****

**Southwest university of science and technology**

本科毕业设计（论文）

门禁监控视频软件设计与实现

|  |  |
| --- | --- |
| 学院名称 | 计算机科学与技术学院 |
| 专业名称 | 软件工程 |
| 学生姓名 | 张浩 |
| 学号 | 20111689 |
| 指导教师 | 刘畅 |

二〇-五年六月

门禁监控视频软件设计与实现

摘要：传统的门禁系统一般是无法实现门禁和视频在软件层面联动的，只能通过简单的硬件连接看到现场的实时监控录像，对非法闯入这一类可以进行报警，但对一些复杂情况，例如前面的人刷了名牌之后，后面的人在没有刷卡的情况下混入办公场所或者混出办公地点是无法进行检测预警的。

视频门禁系统则可以在人员刷卡、开门、按下出门按钮等动作和事件或者报警产生时，联动视频监控系统对现场的照片和录像进行抓拍并保存在门禁服务器中，同时还可以及时在软件界面上弹出对应的现场监控画面。这样不仅让照片、录像和对应的门点事件绑定在一起，可以随时通过查询将现场情况进行再现，还可以确保系统内的警情得到及时确认和处理。

基于WPF的门禁视频监控系统，采用C#与EmguCV的技术，是实现对人员出入门禁系统时进行人头检测。论文主要分析了人头样本数据的建立与人头的检测的AdaBoost和SVM算法，设计出一个可以检测人头的门禁监控视频。

关键词：视频监控； 人头检测； EmguCV

**Access control monitoring of video software Design and Implementation**

**Abstract**：Traditional access control systems are generally unable to achieve the level of access control and video software linkage, it can connect in real-time surveillance video to see the scene through simple hardware, for unlawful entry in this category may be the police, but for some complex situations, such as after the brand name in front of people brush the back of people without credit card or mixed with a mix of office space office is unable to detect warning.

. Video Control System personnel card can open the door, press the exit button and other actions and events or alarm occurs, the video surveillance system linked to the scene of the photos and videos were captured and stored in the access server, and timely in the software interface on the pop scene corresponding monitor screen. This will not only allow pictures, videos and a corresponding gate point events bound together, you can always query the scene reproduction, and to ensure that alarm system has been identified and addressed in a timely manner.

WPF-based access control video surveillance systems, using C # and EmguCV technology is to achieve when people head out access system testing. Paper analyzes the establishment of a human head detection algorithm AdaBoost and SVM poll sample data, designed to detect a surveillance video access control head.

**Key words:** Video Monitor, Head Detection, EmguCV

目 录

[第1章 绪 论 1](#_Toc420954550)

[1.1 选题背景和意义 1](#_Toc420954551)

[1.2 视频监控研究现状 1](#_Toc420954552)

[1.3 本文研究内容和章节安排 2](#_Toc420954553)

[第2章 视频监控系统中的关键技术 4](#_Toc420954554)

[2.1图像处理技术 4](#_Toc420954555)

[2.2 WPF 4](#_Toc420954556)

[2.3 人头检测技术 5](#_Toc420954557)

[2.3.1 AdaBoost检测算法 5](#_Toc420954558)

[2.3.2 SVM检测算法 6](#_Toc420954559)

[2.3.2 视频图像及样本采集及训练流程 9](#_Toc420954560)

[2.4 门禁视频监控系统工作流程 11](#_Toc420954561)

[2.5 本章小结 12](#_Toc420954562)

[第3章 视频监控系统设计与实现 13](#_Toc420954563)

[3.1 视频监控系统的需求分析 13](#_Toc420954564)

[3.1.1 系统目标 13](#_Toc420954565)

[3.1.2 功能需求 13](#_Toc420954566)

[3.1.3 功能分配 14](#_Toc420954567)

[3.1.4 性能需求 15](#_Toc420954568)

[3.2 系统开发环境介绍 15](#_Toc420954569)

[3.3系统概要设计 16](#_Toc420954570)

[3.3.1 总体设计 16](#_Toc420954571)

[3.3.2 辅助模块设计 16](#_Toc420954572)

[3.3.3 人头检测模块 17](#_Toc420954573)

[3.3.4 显示模块 18](#_Toc420954574)

[3.3.5 接口设计 18](#_Toc420954575)

[3.3.6维护设计 19](#_Toc420954576)

[3.4 系统详细设计 19](#_Toc420954577)

[3.4.1 辅助功能模块详细设计 19](#_Toc420954578)

[3.4.2 人头检测模块详细设计 20](#_Toc420954579)

[3.4.3 各部分类图 21](#_Toc420954580)

[3.5 系统实现 23](#_Toc420954581)

[3.5.1 样本数据加载 23](#_Toc420954582)

[3.5.2 AdaBoost算法人头检测 25](#_Toc420954583)

[3.5.3 SVM算法人头检测 26](#_Toc420954584)

[3.5.4 视频播放实现 30](#_Toc420954585)

[3.6 系统效果演示 35](#_Toc420954586)

[3.7本章小结 38](#_Toc420954587)

[第4章 实验结果对比分析 39](#_Toc420954588)

[4.1 AdaBoost和SVM算法效果对比 39](#_Toc420954589)

[4.2结论分析 40](#_Toc420954590)

[参考文献 42](#_Toc420954591)

[致谢 43](#_Toc420954592)

第1章 绪 论

1.1 选题背景和意义

随着视频监控技术的迅速发展与设备成本的不断下降，视频监控已经遍布于城市的各个角落成为了一种重要的视频信息来源。目前，视频监控系统在银行的安全防范、道路的交通管理、公安刑事侦查等特殊领域已经发挥了及其重要的重用。由于上述领域对于实用的监控视频采集、处理与分析技术和应用表现了十分迫切的需求，各类监控视频信息处理相关的基础理论研究与应用系统的设计开发也越来越受到政府、科研院以及公司的重视。

目前使用的监控系统系统大多只具有录像功能，因此只能提供事后查询，并不能对当时情况作出及时判断并提出警告，例如门禁监控，传统的门禁系统往往将门禁前的视频图像发到视频墙上，由人工监视什么时间人员可以进出，进出是否刷卡，是否有异常情况发生。所以有很多不足产生：

1. 工作强度大，持续时间长，容易对负责浏览视频的工作人员的视觉会造成很大的影响，危害身体健康。
2. 监控成本上，人力成本的持续投入使得成本不断增加。
3. 长时间高强度、高压力的工作状态下，负责视频浏览的工作人员注意力难以集中，容易出现一些重要信息的遗漏和误判现象。
4. 门禁系统刷卡与视频无关联，在突发情况下，监控人员很难判断出发生的原因。

为了解决现阶段门禁的不足。本文设计出基于门禁的视频监控软件系统，在本系统中实现了门禁硬件与视频监控的交互。减轻监控人员的负担，记录异常情况的发生和截图。

1.2 视频监控研究现状

目前美国以及欧洲地区等西方先进国家仍然掌握着大部分智能视频监控系统的核心算法技术,在国际市场上的智能视频监控系统中占据很大的优势。美国国防部高级研究项目署（Defense advanced research projects agency , DARPA）于1997年建立了视觉监控重大项目VSAM(Visual surveillance and monitoring), 该项目对用于民用场景和战场中视频监控的自动理解技术进行了重点研究。马里兰大学研发成功的实时视频监控系统W4主要任务是是被监控场景中的行人，将形状分析和目标跟踪技术相结合，对人的外表进行模型构建。高级研究计划VACE(Video Analysis and Content Extraction)于2003年由美国ARDA机构开始主持。在对车辆和行人的跟踪及其交互作用识别的相关研究方面上，英国雷丁大学（University of Reading）取得了一系列的研究成果。以色列也在智能视频监控领域取得了很大的研究成果，他们的产品主要用在边界不含以及商业调查分析领域[10-11]。

在对智能视觉监控领域的研究方面，国内起步比较晚，目前，中国科学院自动化研究所模式识别国家重点实验室在国内的研究机构中处于领先地位，取得了一系列研究成果。除此之外，国内还有一些高校也在这一领域投入了相当的研究精力，如上海交通大学、清华大学、华中科技大学、北京航空航天大学、北京理工大学等[3]。

1.3 本文研究内容和章节安排

本文将监控视频作为研究对象，基于监控视频本身的特点，对监控视频摘要技术进行了研究。在门禁视频监控中，分析人进出门禁的情况，并进行统计某一时刻进出人员数量。

基于以上工作，将本文安排为四章，具体如下:

1. 绪论

简单介绍视频监控系统的研究背景，意义以及视频监控系统的现状和发展趋势，总结前人的基础上，结合项目实际应用阐述本论文的主要工作。

1. 视频监控中的关键技术

本章主要介绍了视频监控系统中应用到的一些关键技术，例如EmguCV图像处理、人头检测技术中的AdaBoost和SVM技术、C#编程其中的WPF编程技术、目标跟踪[12-13]等。

1. 视频监控系统软件设计与实现

本章主要从系统的需求分析开始。分析系统的需求，提出需要的功能，然后进行概要设计，划分出功能模块，与各个功能模块之间的接口，从而进行系统各模块的详细设计，在进行类图的设计最终完成各个功能实现本系统的开发

1. 实验结果对比分析

对系统中所使用的AdaBoost算法和SVM人头检测算法进行分析比较，得出各个算法的优势和劣势，从而为后期的系统优化做好准备。

第2章 视频监控系统中的关键技术

2.1图像处理技术

图像处理（Image Processing）,用计算机对图像进行分析，已达到所需结果的技术。又称影像处理。图像处理一般指数字图像处理。数字图像是指用工业相机、摄像机、扫描仪等设备经过拍摄得到的一个大的二维数组，该数组的元素称为像素，其值称为灰度值。图像处理技术一般包括图像压缩，增强和复原，匹配、描述和识别3个部分。

OpenCV[4]（ Open Source Computer Vision Library）是一个基于（开源）发行的跨平台计算机视觉库，由一系列C函数和少年C++类构成，可以运行在Linux、Windows和Mac OS操作系统上。它轻量级而且高效——由一系列 C 函数和少量 C++ 类构成，同时提供了Python、Ruby、MATLAB等语言的接口，实现了[图像处理](http://baike.baidu.com/view/14662.htm" \t "_blank)和计算机视觉方面的很多通用算法。

由于门禁视频监控系统在.NET平台下实现，不能直接运行C/C++类库，所以需要对OpenCV进行封装，所以在本论文中使用了EmguCV。

EmguCV是.NET平台下对OpenCV图像处理库的封装。也就是OpenCV的.NET版。它运行在.NET兼容的编程语言下调用OpenCV的函数，如C#等。

2.2 WPF

WPF（Windows Presentation Foundation）是微软推出的基于Windows Vista的永华界面框架，属于.NET Framework3.0的一部分。它提供了统一的编程模型（MVVM模型）、语言和框架，真正做到了分离界面设计人员与开发人员的工作；同时它提供了全新的多媒体交互用户图形界面。

WPF具有强大的数据绑定功能，存储和访问数据由Microsoft SQL Server和ADO.NET之类的技术提供，涉及两个步骤：

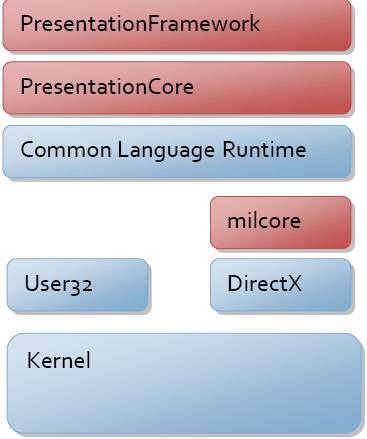
1. 将数据从托管对象复制到控件中，在控件上可以显示和编辑数据。
2. 确保将使用控件对数据进行的更改复制回托管对象。

为了简化应用程序开发，WPF 提供了一个数据绑定引擎以自动执行这些步骤。数据绑定引擎的核心单元是 [Binding](https://msdn.microsoft.com/zh-cn/library/system.windows.data.binding.aspx) 类，它的任务是将控件（绑定目标）绑定到数据对象（绑定源），关系如图所示：



WPF 主要编程模型是通过托管代码公开的。在 WPF 的早期设计阶段，曾有过大量关于如何界定系统的托管组件和非托管组件的争论。CLR 提供一系列的功能，可以令开发效率更高并且更加可靠（包括内存管理、错误处理和常规类型系统等）,但这是需要付出代价的。图2-1说明了 WPF 的主要组件.关系图的红色部分（PresentationFramework、PresentationCore 和 milcore）是 WPF 的主要代码部分。 在这些组件中，只有一个是非托管组件–milcore。

milcore 是以非托管代码编写的，目的是实现与 DirectX 的紧密集成。WPF 中的所有显示都是通过 DirectX 引擎完成的，因此硬件和软件呈现都很高效,WPF 还要求对内存和执行进行精细控制,milcore 中的组合引擎受性能影响关系大，需要放弃 CLR 的许多优点来提高性能。



**图2-1 WPF关系图**

2.3 人头检测技术

模板匹配是数字图像处理的重要组成部分之一。把不同传感器或同一传感器在不同时间、不同成像条件下对同一景物获取的两幅或多幅图像在空间上对准,或根据已知模式到另一幅图中寻找相应模式的处理方法就叫做模板匹配。支持向量机（Support Vectoy Machine,SVM）和Casacde结构的AdaBoost（Adaptive Boost）检测算法是目前较为流行的两种目标检测方法。

2.3.1 AdaBoost检测算法

人头检测可视为一个两类分类问题，其中一类为人头(正样本)，另一类为非人头(负样本)，假设 AdaBoost 学习算法的输入为训练样本集

LS = {( xi,yi ), i=1, 2,3,… ,N}

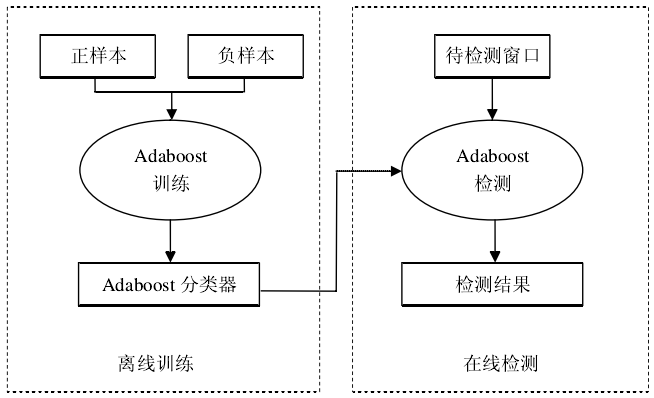
其中，Xi表示第 i个训练样本，yi表示样本类别，且yi∈(+1,−1),+1 表示正样本,−1表示负样本。因此，对于每个样本，在训练过程中都事先知道其样本类型，以便对算法做出的预测正确与否进行判断。

在算法训练阶段，首先对每个样本赋予相同的初始权重W ,之后进行K 轮训练，并且在每轮训练结束后根据当前分类结果调整训练样本的权重，减小正确分类样本的权重，同时增加错误分类样本的权重，从而使每个弱分类器主要围绕难以正确分类的样本进行学习。经过K 轮训练得到一个由K 个弱分类器(每个弱分类器对应一个特征)组成的弱分类器序列

H={(h1,α1), (h2,α2), (h3,α3), …, (hk,αk)}

其中，hi表示第 i 个弱分类器，αi为弱分类器的加权系数。最后对K 个弱分类器进行组合，得到最终的强分类器H 。虽然每个弱分类器hih 的分类预测能力只比随机稍好，但通过对K 个弱分类器进行组合，能够使强分类器H 获得任意高的精度。

AdaBoost 算法流程如图 2-2 所示。算法主要由离线训练和在线检测两部分组成，在训练阶段首先对正负样本进行训练，从特征库中选取最优特征并组合成强分类器，在检测阶段，用训练好的分类器对各个待检测窗口进行检测。



**图2-2 Adaboost 算法流程**

2.3.2 SVM检测算法

假设训练样本集为LS = {(xi,yi),i = 1,2,…,n}，其中xi表示第i个样本，n为样本数，∈(+1,-1)为类别标志。对线性和非线性的情况如下:

1. 线性SVM

如存在分类超平面方程：

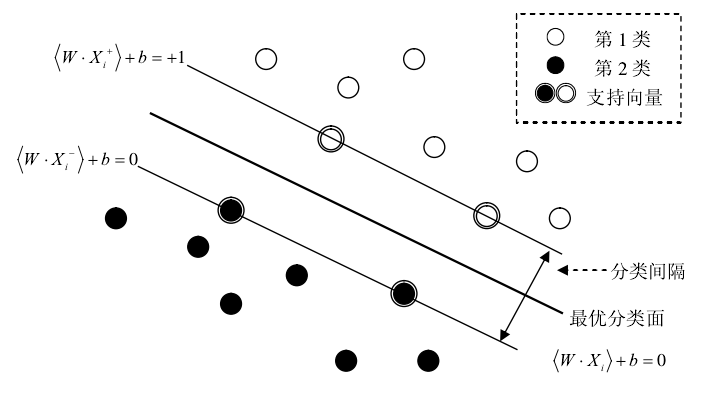
**(1)**

满足

**（2）**

则认为训练样本集是可分的。其中，<W ⋅ Xi>为向量W 与Xi的内积。

SVM 算法就是在这种线性可分情况下寻找最优分类面。如果一个分类面不但能将两类样本无错误地分开，并且到两类中离该分类面最近样本点的距离最大，则这个分类面就是最优分类面，如图 2-2 所示。



**图2-2 最优分类超平面**

从以上分析可知，线性判别函数的一般形式为 g ( X )= W ⋅ X + b，若W 为权重向量，且假设最小函数间隔为 1，即在正样本点 X+和负样本点 X−上使函数间隔为 1，即对于离分类面最近且平行于最优分类面上的样本点满足：

**（3）**

则分类间隔为:

**(4)**

其中，||W||2为向量W的几何范式。

这样，在两类中所有样本都满足|lg(X)|≥1.则最优分类超平面满足：

**(5)**

而求解最优分类超平面需要最大化分类间隔2/||W||2，即最小化||W||2，则问题转化为：

**（6）**

但是，当两类分类数据不能完全线性分开时，最大间隔将是负数。因此需要对公式（6）的约束条件进行调整：

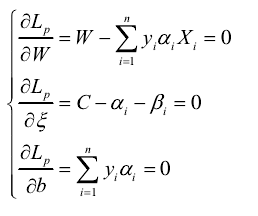
**（7）**

其中，C为惩罚系数，C越大表示对错误分类的惩罚越大。根据拉格朗日乘子算法，有：

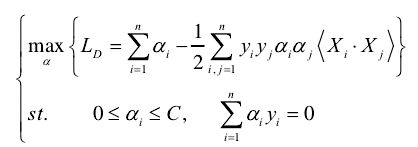
**（8）**

其中，αi≥0、βi≥0为拉格朗日乘子。

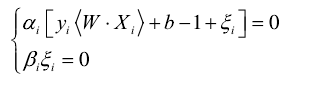
对W，ξ ,b求偏导，并置0，有：

 **（9）**

将式（9）带入公式（8）中，得到：

 **（10）**

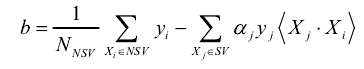
式中，0<αi<C对应标准支持向量（Normal Support Vector,NSV）,αi = C对应边界支持向量。由Kaeush-Kuhn-Tucker条件[]，在最优点，有：

 **（11）**

结合式（9）-（11）可计算出阀值参数b为

 **(12)**

假设共存在Nnsv个标准支持向量，则为了提高计算的准确性，对每个标准支持向量分别计算b，并取均值，从而得到最终阀值b为：

 **(13)**

2) 非线性SVM

非线性 SVM 与线性 SVM 的区别在于，非线性 SVM 是通过一个非线性函数 () 将样本X映射到一个高维线性特征空间，并在这个空间中构造最优分类超平面，并得到分类器。因此在该情况下，分类超平面为

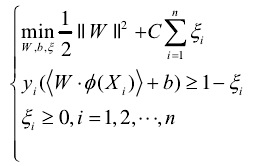
 **(14)**

其中，()为非线性函数

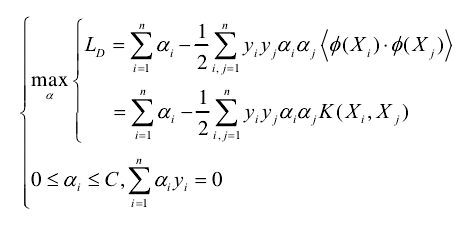
判别函数为：

**(15)**

最优分类超平面问题描述为

  **(16)**

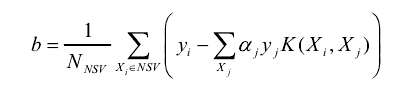
得到的对偶最优化问题为：

 **(17)**

其中，为核函数，判别式为

 **(18)**

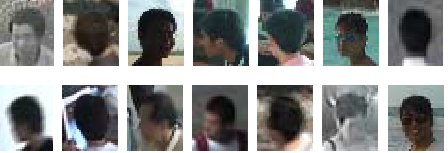
其中，阀值为

 **（19）**

2.3.2 视频图像及样本采集及训练流程

1） 样本采集

进行人头训练与检测需要大量的视频序列和训练样本。本文视频序列部分源自实际监控录像，部分采用Kinect拍摄，人头样本由手工截取，选择不同背景下的不同行人，并截取2000余张，人头样本一千余张，负样本为不包含人头的背景图片随机产生。为了方便训练，所有样本均归一化到相同尺度，本文为20 24像素大小。人头正、负样本示例如图2.2和图2.3所示。



**图2.2 人头正样本示例**



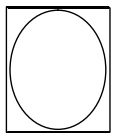
**图2.3负样本示例**

2） 训练流程

本论文训练人头的流程如下：

1. 准备训练样本集合；包括正样本集和负样本集；根据机器学习的基础知识我们知道，要利用机器学习算法进行样本训练，从而得到一个性能优良的分类器，训练样本应该是无限多的，而且训练样本应该覆盖实际应用过程中可能发生的各种情况。实际应用过程中，训练样本不可能无限多，本次使用了一千个正样本，一千个负样本
2. 收集到足够的训练样本之后，需要手动裁剪样本。本人使用的方法为利用收集到门禁视频行人进出大门的的视频作为训练样本集合，写个了简单程序，只需要鼠标框选一下，就将框选区域保存下来。
3. 裁剪得到训练样本之后，将所有正样本放在一个文件夹中；将所有负样本放在另一个文件夹中；并将所有训练样本缩放到同样的尺寸大小。OpenCV自带的例子在训练时，就是将样本缩放为64\*128进行训练的；
4. 提取所有正样本的Hog特征；
5. 提取所有负样本的Hog特征；
6. 对所有正负样本赋予样本标签；例如，所有正样本标记为1，所有负样本标记为0；
7. 将正负样本的Hog特征，正负样本的标签，都输入到SVM中进行训练；采用线性SVM进行训练。
8. SVM训练之后，将结果保存为文本文件。这样就可以利用训练样本训练出来的分类器进行人头检测了。

其中在AdaBoost特征选取上，考虑到人头普遍位于样本的中心位置，因此特征在样本中的位置进行一定的限制，使特征的各个子窗口不落在样本的四个角上，如图2.5所示，在生成特征库时，限制矩阵梯度和自生长梯度特征各特征子窗口全部位于图中的椭圆内，采用该方法一面可以尽可能减少背景梯度对特征 形成的干扰，另一方面可以减少特征库的规模，提高训练速度。



**图2.5特征位置限制示意图**

由于SVM算法是针对有限样本进行训练，参与训练的样本数不亦过大，否则会使训练变慢甚至无法完成。因此，本文在AdaBoost人头样本中随机挑选了四分之一，作为SVM训练的正样本，负样本由背景图随机产生，进行聚类。

2.4 门禁视频监控系统工作流程

门禁视频监控系统监控过程中，人头检测可以采用AdaBoost检测算法，也可采用SVM检测算法，虽然算法不同，但检测结果几乎接近。系统的工作包括以下几步：

1. 预处理

主要包括视频图像的加载、解析。

1. 加载样本数据

加载人头的样本数据，为检测做好准备

1. 进行人头模板检测

与样本数据进行模板匹配，获取视频图像中的人头

1. 人头校验

对检测到的人头进行校验，去除误检。

1. 进行跟踪

对检测到的人头进行跟踪

图2.4.1给出了门禁监控视频的大致流程

**图2.4.1**

2.5 本章小结

门禁视频监控的实现是建立在许多技术基础之上，本章主要介绍一些关键技术，例如使用Emgucv/OpenCV图像处理库，可以更方便的进行图像处理；如何生成样本数据以及使用模板匹配进行人头检测，可以快速检测出人头；使用WPF进行界面设计使代码与界面分离，减少耦合度；使用面向对象的编程技术，可以快速开发、减轻开发人员的压力。WPF其实不仅仅是图形[引擎](http://baike.baidu.com/view/53607.htm)而已，它将给Windows[应用程序](http://baike.baidu.com/view/330120.htm)的开发带来一次革命，因为新的架构提供了一种全新的开发模式。对于开发人员而言，界面显示和[代码](http://baike.baidu.com/view/41.htm)将更好的得到分离，这与从前的[桌面应用程序](http://baike.baidu.com/view/2251864.htm)开发有很多不同（界面设置和代码是融合在一起的），这是比较具有革命性的改变之一。

第3章 视频监控系统设计与实现

3.1 视频监控系统的需求分析

门禁软件已经广泛使用在企事业单位、工厂、公司等办公场所，职员，工人佩带名牌进入办公地点，通过名牌对出入口通道进行管制的功能对单位的组织纪律，以及安防都非常有帮助。

目前很多的门禁软件会加上对出入口通道视频监控来进一步进行监控，来对出入口通道进行更好的监控，传统的门禁系统一般是无法实现门禁和视频在软件层面联动的，只能通过简单的硬件连接看到现场的实时监控录像，而视频门禁系统则可以在人员刷卡、开门、按下出门按钮等动作和事件或者报警产生时，联动视频监控系统对现场的照片和录像进行抓拍并保存在门禁服务器中，同时还可以及时在软件界面上弹出对应的现场监控画面。这样不仅让照片、录像和对应的门点事件绑定在一起，可以随时通过查询将现场情况进行再现，还可以确保系统内的警情得到及时确认和处理。

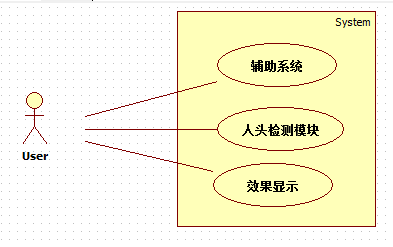
但是目前的视频门禁监控软件的处理的情况还是比较单一的，对非法闯入这一类可以进行报警，但是对一些复杂情况，例如前面的人刷了名牌之后，后面的人在没有刷卡的情况下混入办公场所或者混出办公地点，这种情况在一些安防级别比较高的单位也是不能够允许的，但是传统门禁视频监控软件却无法检测出这种情况。

3.1.1 系统目标

系统开发的主要目标是能够针对上述问题，利用图像处理，目标检测与跟踪，相机标定，透视投影变换等理论知识，设计出对视频图像进行处理的算法，来达到对视频中人员混入混出现象的检测，同时兼顾先进性、可靠性、安全性、经济型、可扩充性、可维护性和规范性做到一切应从实际出发，使监控系统具有较高的实用效能。本系统操作简单便捷，监控人员不需要经过严格复杂的培训技能即可上手使用。

3.1.2 功能需求

为了使用者能够方便使用此系统，系统功能还将包括辅助系统和效果显示系统。系统用例图如图3-1所示。



**图3-1 系统用例图**

辅助系统主要包括选择视频源，启用/禁用视频检测，视频检测算法，显示视频图像。用例图如图3-2所示。



**图3-2 辅助系统用例图**

在显示系统中，主要是显示未开启检测的视频图像和开启视频的视频图像，以及保存图像。图3-3显示了系统的基本用例。



**图3-3 显示系统用例图**

3.1.3 功能分配

根据功能需求部分所描述出的系统用例图，画出该系统主要功能，如下表3-1所示。

**表3-1 功能分配表**

|  |  |
| --- | --- |
| 模块 | 详细功能 |
| 辅助系统模块 | 辅助模块主要处理人头检测之前的操作。例如选择视频源，开启视频检测，添加/删除视频源等功能。 |
| 人头检测模块 | 通过AdaBoost算法或者SVM算法，进行人头检测，将检测到的数据再次进行分析，筛选符合的人头  将检测到符合的人头在图像上进行显示，提供保存图像等操作。 |
| 显示模块 |

3.1.4 性能需求

1. 精度

在显示检测图像中，人头检测错误率不超过2个。

1. 时间

本门禁视频监控系统视频延迟时间在2秒内。

1. 系统灵活性

本系统具有可扩充性，可维护性和规范性。

3.2 系统开发环境介绍

1. 硬件环境

本系统对计算机硬件要求不高，处理器基于X86或X64指令集均可，摄像头最低可为低解330线，拍摄画面清晰，没有用肉眼不可分辨的事物。

1. 软件平台

此系统主要运行在Windows操作系统下，推荐Windows XP及其以上版本。因为Windows XP系统兼容性，对新技术。新产品都有良好支持，是设计开发人员常用的操作系统。另外，Windows XP系统集成了微软的防火墙技术，保障了用户计算机使用安全。

1. 开发环境

Windows平台下选择Microsoft Visual Studio 2013开发工具,它是目前最流行windows平台应用程序开发环境[2]。Visual Studio 2013是一个基本完善的开发工具集它包括了整个软件生命周期中所需要的大部分工具，如UML工具、代码管控工具、集成开发环境(IDE)等等。所写的目标代码适用于微软支持的所有平台，包括[Microsoft Windows](http://baike.baidu.com/view/182749.htm)、[Windows Mobile](http://baike.baidu.com/view/40733.htm" \t "_blank)、[Windows CE](http://baike.baidu.com/view/41539.htm" \t "_blank)、[.NET Framework](http://baike.baidu.com/view/18370.htm" \t "_blank)、[.NET Compact Framework](http://baike.baidu.com/view/1546355.htm" \t "_blank)和Microsoft [Silverlight](http://baike.baidu.com/view/942429.htm) 及[Windows Phone](http://baike.baidu.com/view/2708518.htm" \t "_blank)。

[Visual Studio](http://baike.baidu.com/view/28727.htm)是目前最流行的[Windows](http://baike.baidu.com/view/4821.htm" \t "_blank)平台应用程序的[集成开发环境](http://baike.baidu.com/view/14867.htm" \t "_blank)。Visual studio 2013具有功能全面,灵活性好,效率高,深入底层的优点。另外,Visual Studio 2013具有MFC AppWizard/Class Wizard等功能强大的编程向导工具，在很大程度上简化了应用程序的开发,也提高了编程效率。Visual Studio 2013自然是本项目开发的首选。

3.3系统概要设计

3.3.1 总体设计

系统大体分为3个模块，包括辅助模块、人头检测模块和显示视频图像模块，如图3-4所示



**图3-4 系统主体结构图**

3.3.2 辅助模块设计

辅助系统模块其结果如图3-5所示：



**图3-5 辅助系统模块结构图**

主要完成任务有：

1. 选择人头检测算法：在 WPF中下拉框中选取一种作为接下来的视频图像检测算法。可以选择的检测算法有：SVM 人头检测算法和AdaBoost检测算法。如果不选择，默认为AdaBoost检测算法
2. 选择视频播放源：在WPF中的TreeView控件中对视频进行选择，左键选中视频名称前的复选框，则会获取在后台数据库中存放的对应视频名的视频路径，从而打开该视频，并在视频信息显示框中显示视频名称以及视频路径。
3. 视频源的添加和删除：在WPF中TreeView控件中进行右键操作，会出现添加视频源和删除视频源的功能，在添加视频源功能中，使用者只需添加视频所位于的根节点以及视频名、视频路径即可。在删除视频源的功能中，使用者只需选择视频位于的根节点以及视频名即可。

3.3.3 人头检测模块

系统核心的匹配人头检测模板结构如图 3-6所示



**图3-6 匹配人头检测模板结构图**

此模块主要完成任务有：

1. AdaBoost人头检测：加载人头信息的样本库，将图像进行均衡化，然后进行人头模板匹配。得到匹配后的人头位置。
2. SVM人头检测：加载人头矩阵的信息样本，进行人头模板检测，得到匹配的人头位置。

3.3.4 显示模块

显示模板主要是(1)开启视频检测后显示人头检测后的视频图像，(2)未开启视频检测后视频图像。其主要结构如图3-7所示



**图3-7 显示模块结构图**

该模块只要实现图像的显示和保存操作。显示图像直接利用EmguCV中的ImageBox控件来实现，保存图像默认保存格式为png,用户只需要在C:data\\pic下找到对应的图像文件即可，操作简单容易上手。

3.3.5 接口设计

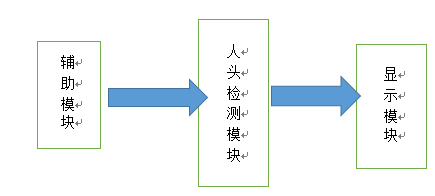
1. 外部接口设计

系统外部接口主要实现系统输入射出设备，例如鼠标，键盘事件的响应。系统利用WPF(Windows Presentation Foundation)，设计相应对待鼠标、键盘监听响应程序。

1. 内部接口设计

内部接口设计主要实现各个子系统之间的数据流的处理

系统首先由用户在辅助系统选择待播放视频和人头检测算法开始，根据显示的已选择视频文件的视频路径，进行播放视频，如果使用者开始视频检测算法，辅助模块会将选择的视频检测算法传递到检测模块，在这里，所有的视频了图像将获得该算法下最大程度的人头检测。待人头检测完成后，人头检测模块会将检测好的视频图像传递给显示模块显示图像和检测信息以及提示保存图片信息。其数据流向如图3-8所示。



**图3-8 系统各模块之间的数据流向**

3.3.6维护设计

系统各模块相互独立，减轻了许多维护升级上的麻烦，只需要在制定模块中作出改动，而不会影响其他模块的正常使用。

3.4 系统详细设计

3.4.1 辅助功能模块详细设计

辅助功能模块主要操作是选择待播放视频，添加视频，删除视频以及选择人头检测算法。

添加播放视频功能中，视频格式为“AVI”或者“WMV”视频文件。主要方法及功能如表3-2所示

**表3-2 辅助模块主要函数以及功能**

|  |  |
| --- | --- |
| 方法名 | 功能 |
| void menuSelectAllChild\_Click(object sender, RoutedEventArgs e) | 选中所有子项菜单事件 |
| void menuExpandAll\_Click(object sender, RoutedEventArgs e) | 全部展开菜单事件 |
| void menuUnExpandAll\_Click(object sender, RoutedEventArgs e) | 全部折叠菜单事件 |
| void menuSelectAll\_Click(object sender, RoutedEventArgs e) | 全部选中事件 |
| void menuUnSelectAll\_Click(object sender, RoutedEventArgs e) | 全部取消选中 |
| void TreeViewItem\_PreviewMouseRightButtonDown(object sender, MouseButtonEventArgs e) | 鼠标右键事件 |
| void TreeViewItem\_PreviewMouseLeftButtonDown(object sender, MouseButtonEventArgs e) | 鼠标左键事件 |
| int ChildrenSelectNum(MonitorCameraTreeModel model) | 获取子项选中的数量 |
| void menuAddItem\_Click(object sender, RoutedEventArgs e) | 点击添加视频文件 | |
| void menuRemoveItem\_Click(object sender, RoutedEventArgs e) | 点击删除视频文件 | |

3.4.2 人头检测模块详细设计

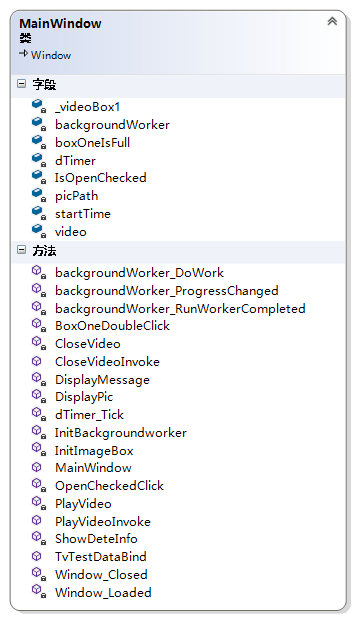
AdaBoost模板检测主要利用人头样本进行检测，SVM模板检测主要利用人头样本矩阵数据进行人头检测。其主要方法和功能如下表3-3所示

**表3-3 人头检测模块主要方法及其功能**

|  |  |
| --- | --- |
| 方法 | 功能 |
| MDeteInfo GetAdaBoostHead(Image<Bgr, Byte> image) | 进行AdaBoost算法人头检测 |
| float[] GetData() | 获取样本数据 |
| MDeteInfo GetSVMHead(Image<Bgr, Byte> image) | 进行SVM算法人头检测 |
| bool StroageImage(Image<Bgr,Byte> image) | 保存图片 |

3.4.3 各部分类图

1. 程序基于面向对象实现设计，其类图为：

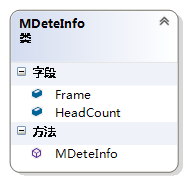


**图3-9 界面类**

详细描述如下:

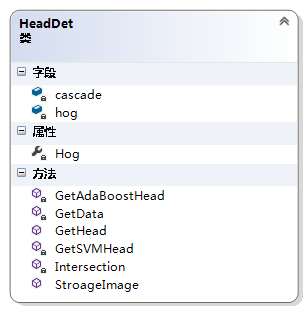
|  |  |
| --- | --- |
| 方法名 | 功能 |
| void PlayVideoInvoke (MonitorCameraTreeModel model) | 播放视频委托 |
| dTimer\_Tick(object sender, EventArgs e) | 定时器委托 |
| void CloseVideoInvoke(MonitorCameraTreeModel model) | 关闭视频委托 |
| void PlayVideo(MonitorCameraTreeModel model) | 播放视频 |
| void DisplayPic(string picPath) | 显示图片 |
| void CloseVideo(Video \_video) | 关闭视频 |
| Window\_Loaded(object sender, RoutedEventArgs e) | 加载窗体时激发 |
| InitImageBox() | 初始化ImageBox |
| void InitBackgroundworker() | 初始化backgroundworker |
| void backgroundWorker\_ProgressChanged(object sender, ProgressChangedEventArgs e) | 报告进度 |
| backgroundWorker\_RunWorkerCompleted(object sender, RunWorkerCompletedEventArgs e) | 操作完成、取消、异常时执行的操作 |
| backgroundWorker\_DoWork(object sender, DoWorkEventArgs e) | 调用RunWorkerAsync时发生 |
| TvTestDataBind() | 数据绑定 |
| Window\_Closed(object sender, EventArgs e) | 窗口关闭 |
| void OpenCheckedClick(object sender, RoutedEventArgs e) | 点击开启检测 |
| DisplayMessage(string message) | 显示操作消息 |
| void ShowDeteInfo(string info) | 检测信息显示 |

1. 显示结果类图



**图3-10 显示结果类图**

1. 进行人头检测类图



**图3-11 人头检测类图**

3.5 系统实现

3.5.1 样本数据加载

1. AdaBoost样本数据加载

在C盘data文件夹下的cascades包含了人头样本数据，使用har分类器加载数据，其代码如下：

HaarCascade cascade = new HaarCascade(@"C:\\data\\cascades.xml");

1. SVM样本数据加载

同样，在C盘data目录文件夹下有个HogDetector.txt，里面包含目标检测的图像特征-HOG特征，使用HOGDescriptor提取特征并进行训练，其代码如下:

if (hog == null)

{

Size winSize = new Size(64,64);

Size blockSize = new Size(16, 16);

Size blockStride = new Size(8, 8);

Size winStride = new Size(8, 8);

Size cellSize = new Size(8, 8);

int nbins = 9;

hog = new HOGDescriptor(winSize, blockSize, blockStride, cellSize, nbins, 1, 1, 0.2, true);

float[] data = GetData();

hog.SetSVMDetector(data);

}

而GetData方法代码如下：

List<float> data = new List<float>();

// 文件读取流读取文本

using (StreamReader reader = new StreamReader("C:\\data\\HogDetector.txt"))

{

while (!reader.EndOfStream) //未读取结束

{

// 获取每行的内容

var line = reader.ReadLine().Split(new[] { ' ' }, StringSplitOptions.RemoveEmptyEntries);

var arr = new float[line.Length];

for (var i = 0; i < line.Length; i++)

{

arr[i] = (float)Convert.ToDouble(line[i]); //将数据转换为float类型

data.Add(arr[i]);

}

}

}

var array = data.ToArray();

return array;

3.5.2 AdaBoost算法人头检测

在加载完AdaBoost所需的样本数据后，将图像进行灰度化和均衡化，并利用EmguCV中的DetectHaarCascade方法进行目标检测，其主要代码如下:

double scale = 1.3;

Image<Bgr, Byte> smallframe = image.Resize(1 / scale, Emgu.CV.CvEnum.INTER.CV\_INTER\_LINEAR); //重新设置图片大小

Image<Gray, Byte> gray = smallframe.Convert<Gray, Byte>(); //获取灰度图像

gray.\_EqualizeHist(); //均衡化

// 开始进行检测

MCvAvgComp[][] faceDetected = gray.DetectHaarCascade(

cascade,

1.1,

2,

Emgu.CV.CvEnum.HAAR\_DETECTION\_TYPE.DO\_CANNY\_PRUNING,

new Size(30, 30));

int count = 0;

// 检测到的目标显示到图像上

foreach (var item in faceDetected[0])

{

count++;

image.Draw(item.rect, new Bgr(Color.Red), 3);

}

model.Frame = image;

model.HeadCount = count;

3.5.3 SVM算法人头检测

利用前面获取的样本数据，利用EmguCV中的检测方法进行目标检测。其主要代码如下：

List<Rectangle> rects = new List<Rectangle>();

Size winStride = new Size(8, 8);

Size winSize = new Size(64,64);

Rectangle roiRect = new Rectangle(144, 206, 406, 360);

int count = 0;

Rectangle[] rectangle = Hog.DetectMultiScale(image);

if (rectangle.Length > 0)

{

Rectangle temp = new Rectangle();

bool pushFlag;

foreach (var item in rectangle)

{

pushFlag = true;

temp.X = item.X + roiRect.X;

temp.Y = item.Y + roiRect.Y;

temp.Width = winSize.Width;

temp.Height = winSize.Height;

foreach(var rectNum in rects)

{

if (Intersection(rectNum, temp) > 0.2)

{

pushFlag = false;

break;

}

}

if (pushFlag)

{

rects.Add(temp);

}

count++;

}

foreach (var r in rects)

{

image.Draw(r, new Bgr(Color.Red), 3);

}

}

而Intersection方法主要实现如下：

if (rect1.X > rect2.X + rect2.Width)

{

return 0.0f;

}

if (rect1.Y > rect2.Y + rect2.Height)

{

return 0.0f;

}

if (rect1.X + rect1.Width < rect2.X)

{

return 0.0f;

}

if (rect1.Y + rect1.Height < rect2.Y)

{

return 0.0f;

}

//交集矩形的坐标右上角点的x和左下角点的y

float right\_up\_x, left\_down\_y;

//交集矩形的左侧x的坐标和上侧的y坐标

float left\_x, up\_y;

//交集矩形的面积,两个矩形的面积

float area\_Intersection, area1, area2;

//对交集区域右上角的x坐标赋值

if (rect1.X + rect1.Width > rect2.X + rect2.Width)

{

right\_up\_x = rect2.X + rect2.Width;

}

else

{

right\_up\_x = rect1.X + rect1.Width;

}

//对交集区域左下角的y坐标赋值

if (rect1.Y + rect1.Height > rect2.Y + rect2.Height)

{

left\_down\_y = rect2.Y + rect2.Height;

}

else

{

left\_down\_y = rect1.Y + rect1.Height;

}

//对交集矩形的左侧x的坐标赋值

if (rect1.X > rect2.X)

{

left\_x = rect1.X;

}

else

{

left\_x = rect2.X;

}

//对交集矩形的上侧y的坐标赋值

if (rect1.Y > rect2.Y)

{

up\_y = rect1.Y;

}

else

{

up\_y = rect2.Y;

}

//计算交集的面积,及两个矩形的面积

area\_Intersection = (right\_up\_x - left\_x) \* (left\_down\_y - up\_y);

area1 = rect1.Width \* rect1.Height;

area2 = rect2.Width \* rect2.Height;

//返回覆盖比例

float coverRate;

coverRate = area\_Intersection / (area1 + area2 - area\_Intersection);

return coverRate;

3.5.4 视频播放实现

视频源的信息来自于xml文件，所以需要对xml文件进行解析

// 加载xml文件

XmlDocument doc = new XmlDocument();

XmlReaderSettings settings = new XmlReaderSettings();

// 忽略文档里面的注释

settings.IgnoreProcessingInstructions = true;

XmlReader reader = XmlReader.Create(xmlPath, settings);

doc.Load(reader);

// 得到根节点videoInfo

XmlNode xn = doc.SelectSingleNode("videoInfo");

// 得到根节点的所有子节点

XmlNodeList xnList = xn.ChildNodes;

foreach (var xnl in xnList)

{

MonitorCameraTreeModel camerModel = new MonitorCameraTreeModel();

// 将节点转换为元素，便于得到节点的属性值

XmlElement xe = (XmlElement)xnl;

// 得到Type和id两个属性的属性值

camerModel.Id = xe.GetAttribute("id").ToString();

// 得到Book节点的所有子节点

XmlNodeList xnl0 = xe.ChildNodes;

camerModel.Name = xnl0.Item(0).InnerText;

camerModel.IsChecked = Convert.ToBoolean(xnl0.Item(2).InnerText);

camerModel.IsExpanded = Convert.ToBoolean(xnl0.Item(3).InnerText);

if (xnl0.Item(4).HasChildNodes)

{

XmlNodeList xnl1 = xnl0.Item(4).ChildNodes;

foreach (var xnl2 in xnl1)

{

MonitorCameraTreeModel child = new MonitorCameraTreeModel();

XmlElement xe1 = (XmlElement)xnl2;

XmlNodeList xnl3 = xe1.ChildNodes;

child.Id = xnl3.Item(0).InnerText;

child.Name = xnl3.Item(1).InnerText;

child.VideoPath = xnl3.Item(2).InnerText;

child.Parent = camerModel;

camerModel.Children.Add(child);

}

}

treeList.Add(camerModel);

}

reader.Close();

播放视频主要是选中视频名称前的复选框，则会自动加载该视频图像，主要流程为（1）选中视频；（2）进行简单判断（3）异步委托开启播放线程，其主要代码实现如下：

// 获取复选框的状态

if (e.ButtonState == MouseButtonState.Pressed)

{

// 将对象转换为自定义树状视频对象

TreeViewItem item = VisualUpwardSearch<TreeViewItem>(e.OriginalSource as DependencyObject) as TreeViewItem;

if (item != null) //如果对象不为空

{

MonitorCameraTreeModel selectModel = (MonitorCameraTreeModel)item.Header;

if (!selectModel.IsChecked) //如果为非选中状态

{

selectModel.IsChecked = true; // 置为选中状态

if (selectModel.Parent != null) // 如存在父节点

{

selectModel.Parent.IsChecked = true; // 父节点也为选中状态

}

if (selectModel.Children != null && selectModel.Children.Count < 2) //如果选中父节点且子节点数小于2

{

selectModel.SetChildrenChecked(true);

}

else // 如果子节点大于2 则输出提示信息

{

MessageBox.Show("最多播放一个视频");

selectModel.IsChecked = false;

return;

}

if (selectModel.Children == null || selectModel.Children.Count == 0)

{

// 选中的子节点 则播放视频 ((MainWindow)Application.Current.MainWindow).PlayVideoInvoke(selectModel);

}

}

else

{

selectModel.IsChecked = false; // 如果已经为选中，则置为非选中

if (selectModel.Children != null && selectModel.Children.Count != 0)

{

// 如果为父节点 则应关闭子节点所播放的视频同时设置子节点为非选中状态

foreach (var model in selectModel.Children)

{

if (model.IsChecked)

{

((MainWindow)Application.Current.MainWindow).CloseVideoInvoke(model);

}

}

selectModel.SetChildrenChecked(false);

}

if (selectModel.Parent != null)

{

if (this.ChildrenSelectNum(selectModel.Parent) == 0)

{

selectModel.Parent.IsChecked = false;

}

}

if (selectModel.Children == null || selectModel.Children.Count == 0)

{

((MainWindow)Application.Current.MainWindow).CloseVideoInvoke(selectModel);

}

}

e.Handled = true;

}

}

而在异步委托播放视频时，使用BackgroundWoker[13]类代替Thread类，具体实现为;

backgroundWorker.RunWorkerAsync(model);

则会激发对应的DoWork事件，主要功能为完成视频的播放:

BackgroundWorker worker = sender as BackgroundWorker;

MonitorCameraTreeModel model = e.Argument as MonitorCameraTreeModel;

string path = model.VideoPath;

if (path.ToUpper().Contains("JPG") || path.ToUpper().Contains("JPEG"))

{

this.DisplayPic(path);

}

else

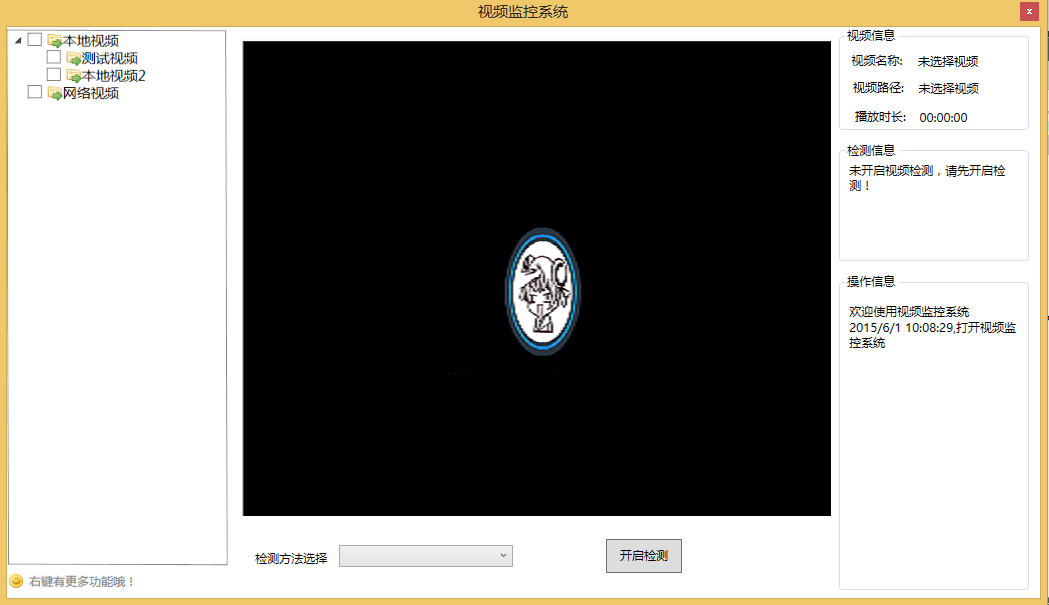
{

this.PlayVideo(model);

}

3.6 系统效果演示

系统主窗体如图3-15所示：



**图3-15**

系统未开启视频检测时运行界面如图3-16所示



**图3-16**

系统开启AdaBoost检测算法如图3-17所示



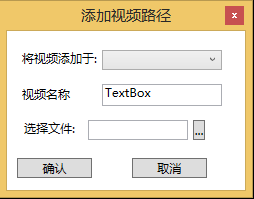
**图3-17**

系统开启SVM检测算法如图3-18所示



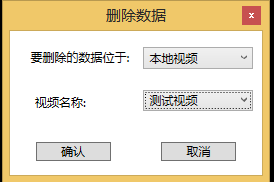
**图3-18**

系统添加视频源如图3-19所示



**图3-19**

系统删除视频源如图3-20所示



**图3-20**

3.7本章小结

本章从系统整体需求出发，得出系统所需的功能，进而设计出各个模块，然后再是各个类来实现各个模块，以及各个类模块的主要实现代码最终实现了满足需求门禁视频监控系统。在进行人头检测中，分别采用了AdaBoost人头检测算法和SVM人头检测算法，并对两个算法进行了比较，在下章会详细介绍两个算法之间的区别。在本章第6小结，还进行了系统效果演示，对门禁视频监控系统了更加深刻的认识。

第4章 实验结果对比分析

4.1 AdaBoost和SVM算法效果对比

采用AdaBoost检测算法，针对第一组测试视频时，其测试结果如图4-1所示，采用SVM检测算法，针对第一组图像，其检测结果如图4-2所示，两幅图片一对比，可以很明显的看出，在监控图像不是很清晰时，由于周边环境不是很明亮时，AdaBoost检测算法存在很大的错误检测。使用SVM检测能够准确检测出人头。但在光线明亮的环境中，如图4-3所示，在光线明亮下AdaBoost检测结果，图4-4在光线明亮下SVM检测结果，很显然AdaBoost也存在很大的错误检测，而SVM虽然检测出了人头，但是也出现了检测数量不对，存在少检测情况。

原因在于

1：虽然本次使用了2000余张样本图像，但还是样本图像过少，使得建立出的样本数据过少，在后面进行特征提取时，不能全面获取所有数据，导致检测结果不准确，从而影响检测质量

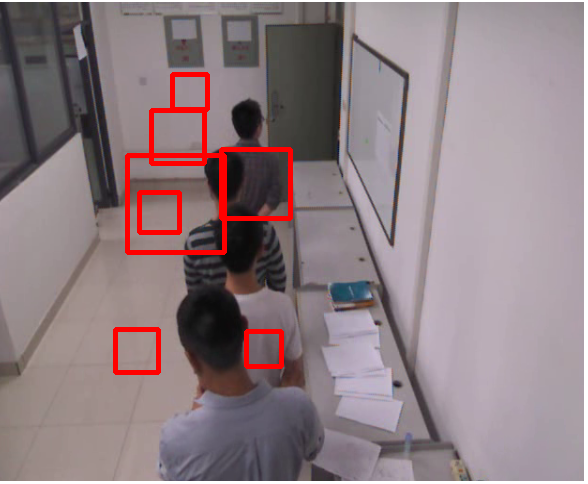
2：本次选取的特征样本正样本和负样本数量几乎一样多，导致加大了被误判样本的权值，在训练过程中，每次迭代都会对那些分类错误的样本进行加权，但多个这样的样本多次分类错误过后，他们的权重过大，进行左右误差的计算和分类器的挑选。使分类器的精度下降。出现典型的“退化问题”[12]。



**图4-1 AdaBoost检测结果**



**图4-2 SVM检测结果**



**图4-3 AdaBoost检测结果**



**图4-5 SVM检测结果**

4.2结论分析

本系统采用了AdaBoost人头检测算法和SVM人头检测算法，显然基于SVM算法的人头检测比简单的AdaBoost人头检测算法精度要高，SVM算法略占优势。虽然在AdaBoost检测算法上出现很多不足，但为后期的算法优化作出铺垫，由于开发周期短，系统仍然存在许多的不完善的地方，需要进一步改进：

1. 随着现在网络的普及，越来越多的系统都采用B/S或者C/S结构，所以系统可以采用C/S或者B/S结构，将系统的检测功能移到服务器上，而操作人员只需要远程点击检测功能，由客户端发送视频图像，在服务器上进行检测，返回检测后的视频图像信息。
2. 系统在安全性方面还需要进一步的加强。为用户提供可靠的保障。
3. 在系统的开发过程中，对视频图像处理的技术有一定的了解，但由于作者本人能力和时间的限制，对视频图像处理处理的难免有些不足之处，不免有些错误，所以还需要在该方面做进一步的研究。
4. 结合门禁卡，实现人员进出门禁时，自动比较刷卡次数与出入人员数量是否一致，是否有人员未刷卡，并及时作出相应的判断。

参考文献

1. 陈东伟,韩娜.嵌入式数据库在基于多核处理器的视频监控中的应用〔J〕.郑州大学2007(4):52一55.
2. 杨富国.Visua1 C++程序开发案例解析[M].北京:清华大学出版社;北京交通大学出版社 2006.
3. 王华. 智能视频监控系统设计与实现. 山东大学 2013
4. 刘瑞祯.OpenCV教程基础篇[M]. 北京：北京航空航天大学出版社 2007
5. http://www.opencv.org.cn/opencvdoc/2.3.2/html/doc/tutorials/imgproc/histograms/template\_matching/template\_matching.html?highlight=matchtemplate
6. 黄佳.基于OPENCV的计算机视觉技术研究.上海.华中理工大学
7. Maggioni C and Kammerer B. "Gesture Computer: history,design, and applications Computer Vision of Human-Machine Interaction[J]". Cambridge Univ. Press, 1998.
8. Haritaoglu I,Harwood D,Davis L. UW4: real-time surveillance of people and their activities[J]H. IEEE Trans PaRem Analysis and Machine Intlligence,2000,22(8): 809—830
9. 郑金荣.视频监控系统中运动目标跟踪算法的研究[D].硕士学位论文,江南大学,2009
10. 徐振兴.基于机器视觉的行人检测和跟踪技术研宄[D].硕士学位论文,浙江理工大学,2012
11. 牛胜石. 基于 AdaBoost 和 SVM 的人头检测[D].硕士学位论文,中南民族大学，2010
12. 廖红文，周德龙.AdaBoost及其改进算法综述[J].浙江工业大学.2012(21)
13. https://msdn.microsoft.com/zh-cn/library/system.componentmodel.backgroundworker.aspx

致谢

光阴似箭,四年的本科生学习生活很快就要结束了。在西南科技大学的这四年里,我周围发生了很大的变化,最重要的是我四年前制定的目标实现了,四年让我收获颇丰。

本论文的工作是在我的导师刘畅老师的悉心指导下完成的。在本课题的研究以及论文的撰写的过程中,刘老师给与我精心的指导;在生活中刘老师也给与我了无微不至的关怀。正是这些指导和关怀使得我在本科生的学习期间取得了很大的进步。刘老师严谨的治学态度、科学的工作方法以及朴实的生活作风对我有着深远的影响。在此向刘畅老师表达衷心的谢意。

前几届的师兄任新宇对于我的工作学习以及论文都提出了许多宝贵的意见,对于论文的内容以及存在的问题提供了许多指导以及帮助。同时感谢所有在我学习期间教导过我的老师,谢谢你们在我求学的过程中给与的关怀和帮助,你们的一言一行都将成为我人生未来的宝贵财富。

在课题研究以及论文的撰写过程中,许多同学对我论文中的研究工作也提供了热情的帮助,在此向他们也表达我的感激之情。最后特别感谢我的父母,他们一直在我的身边默默地鼓励、关心、支持我,是他们的理解和支持使能够专注于研究和学习,顺利完成我的学业。