描，大大缩短处理时间。

3.5 指纹图像的细化算法

指纹图像二值化后，纹线仍具有一定的宽度，而指纹识别只对纹线的走向感兴趣，不关心它的粗细。为了进一步压缩数据，得到更精确的细节特征，提高识别的准确性，对指纹图像进行细化处理是不可忽略的。

所谓细化，就是从原来的图中去掉一些点，但仍要保持原有的形状。实际上，是保持原图的骨架[7]。

指纹图像的细化是指删除指纹纹线的边缘像素，使之只有一个像素宽度，细化时应保证纹线的连接性、方向性和特征点不变，还应保持纹线的中心基本不变。一种好的细化方法应满足下列条件：

1．迭代必须收敛的。(收敛性)

2．不破坏纹线的连接性。(连接性)

3．不引起纹线的逐步吞食。(拓扑性)

4．保护指纹的细节特征。(保持性)

5．骨架纹线的宽度为 1 个像素。(细化性)

6．骨架尽可能接近条纹中心线。(中轴性)

7. 算法简单、速度快。(快速性)

P_1	P_2	P_3	P_{13}
P_4	P_5	P_6	P_{14}
P_7	P_8	P_9	P_{15}
P_{10}	P_{11}	P_{12}	×

图 3-9　统一模块

下面介绍一种适合于指纹图像细化的算法，这种方法满足上面的细化条件，而且可以提出快速算法。

这种算法采用 4×4 模板，如图 3-9 所示，左上角的 3×3 方窗(即 P_1，P_2,…，P_9)为消除模板。即，一黑色象素八临域与消除模板中的一个相匹配的话，该点被认为是可消除的点。

消除模板具体如下：

0	0	0
×	1	×
1	1	1

（A）

0	×	1
0	1	1
0	×	1

（B）

1	1	1
×	1	×
0	0	0

（C）

1	×	0
1	1	0
1	×	0

（D）

×	0	0
1	1	0
×	1	×

（E）

0	0	×
0	1	1
×	1	×

（F）

×	1	×
0	1	1
0	0	×

（G）

×	1	×
1	1	0
×	0	0

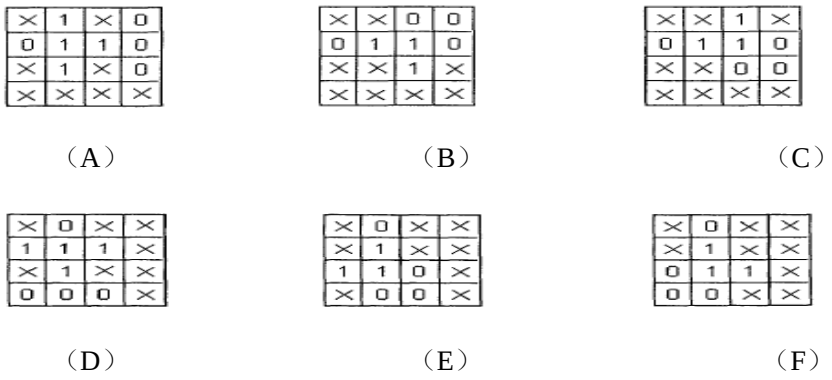
（H）

图 3-10　消除模块图

（A）~（D） 4 个模板能有效去除边缘上的突出物，保证了细化后的指纹骨架处于指纹脊线中心，避免了细化后的指纹骨架出现毛刺。

除消除模板外，还需构造保留模板，保持纹线的连通性。

保持模板内容如下图所示：



（A） （B） （C）

（D） （E） （F）

图 3-11 保留模块

具体算法如下：

从图像的左上角元素开始进行，每个像素(图中为P_5)均抽取出如图 3-9 所示的 15 个相邻像素，并且把其中 8 个邻域像素($P_1, P_2, P_3, \cdots, P_8$)与图 3-10 中的 8 个模板比较，如果和 8 个消除模板中的任意一个都不匹配时，保留；如果与 8 个中的任一个匹配，则抽取的元素再和上图中的 6 个保留模板进行比较，如果与其中任一个匹配的话，则P_5保留，否则P_5删去。重复这个过程，直到没有一个像素的值被改变。

这种算法的具体实现可引入查表法，即对所要处理的像素周围模板内前 15 个点预先进行编码(用双字节 15 个位来表示，双字节的最低位存放P_{15}的值，依此类推，次最高位存放P_{15}的值)，把编码值作为存储地址，按上述细化法则，预先在存储地址上存入相应的细化值(可用 0、1 表示，0 表示保留，1 表示删去)。细化时，移动 4x4 的方窗内象素的编码地址去查找相应的细化值，用细化值替换当前像素值。

3.6 特征提取及其后处理

3.6.1 特征点的提取

特征提取一般是指提取指纹图像的局部特征，也就是细节点特征。在基于细节点的指纹自动识别系统中，特征提取是在细化后的指纹图像上进行的。

特征提取的首要问题是确定细节点和它的位置，细节点的位置和细节点间的相对位置很重要，尽管每个指纹中包括将近 80 个细节，只要确定十几个细节点就已经足够用来识别了。

探测细节点的算法很简单，如图 3-12 所示的 3x3 模板就可以用来确定特征的位置。

M 是待检测的点，$X_1, X_2, \cdots X_8$ 是它的八邻域，沿顺时针方向排列。$R(1), R(2) \cdots, R(8)$ 是细化后图像在 $X_1, X_2, \cdots X_8$ 处的灰度。如果 M 是端点，则它的八邻域满足：

$$C_N = \sum_{k=1}^{8} \left| R(k+1) - R(k) \right| = 2, \qquad R(9) = R(1) \tag{3.13}$$

如果 M 是分叉点，则它的八邻域满足：

$$C_N = \sum_{k=1}^{8} \left| R(k+1) - R(k) \right| = 6, \qquad R(9) = R(1) \tag{3.14}$$

这样我们就可以在细化后的图像中找到细节点(端点和分叉点)，并记录它们在图中的相对位置。

3.6.2 假特征点的去除

这样得到的特征中存在由指纹质量、摄入噪声等原因造成的很多假特征，如下图 3-12 所示(a)和(b)中产生了假的端点；(c)和(d)中形成了错误的断开和连接；(e)中显示的是一个由不平滑的脊引起的毛刺，出现了假端点和假分叉两种特征：(f)~(g)是几种错误连接的例子，分别称为桥形、三角形、梯形结构。

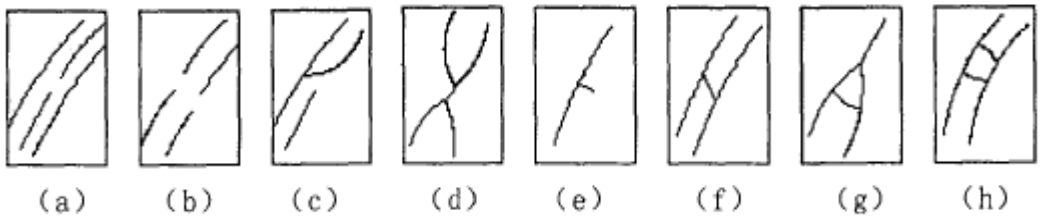


图 3-12　假特征

针对上面这些假特征，我们采用了非常简单的方法：计算特征点间的距离，如果距离值小于所给的阀值，则认为特征点为假特征点。这种算法实现起来也很简单：扫描特征点的某一邻域，如果该邻域中出现其他特征点的话，将该特征点与出现的特征点同时去除，如此循环几次，当没有特征点被删去时，处理完成。

这种方法中邻域的选取很重要，如果取得比较小，则可能起不到去除假特征点的作用；如果取得比较大的话，则可能将真正的特征点也一并删去。在具体实现中，我们取其半径为脊宽的一半。

3.6.3 细节点信息的提取及记录

对每一个细节点，我们记录如下信息：

1．细节点的 x,y 坐标。

2．细节点的方向，这个方向就是该细节点所在的块的块方向。

3．细节点的类型，即脊线端点或脊线分叉点。

4．细节点对应的脊线(d_i, a_i)。

细节点对应的脊线用该脊线上的采样点来表示，采样的距离约为脊线间的平均距离。分叉点对应的脊线是与该细节点的方向最近的那条，端点对应的脊线就是该细节点所在的脊线。采样点用该点与对应细节点的距离d_j，和连接该点与对应细节点的直线与对应细节点方向的夹角a_j来表示，a_j的取值范围在-180 到 180 度之间。下图给出了细节点对应的脊线与脊线上的采样点的例子。在细节匹配中，对应脊线将被用来对两个平面点集进行校准，而且，校准的参数，也就是两个点集中任意一对脊线间的旋转角度，将被用来作为判断它们所对应的细节点能否看作匹配的细节点的条件。

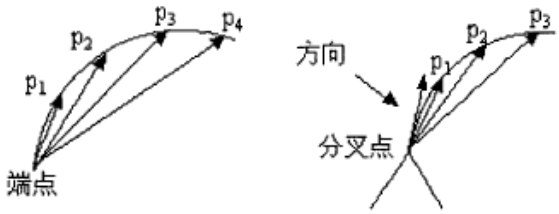


图 3-13　细节点的对应脊线

3.6.4　指纹识别中细节点的匹配

细节匹配一般在极坐标系中进行，因为指纹图像的非线性形变往往呈放射状，在某个区域内的形变比较大，然后非线性地向外扩张，因而，在极坐标中能更好地描述非线性形变；另外，在极坐标中不需要考虑输入图像与模板图像的参照点之间的平移，将一对对应点的坐标相对于参照点转换为极坐标时，平移就被抵消了；还有，在极坐标系中显然比在直角坐标系中更便于处理两幅图像间的旋转[8]。

细节点集的校准：

$$令 \quad P=\left(\left(x_1^P, y_1^P, \theta_1^P\right)^T, \cdots, \left(x_M^P, y_M^P, \theta_M^P\right)^T\right) \tag{3.15}$$

表示模板图像中的 M 个细节点，

$$Q=\left(\left(x_1^Q, y_1^Q, \theta_1^Q\right)^T, \cdots, \left(x_N^Q, y_N^Q, \theta_N^Q\right)^T\right) \tag{3.16}$$

表示输入图像中的 N 个细节点。

为了把细节点转移到极坐标系中去，需要在模板细节点集和输入细节点集中各选一个参照点作为相应的极坐标系中的原点，并算出其它细节点相对于参照点的极坐标。由于事先不知道模板点集与输入点集的对应关系，需要考虑所有可能的参照点对。

对模板点集中的每一点 $P_i(1 \le i \le M)$ 和输入点集中的每一点 $Q_j(1 \le j \le N)$，定义 $\mathrm{rotate}[i][j]$ 为将 P_i 和 Q_j 当作参照点对时，从输入图像到模板图像的旋转角度。如果，P_i 和 Q_j 可以被当作一对对应点，即它们分别对应的脊线相似性到了一定程度，则 $\mathrm{rotate}[i][j]$ 将取 0 度到 360 度间的一个值，否则，我们定义 $\mathrm{rotate}[i][j]$ 取值为 400，以表示 P_i 和 Q_j 不能是一对对应点。如果 P_i 和 Q_j 是不同类型的细节点，也就是说它们一个是端点，一个是分叉点，则它们不是对应点对，$\mathrm{rotate}[i][j]$ 取值为 400。即，$\mathrm{rotate}[i][j] < 400$ 表示 P_i 和 Q_j 对应的脊线相似性到了一定程度。

3.7 本章小结

本章详细介绍了指纹识别过程的原理及算法的研究，具体算法将在后续章节给出。

第 4 章 指纹识别算法的 MATLAB 实现

4.1 MATLAB 简介

4.1.1 MATLAB 的概况

MATLAB 是矩阵实验室（Matrix Laboratory）之意。除具备卓越的数值计算能力外，它还提供了专业水平的符号计算、文字处理、可视化建模仿真和实时控制等功能。

MATLAB 的基本数据单位是矩阵，它的指令表达式与数学，工程中常用的形式十分相似，故用 MATLAB 来解算问题要比用 C、FORTRAN 等语言完成相同的事情简捷得多。当前流行的 MATLAB 包括拥有数百个内部函数的主包和三十几种工具包(Toolbox)。工具包又可以分为功能性工具包和学科工具包。功能工具包用来扩充 MATLAB 的符号计算、可视化建模仿真、文字处理及实时控制等功能。学科工具包是专业性比较强的工具包，控制工具包、信号处理工具包、通信工具包等都属于此类。开放性使 MATLAB 广受用户欢迎。除内部函数外，所有 MATLAB 主包文件和各种工具包都是可读可修改的文件，用户通过对源程序的修改或加入自己编写程序构造新的专用工具包。

4.1.2 MATLAB 产生的历史背景

在 70 年代中期, Cleve Moler 博士和其同事在美国国家科学基金的资助下开发了调用 EISPACK 和 LINPACK 的 FORTRAN 程序库。EISPACK 是特征值求解的 FOETRAN 程序库，LINPACK 是解线性方程的程序库。在当时，这两个程序库代表矩阵运算的最高水平。到 70 年代后期，身为美国 New Mexico 大学计算机系系主任的 Cleve Moler 在给学生讲授线性代数课程时，想教学生使用 EISPACK 和 LINPACK 程序库，但他发现学生用 FORTRAN 编写接口程序很费时间，于是他开始自己动手，利用业余时间为学生编写 EISPACK 和 LINPACK 的接口程序。Cleve Moler 给这个接口程序取名为 MATLAB, 该名为矩阵(matrix)和实验室(labotatory)两个英文单词的前三个字母的组合。在以后的数年里, MATLAB 在多所大学里作为教学辅助软件使用, 并作为面向大众的免费软件广为流传。1983 年春天, Cleve Moler 到 Standford 大学讲学, MATLAB 深深地吸引了工程师 John Little.John Little 敏锐地觉察到 MATLAB 在工程领域的广阔前景。同年, 他和 Cleve Moler, Steve Bangert 一起，用 C 语言开发了第二代专业版。这一代的 MATLAB 语言同时具备了数值计算和数据图示化的功能。1984 年，Cleve Moler 和 John Little 成立了 Math Works 公司，正式把 MATLAB 推向市场，并继续进行 MATLAB 的研究和开发。

在当今 30 多个数学类科技应用软件中，就软件数学处理的原始内核而言，可分为两大类。一类是数值计算型软件，如 MATLAB,Xmath,Gauss 等，这类软件长于数值计算，

对处理大批数据效率高；另一类是数学分析型软件，Mathematical,Maple 等,这类软件以符号计算见长,能给出解析解和任意精确解,其缺点是处理大量数据时效率较低。Math Works 公司顺应多功能需求之潮流,在其卓越数值计算和图示能力的基础上,又率先在专业水平上开拓了其符号计算、文字处理、可视化建模和实时控制能力,开发了适合多学科,多部门要求的新一代科技应用软件 MATLAB。经过多年的国际竞争,MATLAB 以经占据了数值软件市场的主导地位。在MATLAB进入市场前,国际上的许多软件包都是直接以 FORTRAN、C 语言等编程语言开发的。这种软件的缺点是使用面窄,接口简陋,程序结构不开放以及没有标准的基库,很难适应各学科的最新发展,因而很难推广。MATLAB 的出现,为各国科学家开发学科软件提供了新的基础。在 MATLAB 问世不久的 80 年代中期,原先控制领域里的一些软件包纷纷被淘汰或在 MATLAB 上重建。

Math Works 公司 1993 年推出了 MATLAB 4.0 版,1995 年推出 4.2C 版（for win3.X）,1997 年推出 5.0 版,1999 年推出 5.3 版。MATLAB 5.X 较 MATLAB 4.X 无论是界面还是内容都有长足的进展,其帮助信息采用超文本格式和 PDF 格式,在 Netscape 3.0 或 IE 4.0 及以上版本,Acrobat Reader 中可以方便地浏览。时至今日,经过 Math Works 公司的不断完善,MATLAB 已经发展成为适合多学科,多种工作平台的功能强大大型软件。在国外,MATLAB 已经经受了多年考验。在欧美等高校,MATLAB 已经成为线性代数、自动控制理论、数理统计、数字信号处理、时间序列分析、动态系统仿真等高级课程的基本教学工具；成为攻读学位的大学生、硕士生、博士生必须掌握的基本技能。在设计研究单位和工业部门,MATLAB 被广泛用于科学研究和解决各种具体问题。在国内,特别是工程界,MATLAB 一定会盛行起来。可以说，无论你从事工程方面的哪个学科，都能在 MATLAB 里找到合适的功能。

4.1.3 MATLAB 的语言特点

一种语言之所以能如此迅速地普及，显示出如此旺盛的生命力，是由于它有着不同于其他语言的特点，正如同 FORTRAN 和 C 等高级语言使人们摆脱了需要直接对计算机硬件资源进行操作一样，被称作为第四代计算机语言的 MATLAB，利用其丰富的函数资源，使编程人员从繁琐的程序代码中解放出来。MATLAB 最突出的特点就是简洁。MATLAB 用更直观的，符合人们思维习惯的代码，代替了 C 和 FORTRAN 语言的冗长代码。MATLAB 给用户带来的是最直观、最简洁的程序开发环境。以下简单介绍一下 MATLAB 的主要特点:

1)、语言简洁紧凑，使用方便灵活，库函数极其丰富。MATLAB 程序书写形式自由，利用起丰富的库函数避开繁杂的子程序编程任务，压缩了一切不必要的编程工作。由于库函数都由本领域的专家编写，用户不必担心函数的可靠性。可以说，用 MATLAB 进行科技开发是站在专家的肩膀上。

具有 FORTRAN 和 C 等高级语言知识的读者可能已经注意到，如果用 FORTRAN 或 C 语言去编写程序，尤其当涉及矩阵运算和画图时，编程会很麻烦。例如，如果用户想求解一个线性代数方程，就得编写一个程序块读入数据，然后再使用一种求解线性方程的算法

（例如追赶法）编写一个程序块来求解方程，最后再输出计算结果。在求解过程中，最麻烦的要算第二部分。解线性方程的麻烦在于要对矩阵的元素作循环，选择稳定的算法以及代码的调试动不容易。即使有部分源代码，用户也会感到麻烦，且不能保证运算的稳定性。解线性方程的程序用 FORTRAN 和 C 这样的高级语言编写，至少需要四百多行，调试这种几百行的计算程序可以说很困难。MATLAB 的程序极其简短，更为难能可贵的是，MATLAB 甚至具有一定的智能水平，用户根本不用怀疑 MATLAB 的准确性。

2)、运算符丰富。由于 MATLAB 是用 C 语言编写的，MATLAB 提供了和 C 语言几乎一样多的运算符，灵活使用 MATLAB 的运算符将使程序变得极为简短。

3)、 MATLAB 既具有结构化的控制语句（如 for 循环，while 循环，break 语句和 if 语句），**又有面向对象编程的特性。**

4)、程序限制不严格，程序设计自由度大。例如，在 MATLAB 里，用户无需对矩阵预定义就可使用。

5)、程序的可移植性很好。基本上不做修改就可以在各种型号的计算机和操作系统上运行。

6)、MATLAB 的图形功能强大。在 FORTRAN 和 C 语言里，绘图都很不容易，但在 MATLAB 里，数据的可视化非常简单。MATLAB 还具有较强的编辑图形界面的能力。

7)、MATLAB 的缺点是，它和其他高级程序相比，程序的执行速度较慢。由于 MATLAB 的程序不用编译等预处理，也不生成可执行文件，程序为解释执行，所以速度较慢。

8)、功能强大的工具箱是 MATLAB 的另一特色。MATLAB 包含两个部分：核心部分和各种可选的工具箱。核心部分中有数百个核心内部函数。其工具箱又分为两类：功能性工具箱和学科性工具箱。功能性工具箱主要用来扩充其符号计算功能，图示建模仿真功能，文字处理功能以及与硬件实时交互功能。功能性工具箱用于多种学科。而学科性工具箱是专业性比较强的，如 control,toolbox,signl proceessing toolbox,commumnication toolbox 等。这些工具箱都是由该领域内学术水平很高的专家编写的，所以用户无需编写自己学科范围内的基础程序，而直接进行高、精、尖的研究。

9)、源程序的开放性。开放性也许是 MATLAB 最受人们欢迎的特点。除内部函数以外，所有 MATLAB 的核心文件和工具箱文件都是可读可改的源文件，用户可通过对源文件的修改以及加入自己的文件构成新的工具箱。

4.1.4 MATLAB 在指纹识别中的应用

MATLAB 语言是一种优秀的计算机语言，具有数学运算能力是它的突出优点之一。许多在 C 语言中或其它高级语言中很复杂的编程问题在 MATLAB 语言编程中只需要一条专用指令就就可以完成。MATLAB 语言的所有计算都基于矩阵运算来完成，所以，MATLAB 中的所有变量都定义为矩阵，所有的运算都是关于矩阵的运算。它是一种解释型语言，几乎没有格式上的限制。基于 MATLAB 语言描述简单并且图形显示功能比较强大的特点，以它作为指纹图像识别算法仿真的平台。在指纹图像处理算法中所处理的是一个个像素点

的灰度值，可以在 MATLAB 中将图形文件转变成可以在程序中处理的数据形式。

MATLAB 具有专门的图像的读取和显示函数，相对于其它语言而言要方便的多。下面是一个基本的灰度图像的读取和显示方式：

[X,Cmap]=imread(“d：\fingerprint\CASIA_DBI\1_1.bmp);%读取 bmp 格式文件

Cmap;%观察色图矩阵

Imagesc（X）;%显示灰度图像

Colormap（gray）;%借用 MATLAB 的灰度矩阵

X 是程序内一个存储图片灰度值的矩阵，矩阵内的每一个元素的值都对应一个相应的像素的灰度值。对图像的处理可以通过操作矩阵来表达。进行处理完毕后，再用相应的命令将矩阵以图片的形式输出，而进行观察。

MATLAB 语言的变量不用定义，而直接赋值。经过这个命令输入进来的矩阵是一个整形变量，但是程序内处理图形时使用的是双精度浮点数，所以要转换一下数据类型[9][10]。

X1=double（X）;

命令中 X1 代表的是转换之后的双精度数据类型，X 代表指纹图像中像素灰度值输入的数组，这是一个整形矩阵。经过这个命令，整型变量就可以强制转换成双精度变量。

4.2 指纹图像预处理

4.2.1 图像规格化

受采集设备参数和环境的影响，采集到的指纹图像可能总体对比度较差。图像归一化的作用就是使所有指纹图像具有相同的灰度均值和方差，从而将每一幅图像的灰度调整到统一的范围，方便后续处理。归一化的算法是：

（1） 先计算图像的平均值和方差。这部分工作主要通过统计图像中各点像素值得到该图像的直方图，然后利用直方图来计算指纹图像的相关指标。

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

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

（2） 指定期望经过处理后的图像均值和方差，计算归一化后的图像 G。

$$G(i,j)=\begin{cases}M_0+\sqrt{\dfrac{Var_0(I(i,j)-M(I))^2}{Var}}, I(i,j)>M\\ M_0-\sqrt{\dfrac{Var_0(I(i,j)-M(I))^2}{Var}}, others\end{cases}$$

程序见附录 1。

4.2.2 图像分割

指纹图像通常包括纹线区域和周边的无用区域。如果保留这些无用区域，会致使计算冗余度增加。为此要先将其从待处理区域中去除。通常用的分离方法有像素领域特征的方法和基于像素分布概率的方法等。本文采取对各像素邻域特征进行分析，然后分割图像算法。

吧图像分成 T*T 的非重叠块，计算每一块的灰度均值 M 和方差 Var，通过下列条件区分前景与背景：

（1） 如果 M>M1 且 Var<Var1，则认为背景

（2） 如果 M<=M1 且 Var>=Var1，则认为前景，否则待定；

（3） 对所有分块划分完毕后，在对待定块进行判决。如果在 8 邻域中，背景景小于等于 4 则认为是前景，否则是背景

在编程过程中通过几次调试后对图像进行 3*3 分块处理

程序见附录 1。

4.2.3 图像二值化

一般的指纹图像都有比较清晰的方向场，方向场估计得准确性直接决定了图像增强算法的效果。

为估计方向场，我们把指纹脊线的走向分为如下 8 个方向，如下图：

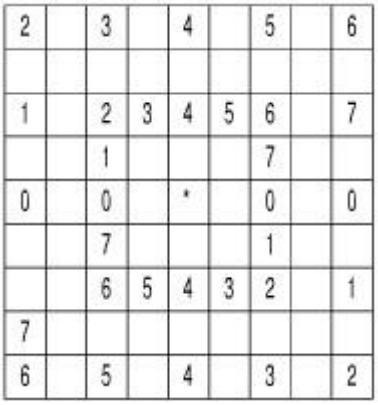


图 4-1 在一个像素处的 8 个指纹脊线方向

我们先对分割后的图像进行了平均滤波，然后对图像的每一个像素，为确定在该像素出的脊线方向，在以该像素为中心的 9*9 窗口内，分别计算 8 个方向上的经过处理后的灰度值，即将图 4-1 中标了 i（i=0,1,…7 分别代表 8 个方向）的位置的像素灰度值去他们中最大 summax 和最小值 summin，若满足(summax+summin+ 4*I(x,y))> (3*summ/8), 则该像素点的脊线方向为 summin, 否则为 summax.

确定完方向后就根据该向场对图像进行二值化。

程序见附录 1。

4.2.4 图像增强

在当前的指纹采集条件下，不可避免的会受到环境，皮肤上的油脂、水分、污渍的影响，使采集到的指纹图像出现纹线粘连、纹线断裂等缺陷，对后续的指纹特征提取带来很大困难。所以我们必须对指纹图像进行图像增强处理，例如分离粘连的纹线，连接断裂的纹线，平滑纹线的边缘等，以保证指纹特征提取的可靠性。

实验中进行了初步去除空洞和毛刺的处理，程序见附录 1。

4.2.5 图像细化

二值化后的纹线仍然具有一定的宽度，因为指纹识别只与纹线的走向有关，所以需要对二值图像进行细化，以减少冗余信息，突出纹线的有效特征，便于后续的特征提取。

程序如下见附录 1。

4.3 特征点提取

4.3.1 找出所有的端点和交叉点

因为特征点必然是从端点和交叉点里找出，所以在得到细化的图像后，我们首先要找出所有的端点和交叉点。首先对通过定义函数 P.m 对图像中每个点的 8 邻域位置进行坐标定义，方便后续编程。

P.m 程序见附录 2。

然后定义函数 point.m 来找出细化后图像的所有端点。

将一个点的 8 个邻域依次两两相减并取绝对值，并将所有结果相加，从细化图像的特征来说，和为 2 时为端点，和为 6 时为交叉点。

程序见附录 2。

运行完程序后，将所有的端点和交叉点全部找出。定义的数组 txy 第一项为横坐标，第二项为纵坐标，第三项为 2 或 6（2 为端点、6 为交叉点）。

4.3.2 纹线光滑处理

在指纹图像预处理中，已经对指纹图像进行过去除毛刺和空洞的处理。这里通过定义函数 guanghua.m 进一步对细化后的图像进行光滑处理。基本原理为：找到每个端点，使其沿着纹线的方向移动 5 个像素，如果在 5 个像素之内遇到交叉点，则认为此端点为毛刺，去除此点。

guanghua.m 程序见附录 2。

光滑完后需再次执行 point 函数来画出新的端点。

4.3.3 去除图像边缘的端点

可以看出，在指纹图像的边缘，由于采集仪器的关系，不可避免的多出很多端点，一方面增加了后续工作量，另一方面还可能产生错误，所以有必要将这些边缘的端点去除。本实验中设计了 cut 函数来进行处理。

程序见附录 2。

4.4 找出特征点

4.4.1 single_point 函数

经过光滑处理和去除边缘端点后进一步减少了端点和交叉点的个数。下面就要找出一些独特的端点来作为特征点。在一幅细化的指纹图上，如果一个端点的周围半径为r个像素的圆内没有任何端点或交叉点，那么随着r的逐渐变大，这样的点会越来越少，也就越来越独特。于是我们设计了一个函数single_point来找出这样的点。
程序见附录3。

4.4.2 walk 函数

在此，我们还定义了一个 walk 函数，用于进一步找出特征点，它的作用是判断离某一端点 num 距离是否有另一端点。

程序见附录 3。

4.4.3 last1 函数

综合以上两个找特征点的函数，可以设计一个新的last1函数。通过执行[pxy3,error2]=last1(thin,r,txy,num)可以找出周围半径为r个像素的圆内没有任何端点或交叉点，并且沿纹线走num个像素内没有另一个端点或交叉点的端点。

程序见附录3。

4.5 特征点匹配

4.5.1 纹线长度匹配

上面，我们已经可以找出每幅指纹图像的特征点，并画出一段纹线。下面就是匹配的问题了。在此，我们设置了三层匹配。

首先是纹线长度匹配。对于上面找出的特征点和纹线，每沿着纹线走 5 个像素测量一下到原始端点的距离。由 distance 函数得到。

程序见附录 4。

最后会得到一个装有长度信息的数组。试想如果两幅指纹图中的指纹是一样的，则它们会包含相同的特征点和从这个特征点出发画出的纹线，则这两个长度数组对应位置的比例应基本相等（考虑到老师所给的 24 幅指纹图像大小相同，这个比例应近似为 1）；最终的函数中定义了一个数 f=(sum(abs((d1./d2)-1)))，所以 f 值越接近于 0，这两幅图像的匹配度越高。

4.5.2 三角形边长匹配

找到一个特征点后，可以找出距离其最近的 2 个端点或交叉点，与原特征点构成三角形，若两幅图像的三角形边长比例相等（本实验中都为 1），则说明这两幅图像匹配。

其中，找到距离最近的端点的函数 find_point 见附录 4。

在最终程序中定义了一个数ff=(sum(abs((dd1./dd2)-1)))，所以ff值越接近于0，这两幅图像的匹配度越高。

4.5.3 点类型匹配

找到一个特征点后，在其周围找到40个端点或交叉点，统计这40个点中端点和交叉点的个数。若两幅图中端点占的比例近似相同，则两幅图像相匹配。在最终函数中定义了一个数fff=abs(f11-f21)/(f11+f12)，所以fff值越接近于0，这两幅图像的匹配度越高。

最终程序见附录 4（里面调用到前面一些函数）。

4.6 本章小结

本章简要介绍了 MATLAB 的概况及其语言特点。MATLAB 是矩阵实验室（Matrix Laboratory）之意，MATLAB 语言是一种优秀的计算机语言，具有数学运算能力是它的突出优点之一。除具备卓越的数值计算能力外，它还提供了专业水平的符号计算、文字处理、可视化建模仿真和实时控制等功能，它有着不同于其他语言的特点。本章详细介绍了指纹

识别的步骤及用 MATLAB 实现指纹识别的具体算法。通过指纹图像预处理，然后提取特征点，再找出特征点，从而通过特征点匹配达到指纹识别的目的。图像预处理包括图像规格化、图像分割、图像二值化、图像增强、图像细化；特征点提取包括找出所有的端点和交叉点、纹线光滑处理、去除图像边缘的端点；在一幅细化的指纹图上，如果一个端点的周围半径为 r 个像素的圆内没有任何端点或交叉点，那么随着 r 的逐渐变大，这样的点会越来越少，也就越来越独特，通过这样可以找出特征点；特征点匹配包括纹线长度匹配、三角形边长匹配、点类型匹配。

结　论

参考文献

1 查振元、朱华炳．电子门禁系统组成. 机电产品开发与创新. 2003，（2）：13—14

2 刘 莎、姜长生．构建基于 Intel PXA255 的指纹识别系统. 微处理机.200（5）：106—108

3 李建华，马小妹，郭成安，基于方向图的动态闽值指纹图像二值化方法. 大连理工大学学报．2002，42(5)：626-628

4 冯星奎，李林艳，颜祖泉．一种新的指纹图像细化算法. 中国图像图形学报. 1999，4(10):835-838

5 吕凤军. 数字图象处理编程入门一一做一个自己的 Photoshop. 北京：清华大学出版社，1999

6　刘文星，王雄沂，母国光. 纹线跟踪及其在细化指纹后处理中的应用. 光电子·傲光，2002，13(2)：184-187
7　刘家锋，唐降龙，赵泉. 一个基于特征点匹配的联机指纹鉴别系统. 哈尔滨工业大学学报，2002,34(1)：132-136
8　简兵，庄镇泉等. 基于脊线跟踪的指纹图细节提取算法. 电路与系统学报，2001
9　罗希平，田捷. 自动指纹识别的图像增强和细节匹配算法. 软件学报，2002-5，13（5）：946-956.
10　乔治宏. 基于细节结构的指纹特征提取及匹配算法研究. 北京：北京工业大学硕士学位论文，2004-5.

致　　谢

时光飞逝如电，随着毕业论文的完成，大学生涯也已接近尾声。回想这半年的毕业设计我感谢我的老师龙佳乐，他在学习上对我总是耐心指导，严格要求，精益求精，在课题研究和论文创作期间，龙一直是我的良师益友，引导我、督促我、甚而跟我一起学习，他严谨的治学态度和平易近人的生活方式让我钦佩值得我学习，在此致以最深的谢意。在毕业设计的研究过程中，寝室的所有同学，他们给我提供了良好的学习、工作和生活环境，谢谢他们。我要深深感谢我的亲人，他们无微不至的爱与期望是我一直奋进的动力之源。对他们，不需要说太多的感谢，我想我不懈的努力是对他们最好的回报。

最后，我想感谢这四年中碰到的所有人和事。所有的经历，无论是快乐的还是悲伤的，

所有的日子，无论是轻松的抑或是沉重的，都将成为我此生重要的财富，将作为一份最珍贵的回忆永远留在我的心中。

姜腾云

2011 年 4 月 21 日

附录 1　图像预处理代码

Pretreatment.m 程序如下：

归一化

```
Function img = My_imread(path)
M=0;var=0;
I=imread(path);
for x=1:m
    for y=1:n
        M=M+I(x,y);
    end
```

```
end
M1=M/(m*n);
for x=1:m
    for y=1:n
        var=var+(I(x,y)-M1).^2;
    end
end
var1=var/(m*n);
for x=1:m
    for y=1:n
        if I(x,y)>=M1
            I(x,y)=150+sqrt(2000*(I(x,y)-M1)/var1);
        else
            I(x,y)=150-sqrt(2000*(M1-I(x,y))/var1);
        end
    end
end
figure, imshow(uint8(I)) ;
**************************************************************************
分割
M =3;          %3*3
H = m/M;    L= n/M;
aveg1=zeros(H,L);
var1=zeros(H,L);
% 计算每一块的平均值
for x=1:H;
   for y=1:L;
       aveg=0;var=0;
 for i=1:M;
            for j=1:M;
                aveg=I(i+(x-1)*M,j+(y-1)*M)+aveg;
            end
        end
         aveg1(x,y)=aveg/(M*M);
% 计算每一块的方差值
        for i=1:M;
            for j=1:M;
                var=(I(i+(x-1)*M,j+(y-1)*M)-aveg1(x,y)).^2+var;
            end
```

```
            end
        var1(x,y)=var/(M*M);
    end
end
Gmean=0;Vmean=0;
for x=1:H
    for y=1:L
        Gmean=Gmean+aveg1(x,y);
        Vmean=Vmean+var1(x,y);
    end
end
Gmean1=Gmean/(H*L);%所有块的平均值
Vmean1=Vmean/(H*L);%所有块的方差
gtemp=0;gtotle=0;vtotle=0;vtemp=0;
for x=1:H
    for y=1:L
        if Gmean1>aveg1(x,y)
            gtemp=gtemp+1;
            gtotle=gtotle+aveg1(x,y);
        end
        if Vmean1<var1(x,y)
            vtemp=vtemp+1;
            vtotle=vtotle+var1(x,y);
        end
    end
end
G1=gtotle/gtemp;V1=vtotle/vtemp;
gtemp1=0;gtotle1=0;vtotle1=0;vtemp1=0;
for x=1:H
    for y=1:L
        if G1<aveg1(x,y)
            gtemp1=gtemp1-1;
            gtotle1=gtotle1+aveg1(x,y);
        end
        if 0<var1(x,y)<V1
            vtemp1=vtemp1+1;
            vtotle1=vtotle1+var1(x,y);
        end
    end
```

```
end
G2=gtotle1/gtemp1;V2=vtotle1/vtemp1;

e=zeros(H,L);
for x=1:H
    for y=1:L
        if aveg1(x,y)>G2 && var1(x,y)<V2
            e(x,y)=1;
        end
        if aveg1(x,y)< G1-100 && var1(x,y)< V2
            e(x,y)=1;
        end
    end
end
for x=2:H-1
     for y=2:L-1
          if e(x,y)==1
              if e(x-1,y) + e(x-1,y+1) +e(x,y+1) + e(x+1,y+1) + e(x+1,y) + e(x+1,y-1) + e(x,y-1)
+ e(x-1,y-1) <=4
                  e(x,y)=0;
              end
          end
     end
end
Icc = ones(m,n);
for x=1:H
    for y=1:L
        if   e(x,y)==1
            for i=1:M
                for j=1:M
                    I(i+(x-1)*M,j+(y-1)*M)=G1;
                    Icc(i+(x-1)*M,j+(y-1)*M)=0;
                end
            end
        end
    end
end
figure, imshow(uint8(I));title('分割');
**************************************************************************
```

二值化

```
temp=(1/9)*[1 1 1;1 1 1;1 1 1];%模板系数        均值滤波
  Im=double(I);
  In=zeros(m,n);
for a=2:m-1;
       for b=2:n-1;
In(a,b)=Im(a-1,b-1)*temp(1,1)+Im(a-1,b)*temp(1,2)+Im(a-1,b+1)*temp(1,3)+Im(a,b-1)*temp(2,
1)+Im(a,b)*temp(2,2)+Im(a,b+1)*temp(2,3)+Im(a+1,b-1)*temp(3,1)+Im(a+1,b)*temp(3,2)+Im(
a+1,b+1)*temp(3,3);
       end
end
I=In;
Im=zeros(m,n);
for x=5:m-5;
     for y=5:n-5;
       sum1=I(x,y-4)+I(x,y-2)+I(x,y+2)+I(x,y+4);
       sum2=I(x-2,y+4)+I(x-1,y+2)+I(x+1,y-2)+I(x+2,y-4);
       sum3=I(x-2,y+2)+I(x-4,y+4)+I(x+2,y-2)+I(x+4,y-4);
       sum4=I(x-2,y+1)+I(x-4,y+2)+I(x+2,y-1)+I(x+4,y-2);
       sum5=I(x-2,y)+I(x-4,y)+I(x+2,y)+I(x+4,y);
       sum6=I(x-4,y-2)+I(x-2,y-1)+I(x+2,y+1)+I(x+4,y+2);
       sum7=I(x-4,y-4)+I(x-2,y-2)+I(x+2,y+2)+I(x+4,y+4);
       sum8=I(x-2,y-4)+I(x-1,y-2)+I(x+1,y+2)+I(x+2,y+4);
       sumi=[sum1,sum2,sum3,sum4,sum5,sum6,sum7,sum8];
       summax=max(sumi);
       summin=min(sumi);
        summ=sum(sumi);
        b=summ/8;
        if (summax+summin+ 4*I(x,y))> (3*summ/8)
                  sumf = summin;
             else
                  sumf =summax;
             end
             if      sumf > b
                Im(x,y)=128;
           else
                  Im(x,y)=255;
             end
    end
```

```
end
for i=1:m
        for j =1:n
              Icc(i,j)=Icc(i,j)*Im(i,j);
        end
end
 for i=1:m
        for j =1:n
              if (Icc(i,j)==128)
                    Icc(i,j)=0;
              else
                    Icc(i,j)=1;
              end;
        end
 end

figure,imshow(double(Icc));title('二值化');
**************************************************************************************
去空洞和毛刺
u=Icc;
[m,n]=size(u) %去除空洞和毛刺
for x=2:m-1
for y=2:n-1
if u(x,y)==0
if u(x,y-1)+u(x-1,y)+u(x,y+1)+u(x+1,y)>=3
u(x,y)=1;
end
else u(x,y)=u(x,y);
end
end
end
figure,imshow(u)
%title('去除毛刺')
for a=2:m-1
for b=2:n-1
if u(a,b)==1
If
abs(u(a,b+1)-u(a-1,b+1))+abs(u(a-1,b+1)-u(a-1,b))+abs(u(a-1,b)-u(a-1,b-1))+abs(u(a-1,b-1)-u(a,
b-1))+abs(u(a,b-1)-u(a+1,b-1))+abs(u(a+1,b-1)-u(a+1,b))+abs(u(a+1,b)-u(a+1,b+1))+abs(u(a+1,
```

```
b+1)-u(a,b+1))~=1%寻找端点
if(u(a,b+1)+u(a-1,b+1)+u(a-1,b))*(u(a,b-1)+u(a+1,b-1)+u(a+1,b))+(u(a-1,b)+u(a-1,b-1)+u(a,b-1)
)*(u(a+1,b)+u(a+1,b+1)+u(a,b+1))==0 %去除空洞和毛刺
u(a,b)=0;
end
end
end
end
end
figure,imshow(u)
%title('去除空洞')
**************************************************************************
```

细化

```
v=~u;
se=strel('square',3);
fo=imopen(v,se);
v=imclose(fo,se); %对图像进行开操作和闭操作
w=bwmorph(v,'thin',Inf);%对图像进行细化
figure,imshow(w)
title('细化图')
```

附录 2　特征点提取代码

P.m 程序如下：

```
%-----Sub functions-------
function j = P (img, x, y, i)
%得到基于图像素值
%   4 | 3 | 2
%   5 |   | 1
%   6 | 7 | 8
switch (i)
    case {1, 9}
        j = img(x+1, y);
    case 2
        j = img(x + 1, y-1);
    case 3
        j = img(x, y - 1);
```

```
        case 4
            j = img(x - 1, y - 1);
        case 5
            j = img(x - 1, y);
        case 6
            j = img(x - 1, y + 1);
        case 7
            j = img(x, y + 1);
        case 8
            j = img(x + 1, y + 1);
End
```

****

Poin 函数程序如下：

```
function txy=point(thin)
count = 1;
txy(count, :) = [0,0,0];
siz=min(size(thin,1),size(thin,2));
for x=40:siz - 40
    for y=40:siz - 40
        if (thin(y, x) )
            CN = 0;
            for i = 1:8
                CN = CN + abs (P(thin, y, x, i) - P(thin, y, x, i + 1));
            end
            if (CN == 2)
                txy(count, :) = [x, y,2];
                count = count + 1;
            end
            if (CN == 6)
                txy(count, :) = [x, y,6];
                count = count + 1;
            end
        end
    end
end
for i=1:count - 1
    x(i) =txy(i, 1);
    y(i)= txy(i, 2);
end
```

```
imshow(double(thin));
hold on;
plot(x,y,'.');
**********************************************************************************
guanghua.m 程序如下：
function    w=guanghua(thin,txy)
for j=1:5
      txy=point(thin);
      pxy=txy(find(txy(:,3)==2),:);
      n=size(pxy,1);
      for i=1:n
            error=0;
            error=walk(thin,pxy(i,1),pxy(i,2),5);
            if    error==1
            thin(pxy(i,2),pxy(i,1))=0;
            end
      end
end
w=thin;
imshow(w);
**********************************************************************************
cut 函数程序如下：
function txy=cut(thin,txy)
s(8,8)=0;
delta(8,8)=0;
n=size(txy,1);
for i=1:8
      for j=1:8
            mp{i,j}=thin(1+31*(i-1):31+31*(i-1),1+31*(j-1):31+31*(j-1));
            s(i,j)=sum(sum(mp{i,j}))/(31*31);
            mp{i,j}=(mp{i,j}-s(i,j)).^2;
            delta(i,j)=sum(sum(mp{i,j}));
            if delta(i,j)<=70
                  for k=1:n
if
(txy(k,1)>=1+31*(i-1)&&txy(k,1)<=31+31*(i-1)&&txy(k,2)>=1+31*(j-1)&&txy(k,2)<=31+31*
(j-1)&&txy(k,3)==2)
                              txy(k,:)=[0,0,0];
                        end
```

```
            end
        end
    end
end
txy=txy(find(txy(:,1)),:);
plot(txy(:,1),txy(:,2),'ro');
```

附录 3　找特征点代码

single_point 函数程序如下：

```
function [pxy2,error]=single_point(txy,r)
error=0;
x=txy(:,1);
y=txy(:,2);
n=length(x);
d(1:n,1:n)=0;
for j=1:n
    for i=1:n
        if (i~=j)
        d(i,j)=sqrt((x(i)-x(j))^2+(y(i)-y(j))^2);
        else
            d(i,j)=2*r;
        end
    end
```

```
end
[a,b]=min(d);
c=find(a>r);
pxy2=txy(c,:);
pxy2=pxy2(find(pxy2(:,3)==2),:);
t=size(pxy2,1);
if t==0
    error=1
else
    plot(x,y,'b.');
    hold on
    plot(pxy2(:,1),pxy2(:,2),'r.');
end
```

****

walk 函数程序如下：

```
function [error,a,b]=walk(thin,x0,y0,num)
error=0;
thin(y0,x0)=0;
t1=0;
for n=1:num
  if error==1
          break;
      else
    x=x0;
    y=y0;
    for x=x0-1:x0+1
        if error==1
            break;
        else
        for y=y0-1:y0+1
            t1=sum(sum(thin(y0-1:y0+1,x0-1:x0+1)));
            if (t1==0||t1>=2)
                error=1;
                a=x0;
                b=y0;
              break;
        else
            if (thin(y,x)==1&&(x-x0)^2+(y-y0)^2~=0)
                if   (t1>=2 )
```

```
                    error=1;
                break ;
                else
                thin(y,x)=0;
                x0=x;
                y0=y;
                a=x0;
                b=y0;
                plot(x0,y0,'r.')
                end
            end
        end
        end
        end
    end
    end
end
```

****

last1 函数程序如下：

```
function [pxy3,error2]=last1(thin,r,txy,num)
error=0;
[pxy2,error]=single_point(txy,r);
n=size(pxy2,1);
l=1;
error2=0;
for i=1:n
    [error,a,b]=walk(thin,pxy2(i,1),pxy2(i,2),num);
    if error~=1
      pxy3(l,1)= pxy2(i,1);
      pxy3(l,2)= pxy2(i,2);
      pxy3(l,3)= pxy2(i,3);
      l=l+1;
      error2=0;
      plot(pxy2(i,1) ,pxy2(i,2),'r+');
    end
end
```

附录 4　特征点匹配代码

distance 函数程序如下：

```
function d=distance(x0,y0,num,thin)
num2=fix(num/5);
for i=1:num2
    [error,a,b]=walk(thin,x0,y0,5*i);
    if error~=1
        d(i)=sqrt((a-x0)^2+(b-y0)^2);
    else
        break;
    end
End
```

****

find_point 函数如下：

```
function pxy=find_point(x0,y0,txy,num)
x=txy(:,1);
y=txy(:,2);
```

```
n=length(x);
l(1,n)=0;
lnn=1;
pxy(num,:)=[0,0,0];
for i=1:n
    l(i)=sqrt((x(i)-x0)^2+(y(i)-y0)^2);
end
ll=sort(l);
for i=1:num
xiao=ll(i+lnn);
nn=find(l==xiao);
lnn=length(nn);
pxy(i,:)=[x(nn(1)),y(nn(1)),txy(nn(1),3)];
end
plot(x0,y0,'bo');
x0;
y0;
hold on
plot(pxy(:,1),pxy(:,2),'ro');
```

点类型匹配最终程序：

```
close all;
tic
clear;
thin1=tuxiangyuchuli('zhiwen8.bmp');
thin2=tuxiangyuchuli('zhiwen9.bmp');
figure;
txy1=point(thin1);
txy2=point(thin2);
[w1,txy1]=guanghua(thin1,txy1);
[w2,txy2]=guanghua(thin2,txy2);
thin1=w1;
thin2=w2;
txy1=cut(thin1,txy1);
txy2=cut(thin2,txy2);
[pxy31,error2]=last1(thin1,8,txy1,60)
[pxy32,error2]=last1(thin2,8,txy2,60)
error=1;
num=20;
cxy1=pxy31;
```

```
  cxy2=pxy32;
  d1=distance(cxy1(1,1),cxy1(1,2),num,thin1);
  d2=distance(cxy2(1,1),cxy2(1,2),num,thin2);
f=(sum(abs((d1./d2)-1)));
if ff<=0.5
      error=0;
else
      error=1;
end
  c11=find_point(cxy1(1,1),cxy1(1,2),txy1,1);
   c12=find_point(cxy1(1,1),cxy1(1,2),txy1,2);
   c21=find_point(cxy2(1,1),cxy2(1,2),txy2,1);
   c22=find_point(cxy2(1,1),cxy2(1,2),txy2,2);
   cxy1(2,:)=c11;
   cxy1(3,:)=c12(2,:);
   cxy2(2,:)=c21;
   cxy2(3,:)=c22(2,:);
   x11=cxy1(1,1);   y11=cxy1(1,2);
   x12=cxy1(2,1);   y12=cxy1(2,2);
   x13=cxy1(3,1);   y13=cxy1(3,2);
   x21=cxy2(1,1);   y21=cxy2(1,2);
   x22=cxy2(2,1);   y22=cxy2(2,2);
   x23=cxy2(3,1);   y23=cxy2(3,2);
   dd1(1)=juli(x11,y11,x12,y12);
   dd1(2)=juli(x12,y12,x13,y13);
   dd1(3)=juli(x13,y13,x11,y11);
   dd2(1)=juli(x21,y21,x22,y22);
   dd2(2)=juli(x22,y22,x23,y23);
   dd2(3)=juli(x23,y23,x21,y21);
   ff=(sum(abs((dd1./dd2)-1)))
   if ff<=1
        error=0;
   else
        error=1;
   end
   cxy1(2:41,:)=find_point(pxy31(1,1),pxy31(1,2),txy1,40);
   cxy2(2:41,:)=find_point(pxy32(1,1),pxy32(1,2),txy2,40);
   f11=length(find(cxy1(:,3)==2));
   f12=length(find(cxy1(:,3)==6));
```

```
f21=length(find(cxy2(:,3)==2));
f22=length(find(cxy2(:,3)==6));
fff=abs(f11-f21)/(f11+f12)
toc
```