Intelligence Door Control System of Face Recognition

人脸识别门禁控制系统

万君社，胡学萱，胡争辉

摘要

人脸是一个包含着丰富信息的模式的集合，是人类互相辩证和识别的主要标志之一，也是图像和视频中视觉感兴趣的对象之一。与指纹、虹膜、语音等其他人体生物特征相比，人脸识别更加直接、友好，是当前模式识别和人工智能领域的一个研究热点。

然而，由于人脸识别将人脸作为生物特征同样带来一些困难，如不同个体之间的区别不大，人脸的外形很不稳定，受表情、角度、光照、年龄等各方面的影响还是比较大，因此人脸识别现在还不是很成熟，需要后继者们进一步的研究。

在本文中，我们利用 PC 作为计算平台，MFC 做界面，配置高清红外摄像机，构成在线人脸识别系统的主要部分，另外我们通过 Oracle 数据库来实现对数据的存储和访问，利用人脸识别的结果来控制门禁，最后形成了一套完整的人脸识别系统，这套系统从结构和整体布局上可以直接应用实际场合，具有很大的实用价值。

在这个系统中，为了解决采集到的图像易受环境光线影响的问题，我们采用红外摄像头对图像进行采集，然后用 Adaboost 算法对采集的图像进行人脸检测，得到大概的人脸区域，再用主动表观模型（AAM）进行人脸描述，获得形状无关图像，接着提取局部二值模式（LBP）特征，得到光照完全无关的特征表达，最后用分类器进行识别。实践表明，这个系统一定程度上克服了光照、环境、表情等因素的影响，具有一定的应用价值。

Obtain probableface region, and then use AAM to describe and acquire Shape-independent image , do LBP to get Light-independent expressionfeature.Finally useclassifierto identify. Practice shows thissystem isto some extentovercome the impact ofthe light, environment, facial expressionsand other factors，and has some application value

关键词：人脸； 识别；Adaboost；AAM；LBP；

ABSTRACT

Face is a collection which contains a wealth of information, and becomes one of the main signs of human dialectic and mutual recognition now, it is also one of the objects of visual interest to images and video. Compared with other biometric such as fingerprint, iris, voice, etc., face recognition shows more directly and friendly. It is currently a hot topic in the field of pattern recognition and artificialintelligence.

However, face recognition also brings some difficulties because it regards face as the biometric feature, the difficulties such as little difference between different individuals, Face shape is very unstable, and largely affected by expression, angle, light, and age , etc. So recognition is not very mature now, successors'further studies are needed.

In this article, we use the PC as a computing platform, use MFC to do interface,deploy definition infrared camera,those constitute the main part of the online face recognition system. In addition, we come to realize the data stored and accessed through Oracle Database, and we use the results of face recognition to control access system. Finally, the formation of a face recognition system is completed. Based on the structure of the system and theoverall layout of the actual situation, it can be directly applied in the actual occasion.

In this system, in order to solve the issue which the collected images vulnerable to environmental light,we use the infrared camera for image acquisition, then use the Adaboost algorithm to capture images for face detection.Obtainprobable face region, and then use AAM to describe and acquire Shape-independent image, do LBP to get Light-independent expression feature. Finally use classifier to identify. Practice shows this system is to some extent overcome the impact of the light, environment, facial expressions and other factors，and has some application value.

Keywords: Face; Recognition; AAM; LBP;

目录

第一章绪论

1.1 课题研究背景及意义

1.1.1 人脸识别的发展现状

人脸识别的研究开始于上世纪七十年代，当时的研究主要是基于人脸外部轮廓的方法。近些年来，随着计算机计算能力的加强，基于视频流下的人脸识别发展迅速，各种面向复杂应用背景的视频人脸识别系统也随之涌现。

由于基于视频流下的人脸识别系统具有如此大的应用前景，它引起了许多国家的高度关注。国内外众多的大学和研究机构，如美国的 CMU, MIT, UIUC 大学、英国的剑桥大学、日本的 Toshiba 公司和国内的清华大学、中科院自动化所等单位都对基于视频的人脸识别进行了广泛而深入的研究，尽管基于视频的人脸识别技术取得了很大成果，但在实际应用中还存在很大的局限性，面临着许多困难与瓶颈，这些问题也决定了基于视频人脸识别技术的发展趋势及今后的研究方向，总体归为以下几类：

（1）人的脸部结构相似，甚至人的五官结构、脸部纹理和外形都很相似。由于不同个体之间的区别不大，这样的特点对于检测人脸的存在有利，但是对用人脸区分人的身份不利。

（2）人可以通过脸部变化产生丰富的表情，同时通过不同的视角观察人脸，其视觉图像差别很大，甚至会出现同一个人在不同情况下的脸内差异大于不同人脸之间的脸间差异，因此人脸的外部形态很不稳定给识别带来很大的困难。

（3）背景的变化及复杂程度在很大程度上会影响人脸检测的准确度，例如将人脸判断为非人脸、将非人脸误判为人脸等。而视频监控系统往往设置在背景较复杂的场所，所以解决背景干扰问题成为一个重要问题，同时也是一个难点问题。

（4）视频监控区域往往出现一个以上的人脸，如何将所有人脸都准确地进行定位是一个比较困难的地方。

（5）与姿势和观察角度的变化相似，不同的光照条件也会造成识别的困难。即使是同一个人，在相同的表情和视角情况下，人脸的区别也很大。

（6）人脸遮挡物的不同，同样会给识别带来很大的难度。而且，同一个人随着年龄的变化，人脸特色会出现很大的不同，因此会出现采集到的人脸与人脸库中的同一个人的人脸差异很大。其中第一类的变化是个体间的变化，称之为类间变化；第二类变化是同一个体间不同情况下的变化，称之为类内变化。很多情况下，类内变化甚至会大于类间变化，从而使受类内变化干扰情况下的人脸识别准确率大大降低。

1.1.2 人脸识别研究的意义

人脸识别是机器视觉和模式识别领域最富有挑战性的课题之一，同时也具有较为广泛的应用意义。人脸识别因其在安全验证系统、信用卡验证、医学、档案管理、视频会

议、人机交互、系统公安(罪犯识别等)等方面的巨大应用前景而越来越成为当前模式识别和人工智能领域的一个研究热点。

而且，人脸识别技术又是一个非常活跃的研究领域，它覆盖了数字图像处理、模式识别、计算机视觉、神经网络、心理学、生理学、数学等诸多学科的内容。如今，虽然在这方面的研究已取得了一些可喜的成果，但是人脸识别在实用应用中仍面临着很严峻的问题,因为人脸五官的分布是非常相似的，而且人脸本身又是一个柔性物体，表情、姿态或发型、化妆的千变万化都给正确识别带来了相当大的麻烦。如何能正确识别大量的人并满足实时性要求是迫切需要解决的问题。

1.2 论文主要研究内容

本文首先对人脸识别算法进行分析，然后搭建在线人脸识别系统，构建数据库，最后实现由人脸对门禁进行控制。主要工作内容如下：

第一章对人脸识别系统的发展历史和现状进行分析，根据现在人脸识别本身的特点和未来的市场潜力提出了本课题研究的目的和意义。

第二章主要是对人脸识别的算法进行详细的讲解和分析，主要围绕人脸识别几个主要的流程加以算法上的说明，包括：基于 Adaboost 的人脸检测，形状无关图像的获取 AAM、以及提取局部二值模式 LBP。

第三章主要介绍了人脸识别中数据库的管理问题，主要包括表的设计，对 BLOB 类型字段的存取，选择 ADO 连接 Oracle 数据库，以及介绍了如何利用 ADO 进行数据库的开发。

第四章主要介绍了人脸识别的门禁系统，包括发送消息的获取，相关协议知识的介绍，以及在这个系统中代码的实现。

第五章着重介绍了人脸识别系统的实现，包括系统硬件介绍，结构流程等信息，又重点讲解了人脸识别的两大重要模块——录入和识别功能的实现，最后进行了测试。

第二章人脸识别算法简介

2.1 人脸识别流程

完成人脸识别的工作需要经过一系列的步骤，他们结合起来构成一个完整的流程。其主要步骤包括：人脸检测/跟踪(face detection/tracking)，特征提取(feature extraction)，特征降维(feature dimensionality reduction)，匹配识别(matching and classification)。其流程如图 2-1 所示。

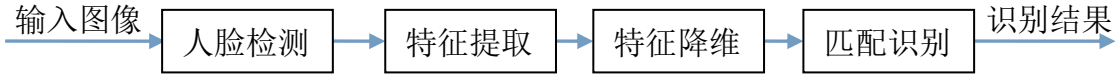

图 2-1

这几个步骤基本上是按照串行步骤来的，当然在有的系统中也有将不同的步骤进行结合，或者加入反馈的流程。在这个系统中按照串行关系进行设计，按步骤进行处理。在本章的剩下的小节中我讲对各个步骤进行详细说明。

2.2 人脸检测/跟踪

人脸检测是完成人脸识别工作的第一个步骤。该步骤的目的是在输入的图像中寻找人脸区域。具体来说：给定任意一幅图像，确定是否图像中有人脸存在，如果存在就标出人脸的位置和范围。在本系统中采用的是基于 Adaboost 的人脸检测方法，得到人脸的位置信息，进行保存，供下一个步骤使用，然后在实时图像上标出人脸区域。

对于该检测方法,在 opencv 中其实已经将其做成了一个人脸检测器,给出开源代码,非常方便。

人脸检测器从上到下分为 3 个组成部分：

(1)Haar 矩形特征的提取。

(2)基于 Adaboost 的强分类器。

(3)强分类器的级联。

下面分别介绍各个模块的细节。

2.2.1 Haar 矩形特征

基本的 Haar 矩形特征包括 4 种形式，如图 2-2 所示。

图 2-2

为了能够尽量使得 Haar 特征适应检测目标的灰度分布，此后的文献对 Haar 特征的扩充不断出现。较具代表性的有 45 度的 Haar 特征，以及在此基础上扩展的 14 种 Haar 特征原型，分别表示为边缘、线形和中心特征，如图 2 - 3 所示。

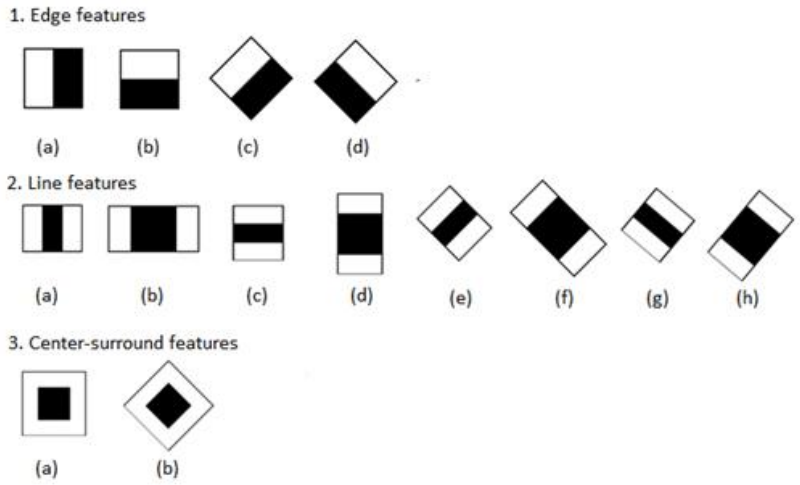

图 2 - 3

为了快速计算 Haar 特征，可借助积分图像，任意图像 f 的积分图像 g 在任何一个像素点(x,y)的取值定义为:

$$g(\mathrm{x},\mathrm{y})=\sum_{x^{'}<=x,y^{'}<=y} f(\mathrm{x}^{'},\mathrm{y}^{'})$$

这样通过积分图像就能加快处理的速度，提升整个算法的效率。

2.2.2 基于 Adaboost 的特征选择

给定 Haar 特征定义和包含正面样本和负面样本的训练集合，可以将训练样本图像转换到特征空间。由于 Haar 特征集合十分庞大，即使每个特征的计算十分高效，利用所有特征进行分类也是不现实的，所以需要采用 Adaboost 算法同时进行特征的选择和分类器的训练。

Adaboost 算法可以将一组弱分类器自适应的提升为强分类器，由于每个弱分类器输出的是±1，所以又称为离散 Adaboost。整个算法包括一下步骤：

(1)初始化每个样本的权值 $\omega_{\mathrm{i}}=1/N,i=1,2,\ldots N$ 。

(2)对每个样本，利用弱分类器学习算训练弱分类器 $\mathrm{f}_t(\mathrm{x})\in\{-1,1\}$，并计算错误率 $\varepsilon_{\mathrm{t}}=E_w[I(\mathrm{y}\neq \mathrm{f}_t(\mathrm{x}))]$ 和加权系数 $\mathrm{a}_{\mathrm{t}}=\log[(1-\varepsilon_t)/\varepsilon_t]$。

(3)更新权值 $\omega_i \leftarrow \omega_i \exp[\alpha_t \bullet I\ (f_t(x) \neq y)]$，并重新归一化使得 $\sum_i w_i=1$。

(4)输出分类器 $H(x)=\mathrm{sgn}[\sum_{t=1}^{T} a_t f_t(x)]$。

结合以上几个步骤可以看出，Adaboost 算法每次选择最佳的弱分类器，等价于选择最佳的特征。假设共有 M 个特征，从中选择错误率最小的特征，每次迭代的计算复杂度为 O(MN)。

当算法执行 T 次迭代时，整个训练流程的计算复杂度为 O(TMN)。基于 Adaboost 的人脸检测结果如 2 - 4 所示。

图 2 - 4

2.3 特征提取

对于特征提取也是分几个步骤进行的，在这个系统中，着重介绍两个重要的步骤，即形状无关图像的获取(AAM)和 LBP 特征的提取。

2.3.1 形状无关图像的获取(AAM)

在一幅人脸图像中，人脸图像可能只占其中的一部分区域，而且其所在的位置和所处的背景都不尽相同。而需要分析的是能够真正地反映出人脸纹理信息的那部分纹理，也就是在标定点的外轮廓范围以内图像的纹理信息。一般来讲先将训练样本的人脸图像变形到平均人脸形状中，使向量的维数统一起来并具有相同的对应关系，从而得到与形状无关的纹理信息。实现上述过程的具体步骤有两个：

(1)采用 Delaunay 三角剖分算法，将形状模型中的平均形状进行三角剖分，通过三角剖分将平均人脸划分成若干三角形的集合，如图 2 - 5 所示。

(2)以每个三角形为单位将人脸变形到平均人脸上。因为三角剖分后，人脸图像就是由三角形集合组成的，所以通过对形状中每个三角形的变形就可以实现整个人脸图像向平均形状的变形。

图 2 - 5

在向平均形状变形之后，任意的对象都可以用形状参数和纹理参数来表示，而且两者之间还具有一定的相关性，可以把二者通过一定的权值联合形成一个新的向量，即:

$$\mathrm{b}=\begin{bmatrix} b_s \\ w_g g_w \end{bmatrix}=\begin{bmatrix} P_s^T() \\ w_g P_s^T() \end{bmatrix}$$

在生成连接参数 b 之后，对参数 b 在进行一次主成分分析，得到变换矩阵 Q，因此任意人脸图像的表观向量可近似表示成:

$$\mathrm{b}=\mathrm{Qc}$$

其中 c 为表观模型参数，可以由下述公式求出：

$$\mathrm{c}=\mathrm{Q}^T\mathrm{b}$$

这样就建立了表观模型参数。

2.3.2 特征提取(LBP)

LBP 英文全称为 local binary pattern,中文为局部二值模式。特是一种描述图像局部区域纹理变化的算子。LBP 计算远离如下：选取某一像素为中心，将中心点像素的灰度值作为阈值并与其淋雨的像素灰度值进行数值大小比较，并把大于预知的点置为 1，反之为 0，从而得到一组二进制数。这组数通过以下式计算得到一个 LBP 值：

$$LBP_{N,R}(\mathrm{x}_c,\mathrm{y}_c)=\sum_{p=0}^{N-1}s(\mathrm{g}_p,\mathrm{g}_c)\bullet 2^p, s(\mathrm{x})=\begin{cases}1 & x\geq 0\\ 0 & x<0\end{cases}$$

其中 g_c 为中心点像素 $(\mathrm{x}_c,\mathrm{y}_c)$ 的灰度值，N 是半径为 R 的邻域内的像素个数，g_p 为邻

域点 p 的灰度值，原始的 LBP 算子定义为在 3*3 的窗口内，以窗口中心像素为阈值，将相邻的 8 个像素的灰度值与其进行比较，若周围像素值大于中心像素值，则该像素点的位置被标记为 1，否则为 0。这样，3*3 领域内的 8 个点可产生 8bit 的无符号数，即得到该窗口的 LBP 值，并用这个值来反映该区域的纹理信息。如下图 2 - 6 所示：

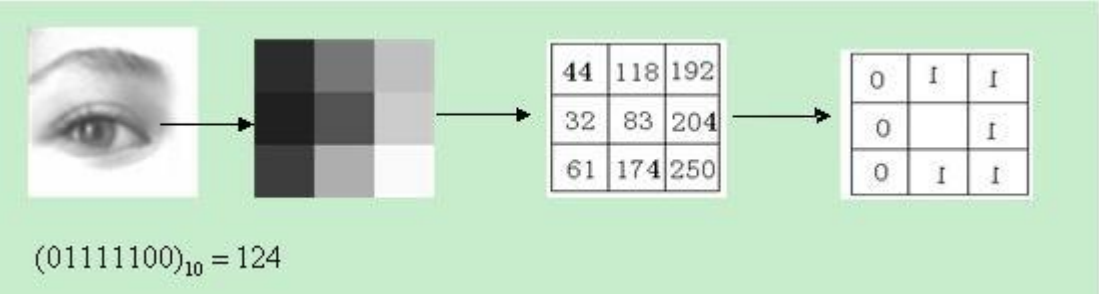

图 2 - 6

在得到每个像素的 LBP 编码描述后，可以采用统计直方图方法对所有像素进行统计，便可以得到图像的 LBP 直方图描述。而实际上在进行直方图统计的过程中也是起到了一个降维的作用。在这里就不再对降低维数进行说明。

2.4 匹配识别

对于向量的匹配方法比较常用的有欧式距离，但是在这个系统中，提取的是 LBP 特征，进行编码，最后得到的是图像的 LBP 直方图描述，因此在这里所做的匹配识别其实是直方图的匹配识别，也就是说将实时图像的 LBP 直方图和训练样本中的直方图进行一一比对，找出样本中最接近的直方图，然后得到对应训练样本的其他信息，显示出识别结果。

2.5 本章小结

本章主要主要是对人脸识别的算法进行详细的讲解和分析。

第一部分介绍人脸识别的流程，从总体上把握做好一个人脸识别系统的关键。

第二部分介绍人脸识别流程的第一个部分——人脸检测，也详细介绍了人脸检测的关机算法基于 Adaboost 的人脸检测。

第三部分介绍介了人脸的特征提取的一些算法包括：形状无关图像的获取(AAM)以及基于局部二值模式的提取特征的方法(LBP)。

第四部分介绍了人脸特征提取后的匹配方法，在这个系统中采用的是直方图匹配法。

第三章人脸识别中数据库管理

3.1 Oracle 数据库

ORACLE 是以高级结构化查询语言(SQL)为基础的大型关系数据库，它功能强大，支持大量多媒体数据，在数据库管理、完整性检查、安全性、一致性方面都有良好的表现。提供了与第三代高级语言的接口软件，支持 C/S 和 B/S 工作模式，因而被大量用到信息系统的开发中。

3.2 表的设计

本项目中，主要使用和管理员工的信息，以及对曾经进行人脸识别的记录，因此，设计两个表：员工表 employee 和比对记录表 comRecord。

（1） 员工表存储员工的基本信息，如员工 ID，员工姓名，员工部门，员工照片，用于比对的该照片的五个特征值。如表 3 -1 ：

表 3 -1 employee 表结构

属性名	属性类型	长度	约束
employeeID	Number	5	not null, primary key
name	Varchar2	20	
department	Varchar2	40	
image	blob		
Eigenvalue1	Varchar2	800	
Eigenvalue2	Varchar2	800	
Eigenvalue3	Varchar2	800	
Eigenvalue4	Varchar2	800	
Eigenvalue5	Varchar2	800	

（2） 比对记录表存储比对信息，如比对记录 ID，比对成功与否，比对时间，比对时的照片，比对成功还存储员工号。如表 3 -2：

表 3 -2 comRecord 表结构

属性名	属性类型	长度	约束
comRecordID	Number	5	not null, primary key
Sorf	Number	1	
currentImage	blob		
comTime	date		

employeeID	number	5	foreign key, references employee (employeeID)
Eigenvalue2	Varchar2	800	
Eigenvalue3	Varchar2	800	
Eigenvalue4	Varchar2	800	
Eigenvalue5	Varchar2	800	

3.3 对 blob 类型字段的存取

照片字段用 blob 类型存储，对 blob 类型数据的读取以及存储，不能用与一般的其他类型数据相同的方法。由于，使用存储过程可提高数据库执行速度、完成复杂功能、重复使用、安全性高。因此，本项目用存储过程实现对 blob 类型数据的存取，ADO 实现对该存储过程的调用。

（1）存储 blob 类型数据

首先，需要创建一个 directory，用以存放图片的文件夹地址；

CREATE OR REPLACE DIRECTORY IMAGES AS 'd:\faceRecognition\images';

其次，创建存储过程，存储 blob 类型数据。

在存储 blob 数据时，先要对 blob 数据初始化为 EMPTY_BLOB()，然后从文件中向数据库加载图片到 BLOB 类型的变量。存储过程及其执行情况如图一。

（2）读取 blob 类型数据

首先，找到要读取记录 blob 字段数据；

然后，读取该数据并存入指定的文件中。

存储过程及其执行情况如图二。

将文件 psb.jpg 中的图片存入数据库的 blob 字段，以及从 blob 字段读取图片存入指定新文件 out.jpg 的存取过程中，所用到的和产生的文件如图 3 -1、3 -2 所示。

```
SQL> CREATE OR REPLACE PROCEDURE IMG_GET_BY_ID (
  2  TID                                NUMBER,
  3  FILENAME       VARCHAR2
  4  ) AS
  5  l_file              UTL_FILE.FILE_TYPE;
  6  l_buffer            RAW(32767);
  7  l_amount       BINARY_INTEGER := 32767 ;
  8  l_pos               INTEGER := 1;
  9  l_blob              BLOB ;
 10  l_blob_len          INTEGER ;
 11  BEGIN
 12  SELECT IMAGE INTO l_blob FROM EMPLOYEE WHERE  ID = TID ;
 13  l_blob_len := DBMS_LOB.GETLENGTH(l_blob) ;
 14  l_file := UTL_FILE.FOPEN('IMAGES', FILENAME, 'wb' , 32767);
 15  WHILE l_pos < l_blob_len LOOP
 16  DBMS_LOB.READ(l_blob, l_amount, l_pos, l_buffer);
 17  UTL_FILE.PUT_RAW(l_file, l_buffer, TRUE);
 18  l_pos := l_pos + l_amount ;
 19  END LOOP;
 20  UTL_FILE.FCLOSE(l_file);
 21  EXCEPTION
 22  WHEN OTHERS THEN
 23  IF UTL_FILE.IS_OPEN(l_file) THEN
 24  UTL_FILE.FCLOSE(l_file);
 25  END IF;
 26  RAISE;
 27  END;
 28  /

过程已创建。

SQL> EXEC IMG_GET_BY_ID(1,'out.jpg');

PL/SQL 过程已成功完成。

SQL>
```

图 3 -1 读取照片的存储过程

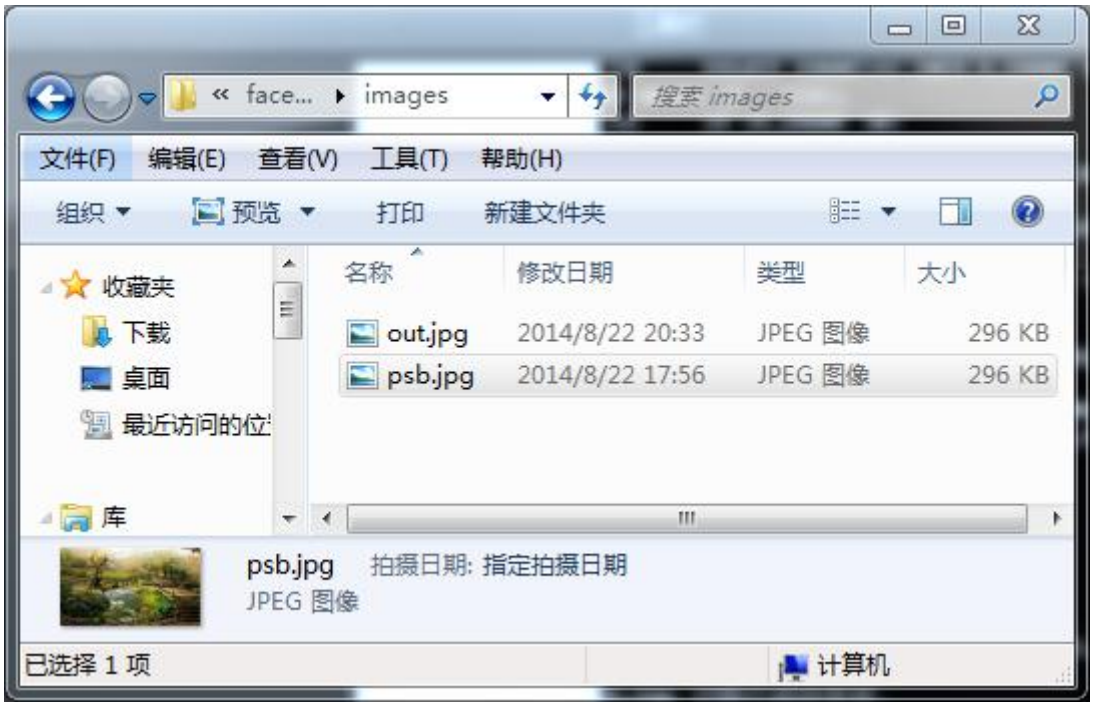

图 3 -2 存取照片所用文件及产生的文件

3.4 选择 ADO 连接 oracle

Microsoft 提供了许多相关组件支持数据库的访问，Oracle 公司也提供了 Oracle 数据库应用程序的开发接口。在 VC 中连接 Oracle 有以下三类方法：

（1）VC 提供多种数据库访问技术，如 ActiveX 数据对象 ADO、开放数据库连接 ODBC、数据存取对象 DAO、对象连接和嵌入数据库 OLE DB 等。

（2）Oracle 公司提供的在第三代高级程序设计语言中嵌入 SQL 语句来访问数据库的技术，Pro*C/C++（简称 PROC）。

（3）Oracle 公司提供的底层接口开发工具 OCI，全面支持 Oracle 的面向对象技术，具有速度快、支持第三代编程语言、对 Oracle 数据库的控制功能强等优点。

比较这 3 种方式，ADO 是是建立在 OLE DB 之上的高层数据库访问技术，它开发起来比较容易；OCI 是一种底层接口，几乎可以操纵 Oracle 数据库的任何对象，开发起来难度大一些，但是它的速度极快；使用 PROC 进行数据库开发效率高，但要求对 Oracle 数据库的运行机制十分了解。ADO 虽没有 OCI 速度快，但足够应付本项目中数据管理的需求，更因为其简单易用，因此选择 ADO 连接 oracle，并实现数据库的各种操作。

3.5 ADO 进行数据库开发

其基本流程如下：

（1）初始化 COM 库，引入 ADO 库定义文件；

（2）用 Connection 对象连接数据库；

（3）利用建立好的连接，通过 Connection、Command 对象执行 SQL 命令，或利用 Recordset 对象取得结果记录集进行查询、处理。

（4）使用完毕后关闭连接释放对象。

使用 ADO 连接数据库，以及对数据进行增删改操作代码如下：

（1）连接 oracle

```
void ADO::Connect(void)
{
    try{
        cout<<"正在连接数据库，请稍等!"<<endl;
```

```
            ::CoInitialize(NULL);
            m_pConnection.CreateInstance(__uuidof(Connection));
            HRESULT hr=m_pConnection->Open("Provider=OraOLEDB.Oracle.1;Persist Security Info=True;User ID=hxx;Password=Manager01;Data Source=ORCL", "", "",adModeUnknown);
            if(hr!=S_OK)
                cout<<"不能连接到数据库，请确认数据库是否开启！"<<endl;
            else
                cout<<"连接成功!"<<endl;
        }
        catch(_com_error e){
            cout<<(char *)e.Description()<<endl;
        }
    }
```

（2）执行增删改操作

```
    {
    string sql;
        ADO record;
        _RecordsetPtr m_pRecordset;

            //插入
            sql = "insert into employee(ID,name,department,eigenvalue1,eigenvalue2,eigenvalue3, eigenvalue4,eigenvalue5) values ('3','liuhai','PD','','','l','19','97')";
            _bstr_t bstr_t(sql.c_str());
            _variant_t RecordsAffected;
            record.m_pConnection->Execute(bstr_t,&RecordsAffected,adCmdText);
            cout<<"插入成功!"<<endl;
```

```
        }
        {
            //修改
            sql = "update employee set name='wangqing',department='LD',eigenvalue1='',eigenvalue2='',
eigenvalue3='',eigenvalue4='',eigenvalue5='' where ID='3'";
            _bstr_t bstr_t(sql.c_str());
            _variant_t RecordsAffected;
            record.m_pConnection->Execute(bstr_t,&RecordsAffected,adCmdText);
            cout<<"修改成功!"<<endl;
        }
        {
            //删除
            sql = "delete from employee";
            _bstr_t bstr_t(sql.c_str());
            _variant_t RecordsAffected;
            record.m_pConnection->Execute(bstr_t,&RecordsAffected,adCmdText);
            cout<<"删除成功!"<<endl;
        }
        record.ExitConnect();
        return 0;
    }
```

执行结果如图 3 -3 所示：

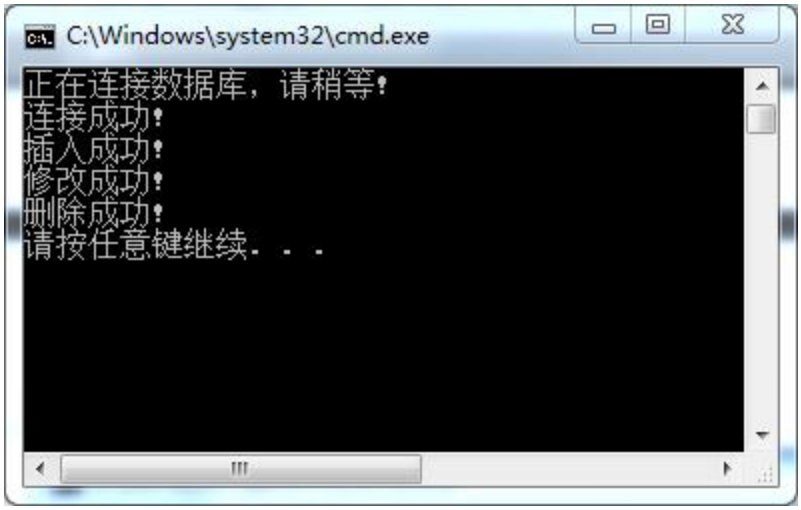

图 3 -3 ADO 操作数据库结果

3.6 ADO 调用存储过程

ADO 调用存储过程的流程如下：

（1）生成并初始化一个_CommandPtr 对象；

（2）生成调用存储过程需要的参数，这些参数都是_ParameterPtr 对象；

（3）按照顺序将使用_CommandPtr 的 Append 方法为存储过程提供参数（包括输入参数和输出参数）；

（4）为_CommandPtr 对象指定需要使用的 ADO 连接；

（5）使用_CommandPtr 的 Execute 方法调用存储过程；

（6）从结果中获取返回参数的值（如果有的话）。

调用代码如下：

```
_RecordsetPtr& call_IMG_INSERTORUPDATEE_BY_ID(int ID,_variant_t filename)
{
    ADO record;
    _ConnectionPtr    conn;
    _CommandPtr          cmmd;
    _ParameterPtr     param;
    _RecordsetPtr     recordset;
    record.Connect();
    recordset=NULL;
```

```
        HRESULT hr = cmmd.CreateInstance(__uuidof(Command));
        if(FAILED(hr))
        {
            cout<<"创建_CommandPtr 对象失败！"<<endl;
            recordset=NULL;
            return recordset;
        }
        param = cmmd->CreateParameter("ID",adNumeric, adParamInput,sizeof(int),ID);
        cmmd->Parameters->Append(param);
        param = cmmd->CreateParameter("filename",adVarChar,
adParamInput,sizeof(string),filename);
        cmmd->Parameters->Append(param);
        cmmd->CommandText=_bstr_t("IMG_INSERTORUPDATEE_BY_ID");
        cmmd->ActiveConnection = record.m_pConnection;
        cmmd->CommandType=adCmdStoredProc;
        cmmd->Execute(NULL, NULL, adCmdStoredProc);
        recordset=(long)cmmd->Parameters->GetItem("ID")->GetValue();
        cmmd.Detach();
    }
```

3.7 本章小结

本章主要的是人脸识别中数据库的管理，首先从需求出发，介绍了本系统中表的设计，然后用 ADO 连接 Oracle 数据库，最后又详细介绍了利用 ADO 进行数据开发的一些技术细节和调用存储过程的方法。

第四章门禁系统

4.1 门禁系统简介

本系统采用的网络门禁基于 C/S 架构，门禁为 Server，通过 TCP/IP 接受客户端的连接。然后通过客户端发送指定协议的命令来控制门禁。

4.2 客户端控制门禁

4.2.1 连接 Server

在连接门禁服务器的时候需要先定义协议，由于卖家给的门禁系统 SDK 包不全，没有完整的协议说明，所以需要自己去监听门禁的协议，通过测试可以得到其中开门命令协议，在这里以数组的形式给出，第一个数组为打开一号门的命令，第二个数组为打开二号门的命令，第三个数组为打开三号门的命令.

```
char    g_CmdArray[4][100] =
{
    {2, 101, 44, 0, 1, 1, 0, 1, 74, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
    {2, 94, 44, 0, 2, 1, 0, 2, 113, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
    {2, 103, 44, 0, 3, 1, 0, 3, 72, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
    {2, 77, 44, 0, 1, 1, 0, 1, 98, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0}
};
```

在这里顺便给出通信类的代码， 在连接门禁的时候需要知道门禁的 IP 地址，例如“192.168.0.15”，以及门禁的端口，例如 8000。门禁通信的类代码如下：

DoorSocketServer.h 的代码:

```
#pragma once
#include     <Winsock2.h>
#include     <stdio.h>
typedef enum _doorindex
{
    DOOR_1 = 0,
    DOOR_2 = 1,
    DOOR_3 = 2,
    DOOR_4 = 3,
}DOOR_INDEX;
class CDoorSocketServer
{
    public:
    CDoorSocketServer(void);
    CDoorSocketServer(char cArrayAddress[] ,int iPort);
    ~CDoorSocketServer(void);
    bool Connect();
    bool OpenDoor(DOOR_INDEX index);
    bool DoorOpen();
    bool Disconect();
    private:
    bool Initiate();
    private:
    WORD     wVersionRequested;
    WSADATA     wsaData;
    int     err;
    SOCKET     sockClient;
```

```
    SOCKADDR_IN      addrSrv;
    int m_iPort;
    char m_cArrayAddress[20];
    };
```

下面的这个函数的功能为连接门禁服务器，若连接成功返回 true,连接失败返回 false

```
bool CDoorSocketServer::Connect()
{
    addrSrv.sin_addr.S_un.S_addr=inet_addr( m_cArrayAddress);
    addrSrv.sin_family=AF_INET;
    addrSrv.sin_port=htons(m_iPort);
    int iConnect = connect(sockClient,(SOCKADDR*)&addrSrv,sizeof(SOCKADDR))
    if (iConnect)
    {
        return true;
    }
    return false;
}
```

4.2.2 向门禁服务器发送开门的命令

在连接完服务器之后,主要的工作就是向服务器发送开门和关门的命令,代码如下,其中参数 index 为门的序号，0-3 表示 1 号到 4 号门。若发送命令成功返回 true，发送命令失败返回 false。

```
bool CDoorSocketServer::OpenDoor(DOOR_INDEX index)
{
    int iSize = sizeof(g_CmdArray[index]);
    int iSend = send(sockClient, g_CmdArray[index],iSize,0);
    if (iSend == iSize)
    {
        return true;
```

```
    }
    return false;
}
bool CDoorSocketServer::Disconect()
{
    closesocket(sockClient);
    WSACleanup();
    return true;
}
```

4.3 本章小结

本章先介绍了本门禁系统的一些协议，架构，然后重点讲解了门禁客户端的控制部分，包括初始化协议，连接门禁服务器，以及开门的过程，且对一些代码进行详细讲解。

第五章人脸识别系统的实现

5.1 系统简介

5.1.1 系统硬件组成

人脸识别的硬件主要由这几个部分组成：台式电脑、高清红外网络摄像头，门禁系统，这几个部分的选择与配置如下：

（1）台式电脑配置：CPU 为 i7-3770、主频为 3.4GHZ，内存为 4G、操作系统为 WIN7，在这里选择台式电脑配置比较高，主要考虑到对于高清摄像头传递回来的图片像素较高，对单张图片的处理时间增长，所以利用强一点的 CPU 来提高速度。

另外由于本系统为在线人脸识别系统，所以对实时性的要求也比较高，从时间上的推算来讲，要求在 1~2 之内识别出一个人脸，也就是说从处理到匹配的过程是非常短的，而且充分考虑到摄像头传递回来的图片由于人运动过快导致图片模糊无法识别的问题，以及由于人的姿势表情等的影响，也会造成识别率不高或者无法识别的问题。所以在这里对于处理速度的要求就更高了，所以在这里用了一个性能比较高的电脑。

（2）摄像机的选择：刚开始的时候选择的是普通 usb 摄像头，从处理速度上来讲，速度还是相当快的。一般的 USB 摄像头的像素为 30 万像素的样子，这样就算人坐在电脑面前进行人脸抓取识别的话，那么最终传递给系统的处理图片大小为 50*50 的样子，这种分辨率对于电脑而言是非常小的，而且搜索人脸的时候速度也会加快。但是问题也随之而来了，那就是识别率的问题，由于尺寸有限，导致特征信息不够多，这样在几个人之间区分识别虽然能够做到，但是人一多比如说 10 多个人那样系统就会识别错误，对于这样的识别系统是无法用的。

所以利用 USB 的普通摄像头是不行的，根据出现的问题，我们换成了另外一款专用来监测的工业摄像头，红外网络摄像头，品牌为 Anspo，型号为 ASP-524S70，这款摄像头在市场上的销量还是不错的，其像素为 200 万，8mm 广角镜头，这样就能将摄像头架设的比较高，和人的识别区域保持一定的距离，形成空间上的方便和优势，高清 200 万像素能够满足捕捉人脸信息不足的特点。经过测试，当人隔摄像头保持 3 米的距离的时候，捕捉到人脸的区域基本上还有 200*200 的样子，这样对于系统来讲像素足够多，而且又不是特别大。既能够满足处理速度的要求，又能够保证足够多的特征，提高识别的准确率。

（3）门禁系统的选择：门禁系统选择的是市场上比较常用的带多个网线接口的门

禁，在这里考虑的因素有两点，一个是由于摄像头本身是网络摄像头，而现在又需要和门禁实现通讯，所以如果门禁能够带有多个网口，那么就省去了路由，只需要将摄像头 IP 和门禁 IP 统一起来就可以，另外门禁系统的接口函数必须用 MFC 写成的，这样方便直接在软件中调用。

5.1.2 系统结构设计

本系统的结构主要由两个模块组成，即录入和识别，录入即采集样本存入样本相关的信息，是在识别之前必须做的一件事情可以离线完成，录入完成之后就是识别了，识别之后则可以将结果显示在界面上来，并能通知门禁控制大门的开关。其中录入的流程图如图 5 -1 所示，识别的流程图如 5 -2 所示。

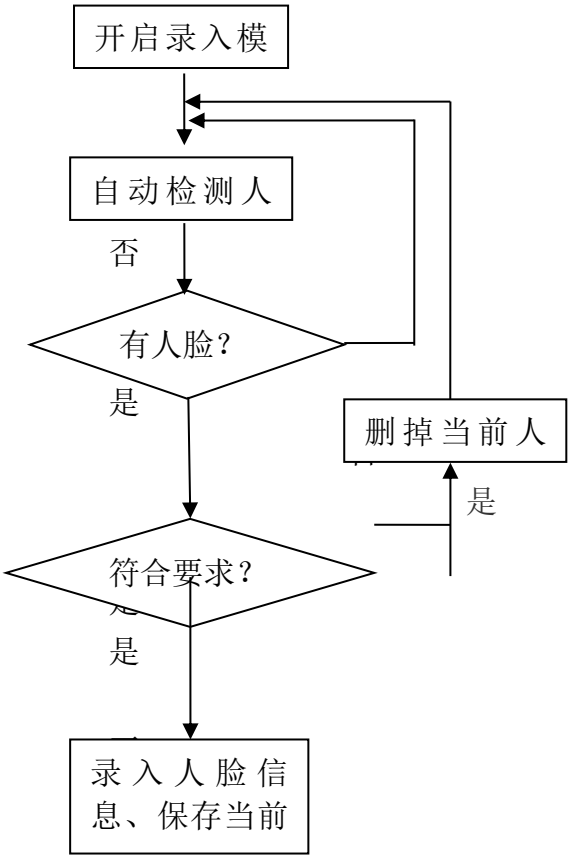

图 5 -1

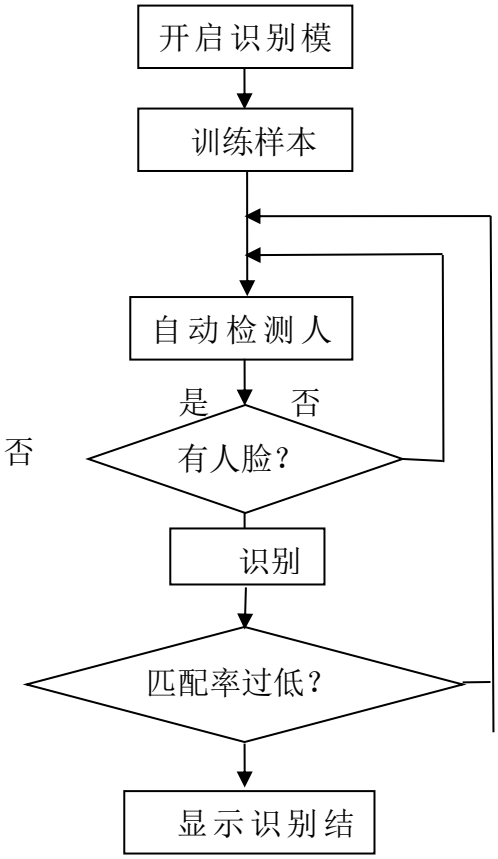

图 5 -2

5.2 系统实现

本系统围绕上一节中的录入和识别这两个主要模块进行了以下工作：

（1）利用网络摄像头的 SDK 开发包进行二次开发，并利用 MFC 做界面，将摄像头的视频显示到电脑上并做好接口函数供下一个录入功能提供样本图片。

（2）训练已经存在 Oracle 数据库中的图片，提取特征，加载到内存中，随时准备和实时图片特征进行匹配。

（3）在多线程中对实时数据和图片进行采集，进行处理，提取特征，然后与样本特征进行匹配，在界面上显示出结果。

（4）将匹配的结果通知门禁，门禁控制继电器进行开关操作。

5.2.1 开启网络摄像头

在摄像头的选用上，我们采用的是 200 万的高清网络摄像头，因此对于图像的传送需要用到产品的 SDK 包，进行二次开发，将代码移植到我的软件之中，并配置好相关文件，成功将视频在 MFC 的 Picture 控件上显示出来。其主要代码如下：

```
USES_CONVERSION;
//连接服务器
CString str=_T("admin");
CString port=_T("192.168.1.100");
DWORD dwError = JBNV_OpenServer(
        T2A(port),                  //地址
        8200,                       //端口
        T2A(str),                   //用户名
        T2A(str),                   //密码
        &m_hServer);                //如果成功，则返回一个此服务器连接的句柄
if(dwError)
{
    //ShowErrorMessage(dwError);
    AfxMessageBox(dwError);
    return;
}
JBNV_SetStreamCallBack(m_hServer,JBNVRecvStreamDataCallback,(DWORD)this);
if(m_hStreamPlay)
{
    JBPlay_Release(m_hStreamPlay);
    m_hStreamPlay = 0;
}
```

```
JBNV_SERVER_INFO si;
si.dwSize = sizeof(si);
JBNV_GetServerInfo(m_hServer,&si);
JBPlay_OpenStream(4,GetDlgItem(IDC_OPEN)->GetSafeHwnd(),FALSE,&m_hStream
Play);
WAVEFORMATEX wfx;
memset(&wfx,0,sizeof(wfx ));
wfx.nChannels = 1;
wfx.nSamplesPerSec = 8000;
wfx.wBitsPerSample = 16;
wfx.wFormatTag = 0x003E;
wfx.nAvgBytesPerSec = 4000;
JBPlay_SetAudioFormat(m_hStreamPlay,&wfx);
JBPlay_SetAudio(m_hStreamPlay,TRUE);
AfxMessageBox(_T("连接成功"));
m_dwFrameCount = 0;
dwError=JBNV_OpenChannel(m_hServer,0,0,0,0,GetDlgItem(IDC_OPEN)->GetSafeHw
nd(),FALSE,0,&m_hChannel);
JBNV_SetVideoCallback(m_hChannel,myVideoCallback,(DWORD)this);
JBNV_Mute(m_hChannel,FALSE);
JBNV_CHANNEL_INFO jbci;
jbci.dwSize = sizeof(jbci);
JBNV_GetChannelInfo(m_hServer,0,&jbci);
```

5.2.2 训练 Oracle 数据库中的图片

在这个系统中，由于是对人脸进行实时识别的，所以对于速度的要求就比较高，因此在这里我的做法是先进行一些训练和预处理，把存在数据库中的一些样本图片进行处理，计算特征码，以及将人的一些信息比如说名字、工号、部门等加入内存中，这样当人脸识别系统开启之后，只需要做一个匹配和查询工作就能够得到所匹配的图片，从而将识别结果显示出来。

5.2.3 录入模块的实现

对于录入部分的一些功能，是充分考虑到一些现实因素的。从现阶段的应用来看，

人脸识别一般应用于企业或者小区的人员管理，本系统以企业为例，录入部分由以下几个部分组成，部门、工号、姓名、采样图像。

录入部分的界面如图 5 -3 所示：

图 5 -3

录入模块的作用是存储样本信息，为识别做准备，对于录入模块又可以分解成三个部分：捕捉图像、样本处理、存储信息。

（1）捕捉图像

捕捉图像这一功能是通过点击捕捉按钮来实现的，在这里，为了能提高在不同表情或者角度的情况下的准确率，在捕捉的过程中，需要连续捕捉 5 种不同表情或者角度的人脸图像。存到 vector 向量中，如果样本不符合要求，那么可以点击删除按钮来进行删除，同样当不小心点击多了的话，也可以点击删除按钮进行删除。另外如果需要连续录入多个样本的话，也可以将当前 vector 中的样本全部删除，重新录入即可。

部分代码如下：

```
rec_flag=FALSE; //获取 roi
while(1)
{
    if(recdetect_flag)break;
    Sleep(10);
}
IplImage* img=NULL;
img=roi;
IplImage* imgface=NULL;
imgface=roiface;
rec_flag=TRUE;

IplImage *imageProcessed=NULL;
IplImage *imageProcessed1=NULL;
```

```
try
{
    imageProcessed1=cvCreateImage(cvSize(200,200),IPL_DEPTH_8U,1);
    imageProcessed= cvCreateImage(cvSize(80, 70), IPL_DEPTH_8U, 1);
    cvResize(img, imageProcessed, CV_INTER_LINEAR);
    cvResize(imgface, imageProcessed1, CV_INTER_LINEAR);
    picsample.clear();
    picsample.push_back(imageProcessed1);
    DrawPicToHDC(imgface, IDC_WRITE);
    imagesample.push_back(imageProcessed);
    CString str;
    str.Format(_T("%d"),imagesample.size());
    SetDlgItemText(ID_COUNT,str);
}
catch(cv::Exception& e)
{
    CString str;
    string rec;
    rec=e.msg;
    str=rec.c_str();
    AfxMessageBox(str);
}
```

删除按钮代码：

```
if(picsample.size()>0)
{
    picsample.pop_back();
    imagesample.pop_back();
    CString str;
    str.Format(_T("%d"),imagesample.size());
    SetDlgItemText(ID_COUNT,str);
}
```

（2）样本图像处理

从摄像头采集到的人脸区域显然是无法直接进行识别的。为了克服光照和不同尺寸图像对识别的影响，在存储样本图像之前需要做亮度归一化和尺寸归一化的处理。

其中亮度归一化的大概思想为，将采集到的图片无论强或者弱，都将其进行亮度归一化，在这里直接用到 opencv 中的函数来进行处理，方便又简单。另外之所以要做尺寸归一化的处理，是因为人离摄像头的距离并不是固定的，或者看摄像头的角度会有区别，系统捕捉到的人脸区域就会有区别。显然不同尺寸的图片从像素的角度去匹配是不可能的。电脑在这一点是没有人类智能的。在这里基于摄像头的像素和一些客观因素，取归一化后的尺寸为 200*200。

部分代码如下：

```
IplImage* imageProcessed = cvCreateImage(cvSize(200, 200), IPL_DEPTH_8U, 1);
cvResize(&image11, imageProcessed, CV_INTER_LINEAR);
roiface=imageProcessed;

//show AAM, and do some error handle
image1=imageProcessed;
bool flag =    AAM_model.InitShapeFromDetBox(Shape, facedet, image1);
if(flag == true)
{
    IplImage* imageroi=cvCreateImage(cvSize(80,70),IPL_DEPTH_8U,1);

    if(Draw(image1,imageroi,Shape))
    {
    cvEqualizeHist(imageroi, imageroi);
    roi=imageroi;
    Mat testImage(imageroi);
    }
}
```

（3）存储信息

在采集到人脸图像之后，接下来的一部分就是录入信息了。本系统采用 Oracle 数据库对数据进行管理，数据分别存在对应的几个表中，这样在捕捉完人脸之后只需要将对应的姓名、部门、工号等信息存储在相应的表中就可以了。

部分代码如下：

```
try
```

```
{
    if(imagesample.size()>0&&imagesample.size()<5)
    {
        Mat a=imagesample[0];
        int length=imagesample.size();
        for (int i=1;i<=5-length;i++)
        {
            imagesample.push_back(a);
        }
    }

    if( _mkdir( str ) == 0 &&imagesample.size()==5)
    {
        //Mat saveImage(imageGet);
        int countpic=countFile(_T("SamplePic"));
        countpic++;
        char strpic[100];
        sprintf(strpic, "SamplePic\\%d.bmp", countpic);
        CString strPathDlg(strpic);
        if(picsample.size()>0)
        {
            imwrite(format("%s",strpic),picsample[0]);
        }

        CString strNameDlg;
        GetDlgItemText(IDC_EDIT1,strNameDlg);
            //part
        CString strDepartmentDlg;
        GetDlgItemText(IDC_PART,strDepartmentDlg);
            //ghao
        CString strSnDlg;
        GetDlgItemText(IDC_GHAO,strSnDlg);
        AddEmployeeInfo(strNameDlg,strSnDlg,strPathDlg,strDepartmentDlg);
```

```
            for(int i=1;i<=5;i++)
            {    imwrite(format("%s\\%d.bmp",str,i),imagesample[i-1]);
                 CString strw;
                 char stt[1024]="";
                 sprintf(stt, "at\\s%d\\%d.bmp\n", count,i);
            //添加工号图片到数据库
                 AddSnPicInfo(strSnDlg, strw);
            }
    }
    Catch()
    {
    }
```

5.2.4 识别模块的实现

在将样本录入之后就能够进行在线识别了，在前几节中讲过，对于样本图片的录入部分，我是从一个专门的处理图片、捕捉图片的线程得来的，在这里，识别图片的来源同样来自那个线程，这样做的好处就是无论处理速度怎么样，识别速度怎么样，都不会影响主线程的运行，从效果上来看，实时监控的部分一直是保持流畅的。识别模块的界面如图 5 -4 所示。

图 5 -4

具体识别流程如下：

（1）从线程中得到在线人脸图片，进行尺寸归一化和亮度归一化处理，提取特征，然后利用训练好的模型对其进行降维和特征的进一步处理。在我的处理中，最终得到的是和样本保持一致的 16988*1 维信息。这样就能够进行距离的比较和显示人脸区域了。

（2）在捕捉到人脸之后，一个重要的步骤就是将人脸在实时监控中画出来，在这里我用的办法是用一个绿色的框来画出这是当前捕捉到的人脸。这样就能看算法对捕捉的准确性能了。

而在界面上实时画出人脸区域其实不是一件简单的事情，具体的方法可以分为两种，刚开始的时候尝试的是利用 opencv 对一张图片进行处理。对 yuv 视频数据进行转换之后变成 opencv 中比较常用的 IplImage 类型，然后利用 opencv 中的函数在图片中画出一个绿色的框来，再将其实时加载到监控画面中去，这样子虽然能够将效果显示出来，但是因为处理需要一些时间，而实时监控的刷新速度又比较快，所以看起来的效果就是捕捉到的人脸框总是有些滞后，造成卡顿的现象，后来调整方案，利用 MFC 的画图工具直接获取相关区域，然后直接在区域上画出矩形框来，这样子就和图片分开来，提高了处理的速度，显示效非常理想。

（3）将处理后的特征和样本特征进行比对，刚开始的时候，采用的是欧式距离直接对一些具体的值进行比对，效果还是可以的。但是后来进行改进，先对特征模块进行直方图处理，再直接利用 opencv 的函数进行直方图的比对，效果明显提高，速度也明显加快。

（4）显示匹配结果，从软件的角度上来讲，当匹配值低于设定的某一个值的时候，就认为匹配成功，但是此时得到的只是图片所在的一个编号，此时需要利用这一个编号将存在 Oracle 数据库中的对应的名字、工号等信息全部加载出来，当然充分考虑到速度的要求在软件刚开始运行的时候，就已经将着这些信息全部加载到内存中来了。提高了显示速度。

在显示的部分，主要显示的是名字、部门、工号、匹配率、样本图片。在这里，匹配率的作用主要还是给一个参考的作用，可以告诉保安人员，当前的匹配率是多少，然后样本图片是为了将当前人脸和样本进行对比，有没有识别准确一目了然，相当方便。

最后将匹配的结果通知门禁，来控制继电器的开和关。

线程中的代码如下：

```
void ThreadFunc()                    //detect face thread
{
    //cvNamedWindow("11");
    // init some value
    int nWidth,nHeight;
    double total,aa;
    int ms;
    int screenHeight=GetSystemMetrics(SM_CYSCREEN);
    int screenWidth=GetSystemMetrics(SM_CXSCREEN);
```

```
while(1)
{
    IplImage* image=NULL;
    IplImage* inputImg=NULL;
    IplImage* image1= NULL;
    /*IplImage* m_Frame;
    m_Frame=cvQueryFrame(capture);*/
    while(1)    //
    {    if(get_flag)break;
        Sleep(50);
        getdetect_flag=TRUE;
    }
    getdetect_flag=FALSE;

    huidiao_flag=FALSE; //c;//get yuv data
    while(1)
    {    if(detect_flag)break;
        Sleep(20);
    }
    //char* ppBufGet=pBufGet;
    /*unsigned char*ppBufGet=(unsigned char*)pBufGet;
    nWidth=pWidth;
    nHeight=pHeight;*/
    nWidth=Jwidth;
    nHeight=Jheight;
    unsigned char*ppBufGet=(unsigned char*)JBufGet;
    huidiao_flag=TRUE;

    image=YUV420_To_IplImage(ppBufGet, nWidth, nHeight);
    //try
    //{
    if((*image).height>0&&(*image).width>0)
    {
```

```
if (image->nChannels == 3)
{
    inputImg = cvCreateImage( cvGetSize(image), IPL_DEPTH_8U, 1 );
    cvCvtColor( image, inputImg, CV_BGR2GRAY );
    //t = (double)cvGetTickCount();
    faceRect= detectFaceInImage(inputImg, faceCascade);

    if(faceRect.width>=200&&faceRect.width<350)//200
    {   // draw rectangle on the face
        CvPoint pt1;
        CvPoint pt2;
        int xnow=cvRound(faceRect.x-faceRect.width*0.2);
        int ynow=cvRound(faceRect.y);
        if(xnow>0&&ynow>0)
        {
            faceRect.x=xnow;
            faceRect.y=ynow;
            faceRect.width=cvRound(faceRect.width*1.4);
            faceRect.height=cvRound(faceRect.height*1.2);
            //int recwidth=cvRound(faceRect.width*9/7);
            //int recheight=cvRound(faceRect.height*9/7);
            pt1.x=faceRect.x;
            pt1.y=faceRect.y;
            pt2.x=faceRect.x+faceRect.width;
            pt2.y=faceRect.y+faceRect.height;
            Mat mtx(inputImg);
            Matroi1(mtx,Rect(faceRect.x,faceRect.y,faceRect.width,
            faceRect.height));
            IplImage image11=roi1;
            //roi=&image11;

            IplImage* imageProcessed = cvCreateImage(cvSize(200, 200),
            IPL_DEPTH_8U, 1);
            cvResize(&image11, imageProcessed, CV_INTER_LINEAR);
```

```
            roiface=imageProcessed;

            //show AAM, and do some error handle
            image1=imageProcessed;
    bool flag =    AAM_model.InitShapeFromDetBox(Shape, facedet, image1);
        if(flag == true)
        {
        IplImage* imageroi=cvCreateImage(cvSize(80,70),IPL_DEPTH_8U,1);

        if(Draw(image1,imageroi,Shape))
        {
        cvEqualizeHist(imageroi, imageroi);
        roi=imageroi;
        Mat testImage(imageroi);
        predict1=lbph.predict(testImage,mindist);

        //open door
        if(mindist<acrate)
        {
        g_pCDoorSocketServer->OpenDoor(DOOR_1);

        //int ghao;
        //ghao=labels[predict1-1];//predict1*5-1
        CString strghao ;
        strghao.Format(_T("%d"),predict1);
        //CString strtime;
        CTime time;
        time=CTime::GetCurrentTime();
        //strtime=time.Format("%Y%M%D");
        char* folder="RecPic\\";
        char path[1024];

        sprintf(path,"%s%Y%m%d%X.bmp",folder,time);
        cvSaveImage(path,imageProcessed);
```

```
        CString strSn(strghao);
        CString strpath(path);
        //AfxMessageBox(strghao);
        AddVisitRecord(strSn,strpath);//L"data\\1.bmp"
        ShowEmployeeToUI(strSn);;



        Sleep(5000);
        }

        }
        while(1)                //acquire roi data and pass to recognition
        {    if(rec_flag)break;
        Sleep(20);
        recdetect_flag=TRUE;
        }
        recdetect_flag=FALSE;
        cvReleaseImage(&imageroi);

}

CPen pen;
pen.CreatePen(PS_SOLID,5,RGB(0,255,0));
::SelectObject(pDC->GetSafeHdc(),pen);
pDC->SetBkMode(TRANSPARENT);
aa=(faceRect.x*rect.Width())/screenWidth;
int x=floor(aa);
aa=(faceRect.y*rect.Height())/screenHeight;
int y=floor(aa);
aa=(faceRect.width*rect.Width())/screenWidth;
int width=floor(aa);
aa=(faceRect.height*rect.Height())/screenHeight;
int height=floor(aa);
pDC->MoveTo(x,y);
```

```
                pDC->LineTo(x+width,y);
                pDC->LineTo(x+width,y+height);
                pDC->LineTo(x,y+height);
                pDC->LineTo(x,y);
                pen.DeleteObject();
                cvReleaseImage(&imageProcessed);
                }
            }
                cvReleaseImage(&inputImg);
        }
    }
                if(image)cvReleaseImage(&image);
    }
    }
```

5.3 系统测试

系统设计与实现完成之后我对系统进行了测试，测试分为几个部分，用来评价在线识别的好坏以及影响识别的一些因素。

5.3.1 室内正常模式下测试

人脸识别系统搭建完毕之后，第一个实验测试场景就是在实验室了，实验室全天开启灯光，受外面太阳直射影响较小，光线比较均衡，环境总的来说还是比较理想的。在这样的条件下，录入实验室 10 个人的信息，进行在线识别，在合理设置匹配值以后，全部能够进行，再关掉系统，在早上、中午、晚上这几个时候进行测试。发现准确率还是比较高。

5.3.2 室内变化表情和环境光线测试

在这个测试环节，首先让测试人员利用不同于平常的表情在电脑面前，让其捕捉人脸进行识别，可以发现，识别率还是比较低的，说明系统对于表情变化的影响还是比较严重，这一点有待改进，再改变室内光线，在关掉全部灯光录入样本，然后开启全部灯光进行识别的时候，测试发现，准确率还是比较低的。

5.3.3 室外条件下测试

将系统搬到门口将摄像头架在门外，在中午太阳光较强的时候进行样本录入和测试，发现准确率还是比较高，关掉系统，当太阳下山的时候，开启系统，利用中午的样本进行测试，发现准确率比较低。

5.3.4 测试结论

经过试验表明，在线识别系统稳定，在室内环境下，识别准确率还是比较高的，利用实验室的 10 个人员进行测试，全部能够进行准确识别，但是对表情和比较大的光线变化还是比较敏感，有待加强。

在室外环境下，在同一时间段的识别效果还是比较理想的，但是同样当光线变化变化比较大，比如中午和傍晚这两个阶段，准确率对比之下就比较低。

所以本系统在对表情和光线这两个部分的鲁棒性还是有待加强的。

5.4 本章小结

本章主要介绍人脸识别的实现部分，首先介绍了系统的硬件组成，接着介绍了系统的整个设计结构和运行流程，然后着重介绍了系统的两个重要模块——录入和识别，再分别详细阐述了录入和识别这两个部分的实现过程，最后对系统进行了测试，表明在线人脸识别别系统能够稳定运行，但是对于表情和光线这两个部分的鲁棒性还是有待加强。

第六章总结与展望

6.1 总结

经过这么长时间的努力，人脸识别系统总算是顺利运行起来了，我觉得很高兴也确实不容易。在接手该项目时，我对这个项目还充满信心，在项目开始后不久，遇到的问题让我十分的头痛。为了解决这些问题不断地去查找资料，以及相关的人员和老师，使用了适当的软件工程方法，最终使项目可以顺利地进行下去。

记得刚开始的一个问题是摄像头 SDK 开发包的问题，拿过来之后很大一堆的代码，而且关键是给的 Demo 还不能用，存在 BUG，这样我就一段一段的去调试，最后花了好几天才搞定，最后看到成功显示出视频时，我笑了，真的体会到搞软件不容易，但是却那么让人心情澎湃。

在将系统搭建完毕，顺利运行之后，我对系统进行了一些测试，表明系统能够稳定运行，在条件良好的情况下识别率还是比较高的，但是在表情或者光线变化比较大时，识别率就比较低，说明人脸识别的算法还是有待进一步改进。

6.2 展望

我的测试结果表明，本系统在实际应用中仍然面临困难，当然我想这也是人脸识别这一领域同样遇到的难题，因此要想不仅要达到准确、快速的检测并分割出人脸部分，而且要有效的变化补偿、特征描述、准确的分类的效果，还需要注重和提高以下几个方面：

(1) 人脸的局部和整体信息的相互结合能有效地描述人脸的特征，基于混合模型的方法值得进一步深入研究，以便能准确描述复杂的人脸模式分布。

(2) 多特征融合和多分类器融合的方法也是改善识别性能的一个手段。

(3)由于人脸为非刚体性，人脸之间的相似性以及各种变化因素的影响，准确的人脸识别仍较困难。为了满足自动人脸识别技术具有实时要求，在必要时需要研究人脸与指纹、虹膜、语音等识别技术的融合方法。总之，人脸识别是极富挑战性的课题仅仅采用一种现有方法难以取得良好的识别效果，如何与其它技术相结合，如何提高识别率和识别速度、减少计算量、提高鲁棒性如何采用嵌入式及硬件实现，如何实用化都是将来值得研究的。

参考文献

[1] Chellappa R,Wilson C,Sirohey S. Human and machine recognition of faces:A survey[A].Proceedings of the IEEE,1995,(5):705-740.doi:10.1109/5.381842.

[2] Shakhnarovich G, Fisher J W, Darrell T. Face recognitionfrom long2term observations Proceedings of the EuropeanConference on Computer Vision. Bari , 2002 : 8512868

[3] Liu X M , Chen T , Thornton S M. Eigenspace updating fornon2stationary process and its application to face recognition.Pattern Recognition , 2003 , 36(9) : 194521959

[4] Liu X M , Chen T. Video2based face recognition using adaptive hidden Markov models Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition. Madison , 2003 : 3402345

[5] Lee K C , Ho J , Yang M H , Kriegman D. Video2based facerecognition using probabilistic appearance manifoldsProceedings of the International IEEE Conference on ComputerVision and Pattern Recognition. Madison , 2003 : 3132320

[6] Chellappa R,Wilson C,Sirohey S. Human and machine recognition of faces:A survey Proceedings of the IEEE,1995,(5):705-740.doi:10.1109/5.381842. [10] Choudhury A,Clarkson B,Jebara T,Penland A. Multimodal person recognition using unconstrained audio and video[A].Washington,DC,1999.176-180.

[7] Jing X Y,Yao Y F,Zhang D,Yang J Y,Li M. Face and palmprint pixel level fusion and kernel DCV-RBF classifier for small sample biometric recognition[J].Pattern Recognition,2007,(11):3209-3324.doi:10.1016/j.patcog.2007.01.034.

[8]Yah Y,Zhang Y J. Multimodal biometrics fusion using correlation filter bank[A].Tampa,2008.

[9] Zhou S,Chellappa R. Beyond a single still image:Face recognition from multiple still images and videos[M].New York:Academic Press,Inc,2005.

[10] Shakhnarovich G,Fisher J W,Darrell T. Face recognition from long-term observations[A].Bari,2002.851-868.

[11] Liu X M,Chen T. Video-based face recognition using adaptive hidden Markov models[A].Madison,2003.340-345.

[12] Lee K C,Ho J,Yang M H,Kriegman D. Video-based face recognition using probabilistic appearance manifolds[A].Madison,2003.313-320.

[13] Zhou S,Chellappa R,Moghaddam B. Visual tracking and recognition using appearance-adaptive models in particle filters[J].IEEE Transactions on Image Processing,2004,(11):1434-1456.

[14] Aggarwal G,Chowdhury A K R,Chellappa R. A system identification approach for

video-based face recognition[M].Cambridge,2004.23-26.

[15] Arandjelovi(c) O,Cipolla R. Face recognition from face motion manifolds using robust kernel resistor-average distance[A].Washington,DC,2004.88-93.

[16] Jia H X,Zhang Y J. Human detection in static images[A].2008.227-243.

[17] Liu X M,Zhang Y J,Tan H C. A new Hausdorff distance based approach for face localization[J].Sciencepaper Online,2005,(1-9).

[18] Yamaguchi O,Fukui K,Maeda K. Face recognition using temporal image sequence[A].Nara,1998.318-323.

[19] Fukui K,Yamaguchi O. Face recognition using multi-viewpoint patterns for robot vision[A].Siena,Italy,2003.192-201.

[20] Nishiyama M,Yamaguchi O,Fukui K. Face Recognition with the multiple constrained mutual subspace method[M].New York,2005.71-80.

[21] Li J W,Wang Y H,Tan T N. Video-based face recognition using a metric of average Euclidean distance[A].广东广州,2004.224-232.

[22] Li J W,Wang Y H,Tan T N. Video-based face recognition using earth mover's distance[M].New York,2005.229-239.

[23] Fan W,Wang Y H,Tan T N. Video-based face recognition using Bayesian inference model[M].New York,2005.122-130.

[24] Yah Y,Zhang Y J. State-of-the-art on video-based face recognition[J].Encyclopedia of Artificial Intelligence,2008.1455-1461.

附件 1：硬件开发清单

附件 2：软件开发源码

附件 3：实际应用视频

见光盘