面向大规模监控系统的视频计算架构及方法研究

作者姓名 田朝会

指导教师姓名、职称 沈玉龙 教授

申请学位类别 工学硕士

学校代码 10701　　学　号 1403121571
分 类 号 TP39　　密　级 公开

西安电子科技大学

硕士学位论文

面向大规模监控系统的视频计算架构及方法研究

作者姓名：田朝会

一级学科：计算机科学与技术

二级学科：计算机系统结构

学位类别：工学硕士

指导教师姓名、职称：沈玉龙　教授

学　　院：计算机学院

提交日期：2017 年 6 月

Research on Video Computing Architecture and Method for Large Scale Surveillance System

A thesis submitted to

XIDIAN UNIVERSITY

in partial fulfillment of the requirements

for the degree of Master

in Computer System Architecture

By

Tian Chaohui

Supervisor: Shen Yulong Title: Professor

June 2017

西安电子科技大学
学位论文独创性（或创新性）声明

秉承学校严谨的学风和优良的科学道德，本人声明所呈交的论文是我个人在导师指导下进行的研究工作及取得的研究成果。尽我所知，除了文中特别加以标注和致谢中所罗列的内容以外，论文中不包含其他人已经发表或撰写过的研究成果；也不包含为获得西安电子科技大学或其它教育机构的学位或证书而使用过的材料。与我一同工作的同事对本研究所做的任何贡献均已在论文中作了明确的说明并表示了谢意。

学位论文若有不实之处，本人承担一切法律责任。

本人签名：_____　　日　　期：_____

西安电子科技大学
关于论文使用授权的说明

本人完全了解西安电子科技大学有关保留和使用学位论文的规定，即：研究生在校攻读学位期间论文工作的知识产权属于西安电子科技大学。学校有权保留送交论文的复印件，允许查阅、借阅论文；学校可以公布论文的全部或部分内容，允许采用影印、缩印或其它复制手段保存论文。同时本人保证，结合学位论文研究成果完成的论文、发明专利等成果，署名单位为西安电子科技大学。

保密的学位论文在___年解密后适用本授权书。

本人签名：_____　　导师签名：_____

日　　期：_____　　日　　期：_____

摘要

视频监控技术作为实现“平安城市”和“智慧城市”系统的重要技术手段受到了社会各界的重视。为提高监控系统效率，实现智能监控，面向大规模监控系统的视频计算技术已经成为迫切需求。本文聚焦于大规模视频分析所面临的数据传输瓶颈、计算能力瓶颈以及视频分析算法误报率较高等问题。

针对数据传输瓶颈问题，提出一种边缘化视频计算架构，围绕视频数据源部署和分配计算资源，克服传统计算方式需要传输大量视频的问题。通过部署区域计算节点处理区域内的视频数据，部署核心节点实现区域计算节点的统一调度和管理，实现基于区域和视频源的视频计算资源部署和分配模式。计算节点只需将计算后的信息片段发送到核心节点，有效降低通信带宽的消耗。

针对计算能力瓶颈问题，提出一种计算资源动态调整策略。针对监控视频价值密度低的特点，提出一种自适应的视频丢帧策略和基于视频重要程度的计算资源分配方法。当计算任务骤增，计算能力不足时，根据丢帧策略选择性丢弃部分摄像头的视频帧，减小其对计算资源的消耗，保证系统稳定的同时使重要计算任务得到充足的计算资源。

针对视频计算结果误报率较高的问题，结合视频计算中常用的行人检测算法，提出两种降低误报率的方法：1) 基于可信度的自适应行人检测方法。在时间和空间两个维度对行人检测分类器的可信度进行划分，同时引入用户对判断结果的反馈，不断更新可信度，使误报率逐渐降低；2) 基于贝叶斯推断的行人检测结果修正方法。使用贝叶斯算法训练历史结果样本集，形成修正分类器，使用修正分类器对检测分类器的识别结果做二次判断。

基于以上提出的计算架构与方法，设计实现了面向大规模监控系统的视频计算平台，在多个平安城市系统中得到应用，验证了所提出的架构和方法的可行性、有效性和实用性。

关 键 词：视频监控， 平安城市， 传输瓶颈， 计算架构， 动态调整， 误报率

ABSTRACT

To realize the concepts behind the term "safe city" and "smart city", video surveillance technology has received a great attention by the research community. In order to improve the efficiency of the surveillance system and realize intelligent surveillance, the video computing technology for the large-scale surveillance system was proven to be critical. This paper focuses on studying the bottleneck of data transmission, computing ability and the high false alarm rate of video analysis algorithms, hence, these issues were being faced when a large-scale video analysis system is in consideration.

Considering the bottleneck of data transmission, A marginalized video computing architecture is proposed, which deploys and allocates computing resources around the video data source, in order to overcome the problems faced when large amounts of video data are being transmitted. By deploying regional computing nodes to process video data and core node to unify the scheduling and manage regional computing nodes, the video computing resource deployment and the distribution mode based on the region of the video source are being realized. The computing nodes send the calculated information fragment to the core node, which can effectively reduce the communication bandwidth consumption.

Considering the bottleneck of computing ability, a dynamic adjustment strategy of computing resources is proposed. Considering the low value of the video, an adaptive video frame dropping strategy and a computing resource allocation method based on the value of video are proposed. When the computing task is increasing and the computing power is insufficient, the video frame of some cameras will be discarded according to the frame discard strategy, which will reduce the cost of computing resources, to ensure the stability of the system and the computing tasks with high urgency will obtain sufficient resources instantly.

Considering the problem of high false alarm rate in video computing, combined with the pedestrian detection algorithm, which is commonly used in video computing, two methods are proposed, in order to reduce the false alarm rate and they are as follows: 1) Adaptive Pedestrian Detection Method Based on Credibility. The credibility of the pedestrian detection classifier is divided into two dimensions (i.e. time and space), and the feedback of

the judgment result is introduced to update the reliability. So that the false alarm rate gradually reduced. 2) Correction Method of Pedestrian Detection Results Based on The Bayesian Algorithm. The Bayesian algorithm is used to train the historical judgment results, and a modified classifier will be formed. The modified classifier will be used to make the second judgment on the recognition result of the detection classifier.

Based on the aforementioned calculation architecture and methods, a video computing platform for large-scale surveillance system is designed and implemented. The proposed system was applied in many "safe city" systems, which insures its feasibility, effectiveness and practicability.

Keywords: Video Surveillance, Safe City, Transmission Bottleneck, Computing Architecture, Dynamic Adjustment, False Alarm Rate

插图索引

表格索引

缩略语对照表

缩略语	英文全称	中文对照
IVS	Intelligent Video Surveillance	智能视频监控
HOG	Histogram of Oriented Gradient	梯度直方图特征
SVM	Support Vector Machine	支持向量机

目录

第一章　绪论

1.1 课题研究背景及意义

随着“平安城市”、“智慧城市”的建设不断深入，视频监控技术作为实现“平安城市”和“智慧城市”的重要技术手段也越来越受到社会各界的重视，同时在各个行业得到了广泛的应用。2010 年，为了顺利、安全地举办世博会，上海市大规模增加监控网络的范围和密度，仅在浦东新区就建设了包含一万多个监控点位的高清监控系统[1]；截至 2011 年，黑龙江投入 8 亿多元建设资金用于监控系统建设，共建设近千个监控中心，二十多万个监控点[2]；武汉市为推进“平安城市”建设，建设了覆盖跨江大桥、主干道和主城区城道口等重点部位的监控系统[3]。

视频监控系统的发展主要分为以下三个阶段[4]：

第一阶段：以模拟摄像机及模拟录像机为主的模拟视频监控系统，这种监控系统多是以摄像机、录像机、分割器为核心，通过模拟信号输出视频，传输距离短，图像质量差，目前已逐步退出市场。

第二阶段：基于模拟摄像机和数字录像机的视频监控系统，采用模拟信号传输视频，数字方式处理与贮存，属于过渡阶段。

第三阶段：基于网络摄像机的全数字视频监控系统，以数字方式传输视频，传输距离长，图像质量好，目前正得到广泛应用。

以上三代视频监控系统，主要解决视频捕捉、传输、存储等问题，并不具备视频智能分析的功能，真正起到安全防范的作用则需要人为观察视频从而做出安全性评估。一种方法是实时观看，建设一个监控中心，将监控点所有的视频流导入监控中心，同时配备多块显示屏来播放视频画面，由工作人员通过观看屏幕上的监控流来判断当前监控点所发生的事情；另一种方法是事后查看，用存储设备记录监控点的视频，事后进行调阅，本质上还是以人观看的方式来查找感兴趣的信息。

然而人工监测具有本身固有的缺陷，例如：消耗大量人力、易使人产生疲劳、可扩展性差等等。尤其随着“平安城市”监控系统规模不断扩大，面对数以万计的监控点和海量视频数据，以人工为核心的监控系统越来越无法满足需求。如何解决人工监测带来的种种问题，加强监控系统的事前预警和事后的检索，成为当前摆在“平安城市”发展前的重要问题。

监控界提出了智能视频监控（Intelligent Video Surveillance, IVS）[5]的概念。智能监控即使用计算机强大的计算能力以及图像分析技术对传统的监控视频进行处理、分析和理解，过滤无用信息，提取关键数据。目前智能视频监控在计算机视觉领域中备

受关注。

相比于传统监控系统，智能监控系统具有以下优势：

1) 全天候的监控能力：使用计算机代替人来监控视频画面，实现对监控画面的不间断分析，彻底改变以往由监控人员对监控画面进行监视和分析的方式。

2) 快速的响应能力：通过设置某些可疑行为的识别规则，在其发生危害之前通知相关人员采取相应措施，为潜在威胁做好准备工作。

3) 扩展视频资源的应用领域：传统视频监控系统只能获取单一的视频数据，而智能监控系统可以提取视频中的各种信息，为其他应用提供支持。

随着“平安城市”监控系统规模的不断扩大，面向大规模监控系统的视频计算对智能监控系统提出了新的挑战：1) 数据传输瓶颈。监控视频流数据量大，需要较高的网络带宽，如何解决数据传输瓶颈，将视频流导入计算分析平台，是智能监控系统需要解决的首要问题；2) 计算能力不足。图像分析算法是计算密集型的工作，需要消耗大量的计算资源，如何保证充足的计算资源也是亟待解决的问题；3) 图像分析算法易受环境影响。视频计算技术使用的图像分析算法在不同场景下，受光照变化、背景变化的影响，其识别效果的准确性也会受到影响。

本文针对以上三个问题进行研究，有利于实现智能监控，推进“平安城市”建设，对公共治安、人身安全、社会稳定等具有重要意义。

1.2 国内外发展研究现状

视频计算就是依靠图像、视频分析算法实时分析监控视频，进行图像识别、行为分析，最终达到异常预警、轨迹追踪等目的。作为计算密集型的任务，视频计算需要消耗大量的计算资源，同时为了满足实时处理的需求，视频分析算法的时间需要尽可能短。目前，视频计算的研究方向主要有两个，一个是视频计算算法的研究，另一个是视频计算架构研究。

在视频计算算法研究方面，1) 为满足实时处理需求，部分研究者进行了以下研究。Diamantopoulos G 和 Span M [6]提出一种自适应背景学习算法，智能分析停车场环境下的物体遮挡情况，并达到准实时分析要求。Wenying Wang, Zhang D, Zhang Y [7]等人通过使用 CUDA 编程模型，在图形处理器（GPU）上并行加速，来减小耗时以满足实时性要求。Peter Ahrendt, Gregersen T, Karstoft H [8].等人通过使用 C 改写 Matlab 代码，来实现圈养猪的实时跟踪。Pramodat P 和 Shirvaikar M[9]针对视频流中的人脸识别算法进行优化，同时使用高性能的 arm 处理器进行加速处理，Qiuwen Chen, Qiu Q, Li H [10]等人通过使用电阻器的交叉队列来加速计算。Xu R,Tsai C Y, Kender J R[11]等人针对新闻视频分析，提出一种锚帧检测算法，可以快速高效地从视频中检测锚帧，

显著提高了视觉提取的效率。2) 在多摄像头协作处理算法方面，C.Micheloni,G.L. Foresti and L. Snidaro[12]等人针对停车场中的视频监控系统，提出一种多摄像头协作的视频分析算法，可以在受控区域实时追踪多个目标。Niu, P.K. Varshney[13]等人提出一种可适应复杂天气条件的摄像头协作方法，针对不同场景，通过比较摄像头的置信度，选择最适合的摄像头来执行具体任务。Jacqueline Chistmas[14]提出一种用于估计两个摄像头之间的图像场的运动的方法，通过基于少量参数的空间坐标理论运动函数来指定图像之间像素的位移。

在视频计算架构研究方面，Wang G, Tao L, Di H [15]等人针对智能监控系统提出了一种可扩展的分布式架构。Hong K 和 Voelz M[16]提出一种分布式框架，有助于在大型摄像机网络和后端计算资源上开发和部署时空分析应用。Saini M, Wang X, Atrey P K [17]等人通过使用弹性云的部署方式，实现一种负载均衡的多摄像头监控系统。Geng Chen yao[18]通过使用分布式云计算平台 MapReduce 来满足大量计算需求。李招昕[19]提出一种基于流式计算的大规模监控视频分析方法，采用 Spark 对监控视频进行分析。方权亮[20]提出一种基于云计算的智能高清视频监控系统的解决方案。通过整合计算和存储资源，实现了对硬件资源的统一配置，提高了系统的可扩展性。Sanchez N 和 Menendez J M[21]针对交通安全问题，提出一种视频分析架构，能够处理低级任务和高级行为推理。曹海滨[22]针对多媒体领域大量视频，提出一种海量视频的分布式协作处理架构，有效地简化分布式环境下实现多媒体内容处理的难度。Bansal R, Sehgal P, Bhasin V[23]等人采用一种多 Agent 系统的分布式系统来生成图像的水印指纹，并有效地进行数据分发和结果收集。

如前所述，目前已经有许多研究者在视频计算方面做了很多研究。然而，结合“平安城市”监控系统特点进行视频计算的相关研究依然较少，面向“平安城市”中大规模监控系统的视频计算研究依然处于初级发展阶段。

1.3 本文的研究内容及主要工作

本文针对“平安城市”监控系统的建设方式以及视频分析算法进行了深入研究，分析了当前已有的视频计算平台的优缺点，并结合监控系统规模不断扩大的特点，考虑数据传输瓶颈，计算能力不足以及视频计算结果准确度不高的问题，提出“以数据为中心”边缘化的视频计算架构以及两种降低视频计算误报率的方法。

本文的研究内容包括：

1) 研究了当前监控系统建设的特点以及面向大规模监控系统的视频计算技术所面临的问题。当前监控系统存在规模庞大、区域自治的特点，大量区域如学校、酒店和商场等都自建了本区域内的监控系统。如果集中对海量的摄像头进行分析，将会对

分析平台造成巨大的数据传输压力以及计算能力压力。针对以上问题，本文提出边缘化、分布式的视频计算架构。围绕数据部署计算资源，充分利用以太网的带宽资源，使视频数据在区域内得到计算，同时设置核心节点对所有计算资源进行统一管理，计算节点只需将计算后的信息片段发送到核心节点，从而降低通信带宽的消耗，减小数据传输压力。

2) 视频分析需要大量的计算资源，而且对计算能力的要求并不是一成不变的，会随着图像清晰度、图像质量的变化而动态变化，保证充足的计算能力是必须要考虑的问题。针对计算能力不足的问题，本文提出一种计算能力动态调整策略，主要包括两部分，自适应视频帧丢弃算法和动态增加计算性能算法。当系统出现计算能力不足时，首先启用自适应视频帧丢弃算法，丢弃非重要摄像头的部分视频帧，使计算资源向重要摄像头倾斜，保证系统的稳定。后期通过增加计算节点，提升计算性能，使所有计算任务得到充足的计算资源。本文在第三章将详细介绍计算能力动态调整策略。

3) 图像分析算法容易受光照、背景等外界因素的影响，造成误报率较高。本文结合目前常用的行人检测算法，系统研究了其实现步骤，并提出两种降低其误报率的方法：(a) 基于可信度的自适应行人检测方法。行人检测分类器受外界因素影响，造成在某些时段，某些位置识别效果较差。本文通过在时间和空间两个维度对检测分类器的识别结果进行可信度划分，同时引入用户的反馈，使误报率逐渐降低；(b) 基于贝叶斯推断的行人检测结果修正方法。通过贝叶斯分类算法，对检测分类器的识别结果进行二次判断，得出最终结果。

最后，本文对上述提出的视频计算架构和方法进行了实现和测试，验证了所提出的模型和方法的可行性、有效性和实用性。

1.4 论文结构安排

本文总体分为六章对研究工作进行相应的描述，每章结构安排如下：

第一章基于“平安城市”监控系统的建设背景，分析了目前监控系统的特点和问题，然后介绍了国内外研究现状，最后引出本文着重研究的内容以及论文安排。

第二章针对本文所研究的内容，对相关技术进行了研究。包括视频计算技术、计算机视觉库 OpenCV 以及目前“平安城市”推进过程中，监控系统的主要建设方式。

第三章结合目前监控系统的主要建设方式，提出一种“以数据为中心”边缘化的视频计算架构，同时对模型的任务调度策略以及计算资源动态扩展策略进行了深入分析。

第四章结合视频分析中常用的行人检测算法，针对其误报率较高的问题，提出两种解决方法：基于可信度的自适应行人检测方法和基于贝叶斯推断的行人检测结

果修正方法。

第五章基于前述的视频计算架构，对总体和主要模块进行了详细设计，并给出效果展示；同时针对第四章的两种方法进行了验证。

第六章对研究内容进行总结，指出了研究中的不足和待完善的地方，并对下一步研究进行展望。

第二章 相关技术研究

为提出一种适用于“平安城市”建设的、面向大规模监控系统的视频计算架构，本章对“平安城市”监控系统的建设方式、视频计算技术进行了深入研究。同时介绍了平台实现过程中涉及的行人识别算法以及计算机视觉库 OpenCV。

2.1 “平安城市”监控系统的建设方式研究

本文基于“平安城市”视频监控系统提出面向大规模监控系统的视频计算架构，因此需要对目前“平安城市”监控系统的建设方式进行研究。

目前“平安城市”监控系统的建设，主要有两大方案:方案一，后端布局 IP-SAN 集中存储模式；方案二，前端 NVR 分布存储。

2.1.1 后端布局 IP-SAN 集中存储模式

该种方式着眼于长远布局持续发展，强调打造大平台与强后端存储。主要由以下三部分构成：

1) 高清前端建设：采用至少 130 万像素高清网络摄像机，通过专有网络实时将视频数据传送至各个派出所、公安分局进行实时存储，集中管理。具有存储数据安全的优点，但也大大增加了前端设备到派出所或公安分局的数据带宽压力，同时增加了建设 IP-SAN 存储机器汇聚交换设备的成本。

2) 分局机房存储建设：在各分局或派出所建设视频数据存储中心，实时存储前端传来的视频数据。

3) 集中防控管理平台建设：在核心调度机房建设视频监控中心平台，集中对前端设备、流媒体服务器等进行管理，确保平台易扩展、业务量易扩容。

2.1.2 前端 NVR 分布存储

该种方式较为简单，强调前端建设。前端设备与前端的 NVR 设备组成一个小系统，通过传输网络将该小系统的视频信号传输到后端调度平台，由调度平台将视频信号转发给相关公安分局或派出所。主要由两部分组成：

1) 地区监控小系统：指目前已经存在的区域监控系统，例如学校、医院、社区等自建的监控系统。主要包括，前端摄像头，汇聚交换设备以及 NVR。前端摄像头通过以太网将视频信号传输至 NVR 设备，NVR 存储视频数据，同时提供远程访问方式供其他用户远程访问。

2) 一体化管理平台：在各公安分局或派出所建立一体化管理平台，对各场所、地区的设备统一管理。

2.1.3 方案对比分析

后端布局集中存储方式是一种自上而下的建设方案，系统规划详细，物理介质集中存放，统一管理，在稳定可靠、统筹管理等方面均有优势。然而集中存储需要建立专门的数据中心，需要大量的配套设施，建设成本较高。

前端 NVR 分布式存储是一种自下而上的建设方案，充分利用现有的监控资源、简单易行、方便实施，每个前端可以相对独立，无需投入过多资金。当然也存在后台支撑较弱、数据的安全性较差、运算速度不高的缺陷。

目前两种方案在建设平安城市视频监控项目中均是正常的建设方案，都服务于城市的安防、交通、应急调度等。两者有效互补大大加快了平安城市监控系统建设的深度及广度。其中，由于前端 NVR 分布式存储方式可以有效利用当前已建设的监控系统，投入资金较少，因此应用更为广泛。

2.2 视频计算技术

智能视频监控系统以视频计算技术为基础，使用计算机强大的处理能力，对监控系统视频进行处理分析，充分挖掘传统监控系统的潜力，将事后取证变为实时报警和提前预防。本节将研究视频计算技术，为后续章节展开奠定基础。

2.2.1 视频计算算法流程

视频计算是指使用计算机强大的计算能力代替人工，分析、计算、理解视频流，从中提取有价值的视频信息，例如识别特定的人或物体、确定目标运动区域以及理解目标当前行为等。视频是由一帧帧图像构成的，目前对视频的分析计算，主要是针对视频帧的分析与处理，主要可以三个层次[24]，如图 2.1 所示。

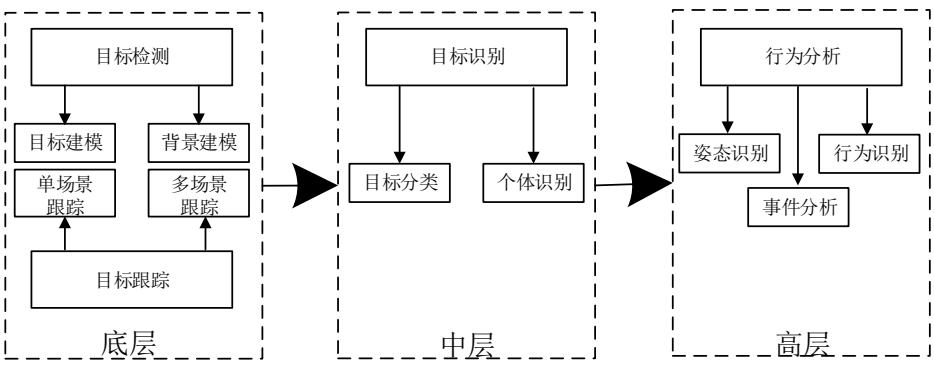

图2.1 视频计算算法流程示意图

1) 底层：从前端设备采集视频流，获取图像序列，检测和跟踪感兴趣的目标。目标检测在图像分析算法中处于基础地位，用来确定目标在当前视频帧中的位置以及大小。目标跟踪用来确定目标在连续的图像序列中的轨迹，具有广泛的应用价值，按场景数量可分为单场景跟踪和多场景跟踪。

2) 中层：主要为目标识别，在底层检测到的目标的基础上，提取目标的各种信息，例如目标的颜色、大小等，目的是识别目标的身份。

3) 高层：在底层和中层的基础上，对识别到的目标进行行为分析，包括姿态识别、行为识别和事件分析。目的是令计算机可以和人一样理解目标的行为。

2.2.2 目标检测与跟踪

1) 目标检测

目标检测就是从图片中识别出特定物体的轮廓，并利用相关图像算法去除场景的背景，同时降低各种环境因素的变化对检测的影响，优化目标区域，从而得到只包含目标的前景图片。目标检测的目的是为后续的图像处理提供有效信息，是视频计算研究的基础。

根据检测原理的不同，可以将目前常用的目标检测算法分为三类：帧间差分法、光流法和背景减除法。

①帧间差分法[25][26]：

帧间差分法是一种常用的目标分割和目标检测方法，适用于背景相对固定的场景。帧间差分法的原理是在图片序列中相邻两帧或者三帧之间采用基于像素的时间差分，并通过阈值化确定图片中的运动区域。以基于两帧的帧间差分法为例，常用的方式是使用当前帧与前一帧相减，如公式(2-1)所示：

$$\Delta I_t = \left| I(x,y,t) - I(x,y,t-1) \right| \tag{2-1}$$

其中，$I(x,y,t)$和$I(x,y,t-1)$分别表示t，$t-1$时刻像素点(x,y)的灰度值。ΔI_t表示差值。假设 T 为事先设定的阈值，当$\Delta I_t < T$时，两帧帧差较小，场景基本无变化，表示此时无运动物体。当$\Delta I_t > T$ 时 代表该像素点为目标运动区域。这样通过计算两帧的差分值，就可以获得运动目标的二值图像$Z_t(x,y)$,如公式(2-2)所示：

$$Z_t(x,y) = \begin{cases} 1 & \left|\Delta I_t(x,y)\right| \geq T \\ 0 & \left|\Delta I_t(x,y)\right| \leq T \end{cases} \tag{2-2}$$

处理流程如图 2.2 所示。

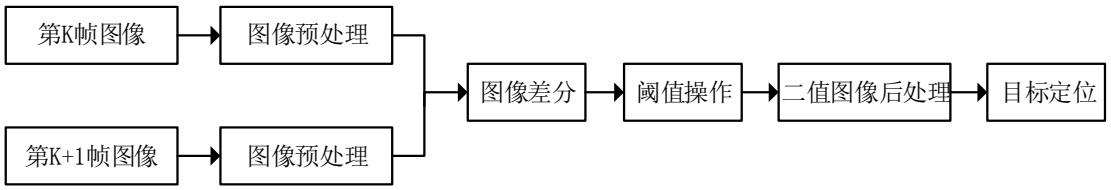

图2.2 帧差法处理流程示意图

帧间差分法容易实现，占用计算资源少，具有较好的实时性。但是对于环境噪声较为敏感，阈值的选择至关重要，过小则不足以压制噪声，过大则难以检测出运动目标。

②光流法

光流是指场景中运动物体的三维速度矢量在观察成像平面上的投影，该概念在1950年由Gibson首先提出来，是可见点在成像平面上对应像素点的瞬时速度。光流法[27]是根据连续的视频帧中相邻帧之间的相关性和像素在时域上的变化，从而计算出相邻帧物体运动信息的一种方法。光流法大致可以分为三类：基于匹配的方法，基于频域的方法和基于梯度的方法。

光流法本质上是通过计算图像中所有像素点的强度随时间的变化值来确定目标移动的方向和速度，其基本原理为：赋予图像中每一个像素点速度矢量属性，并通过投影计算将三维物体上的点与图像中的像素点逐一对应，由此形成一个运动矢量场，通过分析各个像素点的速度矢量特征来检测图像中的运动物体。如果光流矢量连续变化，则代表此时图像中没有运动物体。当图像中存在运动物体时，图像背景和目标之间的相对运动会造成背景的速度矢量与运动物体所形成的速度矢量有所不同，由此便可以计算出运动物体的位置。

值得注意的是，光流法虽然无需获得场景的先验知识，但是其复杂度较高，会消耗大量的计算资源，因此不适合实时视频计算任务。

③背景减除法

背景减除法[28]的基本思想是用当前图像减去背景模型得到运动物体所在区域，通常选取无运动物体的一帧或者综合多帧图像来作为场景的背景模型。在此类算法中，如何建立与更新背景模型是最为关键的问题。通常使用以下几种方法对背景进行建模：中值（均值）滤波的方法；高斯背景建模；非参数化背景模型；CodeBook背景模型等。

2) 目标跟踪

目标跟踪用来确定运动物体在连续视频帧序列中的轨迹，是视频计算中一个重要的环节。目标跟踪用来记录目标的运动参数以及运动轨迹，为更高层次的目标行为分析打下基础[29]。

目标跟踪算法根据应用场景的不同可以归结为两类：单场景目标跟踪和多场景目标跟踪。

①单场景目标跟踪

单场景目标跟踪是指在单个场景即单个摄像头拍摄的画面中跟踪特定物体，根据与目标检测的关系可将其归结为两类：一类是根据特定的策略，在目标检测的基础上跟踪特定物体，可以是基于轮廓[30]-[31],基于特征点[32]或基于核；另一类是同时进行目标的检测与跟踪，其原理是把目标跟踪看成是前景图像和背景图像的二分类问题。代表性算法有 Babenko 等人提出的基于多示例学习的跟踪方法[33]以及 Grabner 等人提出的一种基于在线特征提升的跟踪方法[34]-[35]。

②多场景目标跟踪

多场景目标跟踪是指在多路摄像头拍摄的画面中跟踪特定物体，通过为每一个运动物体设定一个独有的标识，从而达到在全局场景中跟踪物体的目的。

多场景目标跟踪根据场景是否重叠又可以归结为两种情况，非重叠场景的目标跟踪与重叠场景的目标跟踪。

2.2.3 目标识别

目标识别其本质上是一个分类问题，通过判断画面中各个物体所属的类别，进而确定目标身份。目标识别是高层计算机视觉算法的基础，在计算机视觉领域应用广泛，例如行人追踪、人脸识别以及图像大规模检索等。

近些年，在目标识别领域主要有两种方法得到了广泛使用，词袋模型和深度学习模型。

1) 词袋模型

词袋模型(Bag-of-words model)最早应用在信息检索以及自然语言处理领域。该模型忽略语法和语序等要素，独立地对待文本中出现的每个单词，从而将文本作为大量词汇的集合，通过对文本中不同单词出现的频率进行建模来表述文章。

2004 年，Gabriella 等人[36] 基于词袋模型提出了一种图像的分类方法，首次把词袋模型方法应用到计算机视觉领域中。逐步形成标准目标分类框架，该框架由四部分组成，分别是特征聚类、特征编码、特征提取以及特征汇聚和分类。

2) 深度学习模型

深度学习模型建立了类似于人脑的分层模型结构，每一个节点代表一个神经元，并通过引入反馈机制模拟人脑的认知过程。它不同于传统模式识别采用手工设计特征的方式，而是从大数据中自动学习特征，因此可以包含成千上万的参数。

2012 年，Hinton 研究小组使用深度学习方法赢得了 ImageNet 图像分类的比赛，准确度超出第二名 10%以上，在计算机视觉领域产生了极大地震动，掀起了深度学习

的热潮。目前，深度学习模型已经成为计算机视觉领域的研究热点。

2.2.4 行为分析

行为分析是通过对目标行为数据的分析来获取行为的语义描述和理解，目的就是让计算机能够像人一样理解目标的行为。相对于目标检测和识别，行为分析涉及到对人类视觉系统更深层次的理解，是更高层次的目标，同时也是计算机视觉领域要解决的终极问题之一。

2.3 行人识别算法

本文在搭建好面向大规模监控系统的视频架构的基础上，使用行人识别算法对某一场景的入侵检测行为进行了实现，同时结合实现中所遇到的问题，对当前行人识别算法误报率较高的问题进行了改进。为了文章后续章节的顺利展开，本小节对行人识别算法进行简要介绍。

行人识别即在视频帧中识别出有行人的区域，行人识别算法可用到很多场景，例如入侵检测，在一些场景中，在某些时间段不应该有人出现，因此当检测到行人时，需要进行异常报警。

行人检测算法经过了较长时间的发展[37]，早期主要以对图像分割、提取边缘特征以及运动检测等方法为主。例如：1) 以 Gavrila 为代表的基于轮廓的分层匹配算法，是一种全局模板方法；2) Broggi 通过使用大小不一的二值图像模板实现对人体肩部和头部的建模，对比模型与输入图像的边缘部分来识别行人，是一种局部模板方法；3) Heisele 提出的运动检测方法，是通过提取画面中行人的腿部特征来实现行人识别；4) Wohler 提出一种神经网络方法，通过构建自适应时间延迟神经网络来判断图片序列中是否包含行人；5) 以 Lipton 为代表的光流检测方法，通过计算图像中残余光流来实现行人检测。

然而早期的行人检测，误报率较高而且检测速度低。

当前行人检测主要采用基于统计学习的方法，通过训练大量样本，提取目标的一般特征，例如灰度、边缘、纹理等等，来构建行人检测分类器。基于统计学习方法的典型代表是基于 HOG(梯度直方图特征)+SVM（支持向量机）的行人检测算法[38]，该算法由法国研究员 Dalal 提出。

HOG 是一种对图像局部重叠区域的密集描述符，HOG 在一个网格密集的大小统一的细胞单元上进行计算，同时采用重叠的局部对比度归一化来提高性能。

SVM 是在分类与回归分析中分析数据的监督式学习模型与相关的学习算法。

目前 SVM 分类器结合 HOG 特征的图像识别方式已经被广泛用于计算机视觉领

域，特别是在行人检测领域取得了显著成果，其一般步骤为（使用 OpenCV 算法库）：

1) 准备训练样本集合：包括正样本集合和负样本集合，为了得到一个性能优良的分类器，训练样本越多越好。

2) 裁剪样本：收集到足够的样本集之后，手动截取正样本中的行人，并保存在一个文件夹中，将负样本保存在另一个文件夹中。

3) 提取所有正、负样本的 HOG 特征。

4) 为全部样本集赋予样本标签，例如设定正样本集的标签为 1，负样本集的标签为 0。

5) 将正负样本的 HOG 特征和标签，输入到 SVM 中进行训练，并将训练结果保存为文本文件。

6) 文本文件中有一个 support vector 数组，还有一个 alpha 数组，和一个 rho 浮点数。将 alpha 矩阵同 support vector 相乘，乘积结果为一个向量，在向量的最后添加 rho 元素，如此便得到了分类器。

7) 使用分类器对图片进行判断，识别其中是否有行人。

虽然目前该方法得到了广泛使用，但是其易受环境因素影响，误报率依旧可观，本文结合当前行人检测的方法，提出两种降低误报率的方法，将在第四章进行介绍。

2.4 计算机视觉库 OpenCV

视频是由大量连续的视频帧构成的，视频计算技术基于图像分析技术，通过分离场景中的前景和背景，来实现目标检测、识别和跟踪。图像分析技术一般通过智能图像算法并结合相关数学模型分析处理图像的底层特征，进而实现对图像的理解和分析。本文在实现部分主要采用开源的 OpenCV 库做视频计算工作，接下来将对 OpenCV 做简要介绍。

OpenCV[39]是由英特尔创立的一个开源的计算机视觉库，任何人都可以获得其源代码并免费使用。同时 OpenCV 跨平台，可以在 Windows、Linux、Unix 等多种操作系统中使用。

OpenCV 对计算机视觉领域以及图像处理方面的很多算法进行了实现，主要采用 C 语言和 C++语言进行编写，同时提供了 Ruby、Python 等多种语言的编程接口。

OpenCV 采用模块化的组织方式，具有很多静态或者动态库。OpenCV 提供如下模块供用户使用：

1) Core 模块。本模块定义了大量的基本数据结构供其他模块调用。

2) Imgproc 模块。图像处理算法模块，提供了大量图像处理算法，包括图像过滤算法，几何图形转换算法（放射，通用的基于表的重新映射，透视扭曲和大小调整），

色彩空间转换以及直方图等等。

3) Gpu 模块。提供针对不同模块的 GPU 加速算法。

4) Calib3d 模块。多视图算法模块，提供多视图几何算法、3D 构造算法以及立体相关算法。

5) Video 模块。视频分析模块，包含对象跟踪、运动评估以及背景提取等算法。

6) Objdetect 模块。对象检测以及预定义类的检测。

7) Highgui 模块。包括捕获视频接口、相关编码接口以及基本的图形化界面。

8) Feature2d。特征值检测框架，提供大量特征值描述子以及检测器。

本文在实现过程中，主要使用 OpenCV 的 Core 模块、Objdetect 模块以及 Highgui 模块。

2.5 本章小结

本章深入研究了“平安城市”监控系统的两种建设方式，并对智能视频监控系统中使用的视频计算技术进行了研究，详细介绍了平台实现过程中涉及的行人识别算法以及计算机视觉库 OpenCV，为第三章和第四章的展开奠定了基础。

第三章　面向大规模监控系统的视频计算架构

为了降低通信带宽消耗，保证计算实时性，本章设计一种边缘化的视频计算架构，主要包括两部分，中心节点和边缘节点。中心节点，是架构的控制模块，统一管理各个边缘节点，同时负责计算任务的管理；边缘节点，从前端摄像头取视频流进行处理，承担整个架构的视频计算工作任务。并且，本章将对该模型的任务调度策略和计算能力动态调整策略做详细分析。

3.1 计算架构

当前“平安城市”监控系统的建设方式以分布式方式为主，各个区域自建了大量封闭的监控系统，具有大规模、高自治的特点。视频计算任务不仅消耗大量的计算资源同时对实时性要求较高，如果采用集中式的架构模型，不仅会增加数据中心的负担，导致计算能力不足，同时也会给网络输出带来巨大的压力，造成网络延迟过长、实时性无法得到满足、服务质量下降等问题。

3.1.1　边缘化视频计算架构

为了将计算能力带向离数据源更近的地方，本文结合当前平安城市监控系统的建设模式，提出一种边缘化的视频计算架构，将计算任务迁移到网络边缘设备，以提高数据的网络输出性能，保证数据处理的实时性。其示意架构如图 3.1 所示。

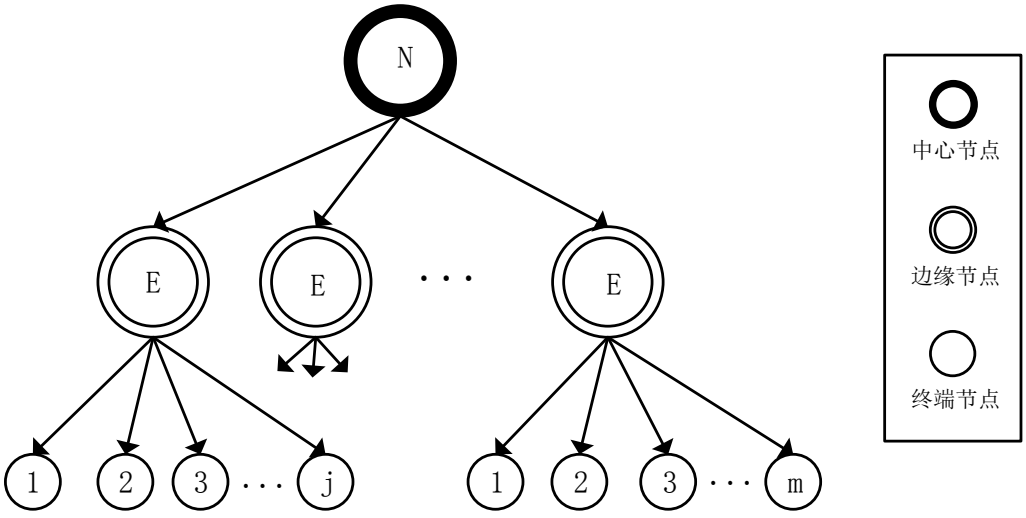

图3.1 视频计算架构示意图

终端节点即为前端摄像头，负责采集实时画面；边缘节点是整个模型的计算模块，

它围绕数据而建，承担着整个架构的计算任务。中心节点是是整个系统的中央枢纽，统一调度、管理所有的边缘节点，同时负责任务的派发和信息的整合。

3.1.2 节点部署方式

1) 边缘节点部署

视频计算不仅消耗大量的计算资源，而且对数据传输能力也是一个巨大的挑战。以一台清晰度为 720p 的网络摄像机为例，当使用 H.264 编码格式对视频流进行编码时，对传输码率的最低要求是 2Mbps。如果一个小区包含有 15 台网络摄像机，以 2Mbps 的速率，则会引起 30Mbps 的上行流量消耗，而在计算节点同样会造成 30Mbps 的下行流量消耗，会出现“双向消耗”的现象。因此网络 IO 能力同样会成为视频计算瓶颈。

为避免 IO 过载问题，本文采用“以数据为中心”边缘化的节点部署策略，就近部署计算节点，避免不必要的数据传输。

同样以上面包含有 15 台网络摄像机的小区为例，可以将计算节点就近部署在小区的机房内，目前以太网技术已发展到了千兆位太网，30Mbps 的网络流量在以太网内不会造成压力。计算节点可以直接访问摄像头或者相关流媒体设备采集视频数据，不用将视频数据上传到广域网，避免了“双向消耗”，同时降低了网络延迟，保证了实时性。

2) 中心节点部署策略

中心节点作为整个模型中的中央枢纽，承担了调度、管理所有边缘节点的任务，因此中心节点宜采取集中部署的方式。以某城市为例，如果在此城市部署视频计算平台，则在该城市设立一个中心节点，在各个区域设置边缘节点，所有边缘节点连接到中心节点之上，接受中心节点的调度管理。并且，中心节点可进行分层扩展，即设立不同层级的中心节点，以某省为例，可在该省设立一个一级中心节点，各城市设立二级中心节点，一级节点通过二级节点，进而调度边缘节点。通过此种方式，可以简单快捷完成计算架构的扩展。

3.2 任务调度策略

3.2.1 分层式任务调度

任务调度就是根据任务信息采用适当的策略把不同的任务分配到相应的资源节点上运行，好的任务调度策略可以降低任务的执行时间、增加系统的吞吐量。按架构模式可将调度系统分为三大类：集中式调度、分布式调度、分层式调度[40]。

集中式调度：由一个主节点和多个计算节点构成。主节点负责全局资源管理，负

责整个系统的资源调度。

分布式调度：在该种调度方式中，存在多个本地调度节点，每个本地调度节点管理多个计算节点，同时这些本地调度节点彼此通信，相互协作将任务分配给计算节点。

分层式调度：存在一个全局调度者和多个本地调度者。全局调度节点将任务下发给本地调度者，本地调度者根据任务和本地资源情况进行匹配与调度。

由第二章介绍的内容可知，目前平安城市建设的推进过程中，主要以分布式存储的方式为主，具有大规模、非集中的特点，各个边缘节点具有较高的自治性。考虑到以上特点，本文采取分层式调度方式作为整个系统的调度策略。

任务调度策略示意图如图 3.2 所示。

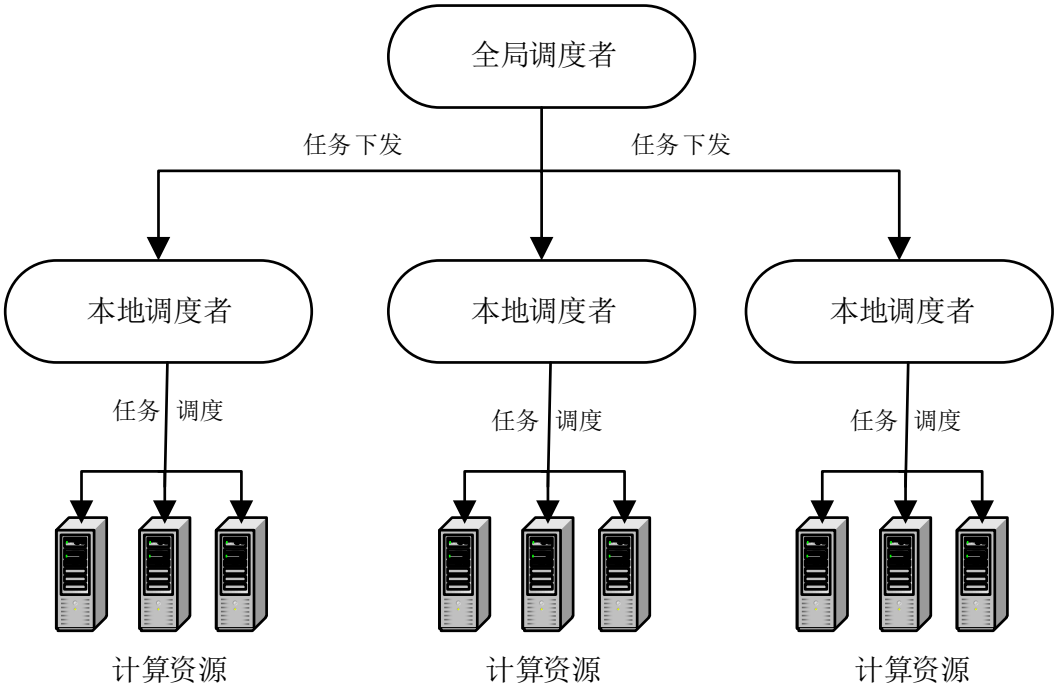

图3.2 任务调度策略示意图

3.2.2 全局调度者

在本模型中将中心节点作为全局调度者，负责全局资源管理和任务调度。中心节点维护一个任务队列，存储待计算的任务，当有任务提交到中心节点时，中心节点将该任务添加到任务队列，由中心节点根据任务信息下发到相应本地调度节点（在本模型中本地调度者属于边缘节点），由本地调度节点调度计算资源执行视频计算任务。其中任务信息主要包括需要执行此任务的边缘节点 ID，任务类型（例如行人追踪、入侵检测、车牌识别等）以及其他任务数据。

中心节点作为全局调度者，还负责与用户进行交互，包括接收用户提交的任务请

求以及向用户反馈视频计算结果。此处以人脸识别任务为例，其调度流程可描述为如下步骤：

步骤 1：用户通过交互界面，向中心节点提交计算任务信息，包括人脸识别任务类型 ID，待执行此任务的边缘节点 ID 以及待检测的人脸模型数据（假设人脸识别任务 ID 为 A1，待执行此任务的边缘节点 ID 为 E1）。

步骤 2：中心节点将此任务添加至任务队列，等待调度。

步骤 3：中心节点从任务队列中取出该任务，根据任务信息将该任务类型 ID 以及人脸模型数据发送给边缘节点 E1。

步骤 4：接收边缘节点 E1 的计算结果包括匹配到目标人脸的时间、位置等，通过交互界面展示给用户。

3.2.3 本地调度者

在本模型中本地调度者属于边缘节点，边缘节点使用本地调度者接受中心节点下发的计算任务，并根据任务信息调度本节点中的计算资源执行计算任务。在 3.2.1 节已经对本模型使用的分层调度策略进行了介绍，本小节将设计本地调度者所使用的任务调度算法。

任务调度算法可分为静态调度和动态调度两种方式[41],其区别在于任务信息与资源信息的映射关系是否是固定不变的。静态调度的方式，任务信息与资源信息的映射关系已经事先确定好，并且保持不变，当有计算任务到达时，调度算法将计算任务映射到事先已经配置好的资源上。静态调度算法无法实时感知计算资源情况，容易造成某个计算节点负载过大，灵活性较差；动态调度是指在决策时刻，根据任务和资源信息，不断更新调度结果。

本文使用动态调度算法，基于计算节点的负载程度，采用“低负载，先调度”的调度方法。本地调度节点维护一个计算节点负载队列，保存各计算节点的负载程度，由计算节点实时上报，当有计算任务到达时，选取负载最低的计算节点优先执行计算任务。调度速度快，同时可保证各个计算节点负载均衡。

1) 负载计算方法

通常使用两个参数来衡量系统的负载情况，执行任务所消耗的时间（*ElapsedTime*）和内存使用情况（*Memory*）。由于视频计算实际上是对视频流中的每一帧图片进行计算，而且监控视频是实时的，所以每一帧图片的检测时间过长，会导致后续视频无法检测，最终造成内存占用率升高。综上考虑，本文采用一帧图片所消耗的 CPU 时间（*ElapsedTime*）来衡量一个计算节点的负载情况。

系统每隔一段时间对 *ElapsedTime* 进行采样，然后使用平均值来表示机器的负载情况。

$$AverageTime = (\sum_{i=1}^{n} ElapsedTime_i) / n \tag{3-1}$$

其中，*AverageTime* 表示平均值，$ElapsedTime_i$表示第 i 个数据采样。通过对多个样本数据求平均值，使数据更具代表性和准确性。

但是，由于各个计算节点的配置不同，例如 CPU 的核数，内存大小等等，所以即使同一张图片在不同计算节点的计算时间也会不同。进行任务调度时，需要比较各个计算节点的负载情况，如果采用 *AverageTime* 进行横向比较，将会有失“公正”。本文采用计算“负载率”的方法，横向比较所有计算节点的负载情况。

$$Load = \frac{AverageTime}{StatisticalTime} \tag{3-2}$$

其中 *AverageTime* 是公式（3-1）的计算结果，*StatisticalTime* 是计算节点在正常情况下分析一张图片所消耗的 CPU 时间（通过实验获得）。*Load* 代表每一个计算节点的负载率，*Load* 越小说明负载越小。调度节点进行任务调度时，选取 *Load* 值较小的计算节点执行视频计算任务。

2) 任务算法描述

考虑到视频计算的任务大多不具有优先级的区别，本文对任务采取先来先服务（FCFS，First Come First Served）的方式,本地调度者维护一个任务队列，按照任务到达的先后顺序进行排序，进行任务调度时，首先调度较早到达的任务。此种方式易于实现并且较为公平。其算法描述如算法 3.1 所示。

算法 3.1 本地调度者任务调度算法

```
begin
2    while length(TaskList) > 0      //直到所有任务被调度完
3    do
4        tk ← firstArrived(TaskList)      //选取第一个到达的计算任务
5        mj ← lowestLoad(LoadList)    //选取负载率最低的计算节点
6        scheduleTask(mj,tk)      //将任务 tk 交由 mj 执行
7        removeTask(TaskList,tk) //从任务集合中删除 tk
8        updateLoad(LoadList)      //更新计算资源负载列表
end
```

3.3 计算资源动态调整策略

视频计算是一项计算密集型的工作，需要对视频每一帧画面逐像素分析，计算量巨大。以帧率为 25 帧/秒的实时监控视频流为例，假设视频分辨率为 1280*720,则每帧图像有 921600 个像素，每秒接收到的像素总和是 2304000，一般相似性计算大约需要 60 个计算，则每秒需要近 14 亿次运算操作。可见，视频计算会造成巨大的计算资源消耗，尤其在面对大规模的监控视频时，必然会对系统的计算能力提出巨大的挑战。

另外，视频计算对计算能力的要求并不是一成不变的，会随着图像清晰度、图像质量的变化而动态改变。以基础的人脸检测应用为例，假设模型文件和图像的黑暗度不变，人脸识别时间会随着图像中人脸数量的增加而增加；假设模型文件和人脸数量不管，人脸识别时间会随着黑暗度的变化而变化，如表 3.1，表 3.2 所示。（实验严格控制在 Interl® CPU E5-2600 @1.8GHZ 环境下进行）

表3.1 人脸数不同时检测时间对比试验

人脸数目	检测时间（ms）
一张人脸	300
二张人脸	380
四张人脸	520

表3.2 光照条件不同时检测时间对比试验

光照条件	检测时间（ms）
正常	310
黑暗	390

当计算任务骤增或者计算任务对计算资源的需求发生抖动时，会增大计算节点负载，影响系统稳定，因此计算能力动态调整策略至关重要。计算能力动态调整策略包括两部分，自适应视频帧丢弃算法和动态增加计算性能算法。当系统出现计算能力不足时，首先启用自适应视频帧丢弃算法，丢弃非重要摄像头的部分视频帧，使计算资源向重要摄像头倾斜，保证系统的稳定。后期通过增加计算节点，提升计算性能，使所有计算任务得到充足的计算资源。

3.3.1 视频帧丢弃算法

在监控系统视频分析中，监控系统的视频价值密度普遍较低，在连续不间断的视频监控过程中，可能有用的数据仅仅有一两秒。同时，各路监控视频的重要程度也不

尽相同，以学校为例，学校门口监控视频的信息量明显大于一些人员较少区域的监控视频。当计算节点出现负载过高，计算能力不足时，可以考虑舍去一些重要程度较低的数据，使计算资源向重要数据倾斜，从而保证重要数据得到充足的计算资源，同时可以保证系统的稳定性。

使用视频帧丢弃策略，需要给各路摄像头添加权重信息 Weight，本文规定权重由整形数值表示，范围由 1 到 100，数值越大重要程度越高。

视频帧丢弃策略基本思想：计算节点不断获取自身负载信息（已在 3.2 节进行介绍）进行判断，如果系统负载过重则触发视频帧丢弃策略。首先，通过配置文件获取视频流的权重信息（提前写入配置文件）并根据权重大小对视频流排序，选取其中权重最小的视频流（如果含有多个，则随机选取其中一个），将其视频流帧数丢掉二分之一，权重加 1，并将该视频流信息放入列表中，方便以后对该视频帧执行加帧策略；同时，继续观察系统负载情况，如果系统负载依然过重，则再次选取权重最小的视频流，将其视频流帧数丢掉二分之一，并继续进行；如果系统负载得到缓解，则停止丢帧策略；如果系统负载不断减小，当低于某个阈值（该值根据程序运行环境进行设定，通常可设为正常负载的百分之八十）时，则可以采取加帧策略。

在视频帧丢弃策略里，为了避免一些视频流被频繁丢帧，最终无法得到计算资源的现象发生，在每次对一路视频流执行丢帧策略后，将其权重加 1，避免其被频繁丢帧。其算法描述如算法 3.2 所示。

算法 3.2 视频帧丢弃算法

```
begin
1    Ln ← getNormalLoad()//获取系统正常负载指数
2    Lo ← getLowThreshold()//获取低负载阈值，当低于该值时，执行加帧策略
3    W ← getStreamWeightList()//获取视频流权重列表
4    D ← createDropList()//创建列表 D，存储被丢帧的视频流信息
5    while true
6    do
7        L ← getLoadInfo()//获取当前系统负载
8        while L > Ln
9        do
10           S ← getLowStream(W)   //获取权重最小的视频流
11           throwAwayFrame(S)
12           increaseWeight(S)          //增加 S 的权重
13           addElement(S,D)            //将被丢帧的视频流信息保存到列表中
```

```
14          L ← getLoadInfo()
15      while L <= Lo
16      do
17          if length(D) = 0
18          then
19              break
20          S ← getStream(D)//从列表中取出被丢帧的视频流信息
21          addFrame(S)   //为视频流 S 加帧
22          deleteElement(S,D) //从列表中删除 S
23          decreaseWeight(S) //降低 S 的权重值
24          L ← getLoadInfo();
end
```

3.3.2 动态增加计算资源

在大规模的视频计算系统中，动态增加计算资源是一项非常重要的特性，可以增加系统的吞吐量。主要分为两种方式：横向扩展和纵向扩展。

纵向扩展的本质是硬件的升级，将原有计算节点的服务器更新换代，例如将 16 核的处理器升级为 32 核甚至 64 核、增加服务器的运存等等。采用这种方式，不需要对原有系统做较大变动，简单易行。然而每台服务器的扩展能力有限，采用此种方式无法无限扩展计算资源，并且成本开销较大，更为严重的是，需要将待扩容的服务器停机以更新硬件设备，将会影响正在进行的计算任务。

本文使用横向扩展的方式增加系统的计算资源。横向扩展的本质是增加服务器的数量，将大量的计算任务分摊到更多的服务器上，从而降低每台服务器的计算负载，其详细步骤为：

步骤 1：新计算节点 m_j 在本地调度节点注册，设置 m_j 负载为 0，转步骤 2；

步骤 2：选取当前负载最高的计算节点 $m_{k,}$，转步骤 3；

步骤 3：通知 m_k 停止当前处于丢帧状态的计算任务，并将其分配给 m_j 进行计算，更新 m_k 和 m_j 负载，转步骤 4；

步骤 4：判断当前是否存在过载的计算节点，否，停止；是，转步骤 5；

步骤 5：判断 m_j 是否过载，否，转步骤 2；是，转步骤 6；

步骤 6：系统计算能力仍然不足，需要继续添加服务器数量，停止。

其示意图如图 3.3 所示。

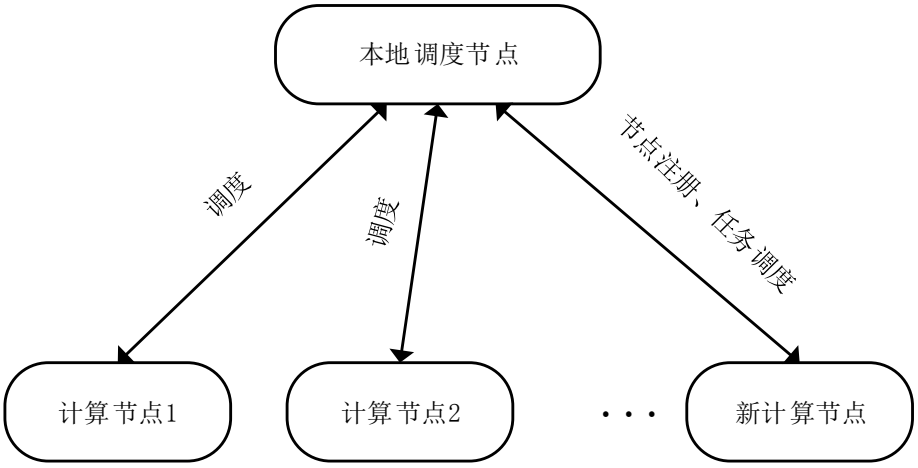

图3.3 动态增加计算节点示意图

3.4 计算架构分析

面向大规模监控系统的视频计算架构具有以下优势：

1) 扩展性

通过增加边缘节点，可以轻松实现计算平台范围的扩展。以某城市为例，如果欲将某学校纳入其计算平台范围，只需在该地点添加边缘节点，并将此边缘节点与中心节点连接即可，易于实施而且不会对原有系统造成影响。

2) 计算能力动态调整

当边缘节点计算能力不足时，执行视频帧丢弃策略，保证高权重的摄像头得到充足的计算资源；当出现数据丢弃情况时，可以通过增加服务器数量提升计算能力。

3) 低带宽消耗

边缘节点在本地对视频流进行分析处理，根据中心节点的任务命令，只需将有价值的视频信息发送到中心节点，降低了通信带宽的消耗。

3.5 本章小结

本章针对大规模视频分析所面临的数据传输瓶颈以及计算能力瓶颈问题，提出一种边缘化、分布式的视频计算架构，并对模型进行了详细剖析。设计了模型的任务调度方式，并提出计算能力动态调整策略，保证系统的稳定性以及充足的计算能力。

第四章　基于行人检测算法的修正方法研究

本章结合目前视频计算中常用的行人检测算法，针对其误报率较高的问题提出两种方法对其进行优化。

4.1 人员检测算法问题分析

目前常用基于统计学习的方法进行行人检测，其中的经典代表是法国研究员 Dalal 提出的基于梯度直方图特征（HOG）和支持向量机（SVM）的行人检测算法。详细步骤已在第二章进行介绍。

虽然目前该方法得到了广泛应用，但算法的识别效果受外界条件影响会产生波动，造成漏检率和误报率上升。本文着眼于误报率，通过研究对识别效果造成影响的外界因素，提出两种降低误报率的方法。

使用 OpenCV 自带的行人检测算法对某小区的停车棚做测试，测试结果如图 4.1 所示。

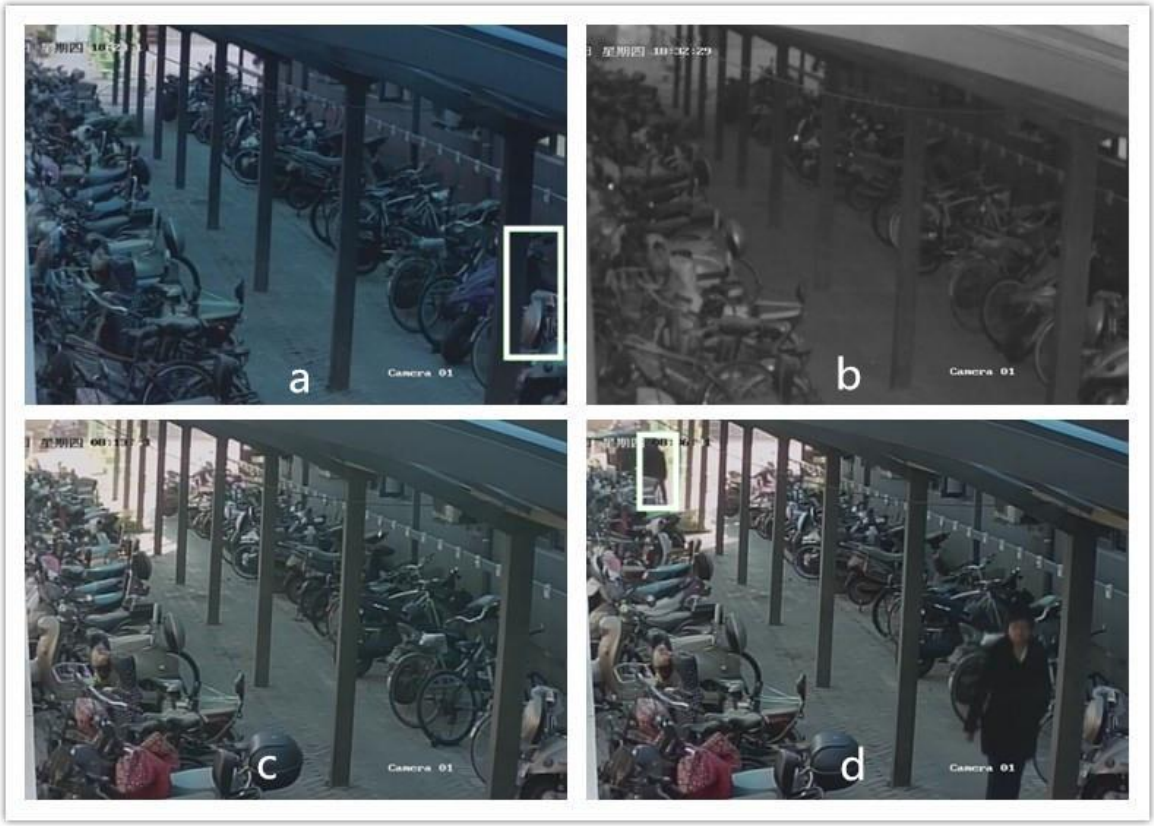

图4.1 OpenCV 行人检测算法测试结果

其中，(a)(b)是对车棚白天和夜间的测试结果，图中方框区域是算法识别到有人的位置。夜间误报情况较少，分析原因为，夜间光照条件差，图像灰度值较高，色彩辨析困难，造成行人检测困难但同时也降低了误报率。可见当光照发生显著变化时，行人检测算法的识别效果会产生波动。

(c)(d)是在光照相同情况下的测试结果。由于背景发生了变化，导致一些非人的物体被算法误报为人，更为严重的是，若(d)图背景一段时间内保持不变，则误报情况会持续发生，大量的误报情况会给用户造成困扰。

目前很多研究通过调整参数，针对特定场景做特定调整的方式来降低误报率。本文从另一种角度出发，针对光照和位置这两个容易对识别效果产生影响的因素，提出两种降低误报率的方法：基于贝叶斯推断的人员检测结果修正方法和基于可信度的自适应行人检测方法。

4.2 基于贝叶斯推断的人员检测结果修正方法

基于贝叶斯推断的人员检测结果修正方法，是通过贝叶斯分类对行人识别算法所作出的有人入侵的结果做二次判断，最终判断是否有人入侵。本方法基于贝叶斯分类算法，因此本小节首先针对贝叶斯分类算法做简要介绍，然后详细介绍本方法的具体步骤。

4.2.1 贝叶斯分类算法

1) 贝叶斯定理

贝叶斯定理是关于随机事件 A 和 B 的条件概率的一则定理，解决了现实生活中常遇到的问题：已知某条件概率，如何得到两个事件交换后的概率，贝叶斯公式：

$$P(B|A)=\frac{P(A|B)P(B)}{P(A)} \tag{4-1}$$

其中，若事件 B 可以划分为 B_1，B_2，…，B_N 个完备事件组，并且他们互不相容，则根据全概率公式，可得出贝叶斯公式的另一种形式：

$$P(B_i|A)=\frac{P(A|B_i)P(B_i)}{\sum_{j}^{n}P(A|B_j)P(B_j)} \tag{4-2}$$

2) 朴素贝叶斯分类

朴素贝叶斯分类的基本思想是：对于待分类项，对其可能出现的结果进行类别划分，并计算在待分类项出现的条件下，各个类别出现的概率，选取条件概率最大的类

别作为待分类的类别。其正式定义如下：

①设$x=\{a_1,a_2,\dots,a_m\}$为一个待分类项，而每个a则为x的一个特征属性。

②有类别集合$C=\{y_1,y_2,\dots,y_n\}$。

③计算$P(y_1|x),\ P(y_2|x),\dots,\ P(y_n|x)$。

④如果$P(y_k|x)=max\{P(y_1|x),P(y_2|x),\dots,P(y_n|x)\}$,则$x\in y_k$

从上述步骤可以看出，关键是计算步骤③中的各个条件概率，具体可以采取以下方式进行计算：

①选取一个已知分类的待分类项集合，称为训练样本集。

②统计各个类别下的各个特征属性的条件概率值：

$P(a_1|y_1),\dots,P(a_m|y_1);P(a_1|y_2),\dots,P(a_m|y_2);..;P(a_1|y_n),\dots,P(a_m|y_n)$

根据贝叶斯定理，可以得到如下推导：

$$P(y_i\,|\,x)=\frac{P(x\,|\,y_i)P(y_i)}{P(x)} \tag{4-3}$$

当特性为x时，计算所有类别可能出现的条件概率，选取条件概率最大的类别即可。由公式可以看出，对于每个类别而言，分母是一样的，所以计算时可以不考虑分母，只计算$P(x|y_i)P(y_i)$即可：

$$P(x\,|\,y_i)P(y_i)=P(a_1\,|\,y_i)P(a_2\,|\,y_i)...P(a_m\,|\,y_i)P(y_i)=P(y_i)\prod_{j=1}^{m}P(a_j\,|\,y_i) \tag{4-4}$$

4.2.2 修正分类器

本方法的核心是修正分类器，本小节主要介绍得到修正分类器的方法。

1) 结果分类

修正分类器是针对检测分类器的检测结果进行再判断，判断检测分类器得出的有人结果是否正确。因此可以将检测分类器的输出结果分为正确与错误两类。其中，错误用 0 表示，正确用 1 表示。结果的类别集合用C表示，则$C\in\{0,1\}$。

2) 确定特征属性及划分

检测分类器得出的有人结果，可以从多个角度进行属性划分，修正分类器根据多个特征属性计算先验概率。本方法使用时间、人员位置这两个特征对分类器得出的有人结果进行划分。

①时间

不同的光照会对检测分类器的准确度产生影响，而一般情况下光照条件会随时间周期性变化，所以提取时间特征。时间特征用 t 表示，将一天分为 24 个时间段，则

$t \in \{1,2,\dots 24\}$。如 00:15 可表示为$t = 1$,表示在第一个时间段。

②人员位置

在场景中经常会有一些非人的物体被检测分类器误报为人，而且这些物体在通常在一段时间内位置固定，造成检测分类器在该位置的识别结果准确度较低，所以提取入侵人员的位置特征。位置特征用 d 表示，表示检测分类器识别到人的位置（左上角坐标）相对于整个场景的位置。如果把整个图片分成θ个区域，则$d \in \{1,2,\dots \theta\}$。

3) 训练样本

收集大量检测分类器得到的有人结果图片，提取其中的时间、人员位置、人员大小特征，并人工判断检测分类器得出的结果是否正确，对每一个结果进行分类，0 或者 1。

计算训练样本中的每个类别的频率，即计算$P(C = 0)$与$P(C = 1)$。

计算每个类别条件下各个特征属性划分的频率,即计算$P(t|C = 0)$、$P(d|C = 0)$、$P(t|C = 1)$、$P(d|C = 1)$。

至此，已经得到修正分类器的各个参数。

4.2.3 方法组成

本方法并不对原有算法进行修改,而是在其基础上添加基于贝叶斯推断的修正分类器，因此本方法主要由以下三个部分构成：

1) 训练部分：提取样本集的 HOG 特征，输入到 SVM 分类器进行训练，得到人员检测分类器（以下称为检测分类器）。

2) 识别部分：使用检测分类器对连续的视频帧进行判断，判断场景中是否有人入侵。

3) 修正分类器：该模块基于贝叶斯定理，收集大量的样本集，使用贝叶斯定理对样本集进行训练得到贝叶斯分类器（以下称为修正分类器），使用修正分类器对检测分类器的识别结果进行修正，降低检测分类器的误判率。

4.2.4 基于贝叶斯推断的人员检测结果修正方法

为了进一步阐明本方法的目的和技术方案，本小节将介绍本方法的详细步骤：

步骤 1：针对摄像头场景准备样本集合，包括正样本集和负样本集。正样本集为多姿态多尺度不同衣着的人的截图，负样本是该场景中所包含的各种背景。

步骤 2：计算正样本集和负样本集的 HOG 特征，使用 SVM 分类器对特征数据进行计算，得到行人检测分类器，并初始化各个系统参数。

步骤 3：截取视频流的单帧图片，计算图片的 HOG 特征，使用步骤 2 中得到的行人检测分类器进行计算，判断图片中是否有人，如有没有，则继续捕捉下一帧图片

进行相应操作；否则转到步骤 4，如果修正分类器训练完成，则转步骤 5。

步骤 4：由用户人工判断检测分类器的判断结果是否正确，并将用户的判断结果作为训练输入，使用 4.1.3 章节中的方法，得到修正分类器。

步骤 5：使用修正分类器对检测分类器得出的有人结果进行修正。首先提取检测分类器判断结果的特征集，即时间、人员位置。然后使用步骤 4 得到的各个概率值计算$P(X|C=0)P(C=0)$和$P(X|C=1)P(C=1)$。其中，

$$P(X|C=0)P(C=0)=P(t|C=0)P(d|C=0)P(C=0) \tag{4-5}$$

$$P(X|C=1)P(C=1)=P(t|C=1)P(d|C=1)P(C=1) \tag{4-6}$$

由 4.1.1 章节的朴素贝叶斯分类方法可知，如果$P(X|C=0)P(C=0)$大于$P(x|C=1)P(C=0)$，则表明检测分类器的识别结果错误，场景中无人员入侵；反之，则表明检测分类器的识别结果正确，场景中有人员入侵。

基于贝叶斯推断的行人检测结果修正方法的示意图如图 4.2 所示。

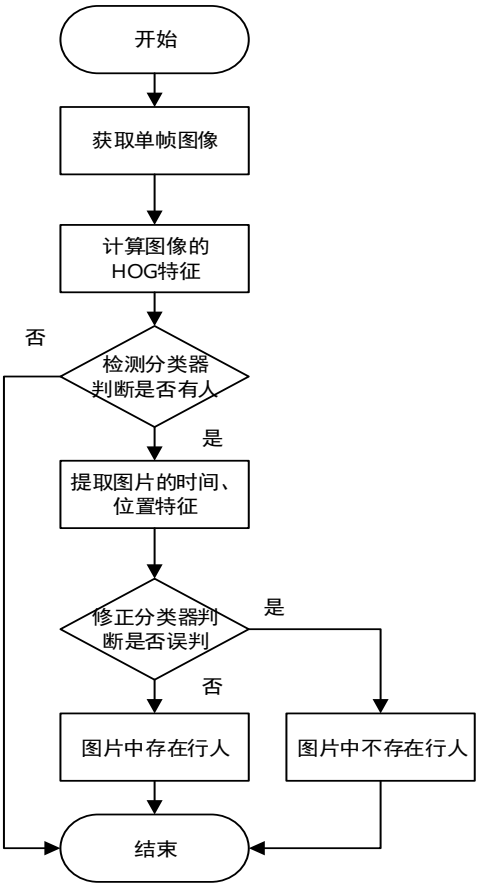

图4.2 基于贝叶斯推断的行人检测结果修正方法示意图

4.2.5 方法分析

本方法使用朴素贝叶斯分类，基于先验概率，通过对历史数据集的分析，得到修正分类器，对检测分类器的结果进行二次判断得到最终结果。该方法易于实现，且有稳定的分类效率。然而该方法具有以下的局限性：

1. 朴素贝叶斯分类假设各属性之间相互独立，如果选取的属性存在某种相关性，分类效果将受到影响。

2. 修正分类器通过对历史数据进行分析，计算各先验概率得到。如果场景中环境，例如光照条件随时间变化的规律，背景等，发生较大变化，则当前修正分类器将不适用此种场景，需要重新训练修正分类器。

4.3 基于可信度的自适应人员检测方法

目前常用的人员入侵检测系统，是在一些无人值守的区域设置摄像头，并通过行人检测算法对摄像头采集的视频进行实时分析，当检测到有行人通过时，对当前图片进行截图，并将图片发给用户，进行提醒。然而目前的系统存在以下两个问题：

1) 分类器缺乏持续学习的能力

行人检测分类器，是通过采集大量有人的图片做为正样本集，大量无人的图片作为负样本集，提取两种样本集的 HOG 特征，再通过 SVM 分类器训练得到行人检测分类器。一般为了提升准确度，通常截取相应场景的图片作为样本集来训练。然而分类器在系统初始时就已固定，不会进行动态更新。这样，针对一些背景变化的场景，比如停车场，则无法很好地适应新场景，造成误报率升高。

2) 识别结果易受环境因素影响

行人检测分类器受环境因素的影响，例如光照，会造成识别结果准确度抖动，具体表现为在某些时段识别效果较好，在某些时段识别效果较差。

以上两个问题造成当前行人入侵检测系统效果不理想，误报率较高，具体表现为用户收到很多系统发来误报的报警图片，并且由于用户无法将系统误报的情况反馈给系统，最终导致系统重复的对某一区域造成误报，严重影响识别效果。通过分析现存问题，本文提出一种基于时间位置复合可信度的自适应人员入侵检测方法。本方法通过引入用户的反馈，对系统的分类器和可信度进行动态更新，从而不断优化识别结果。

4.3.1 时间位置复合可信度

由于分类器的识别结果无法保证一定正确，造成了大量误报的情况。本文通过给分类器赋予可信度的方式，对分类器的识别结果进行可信度判断，综合分类器的识别效果和可信度最终确定当前是否有人员入侵情况发生。

受外部因素影响，分类器识别结果的可信度并不是固定的，本文从时间和位置两个维度对分类器的可信度进行划分。

1) 时间位置复合可信度的定义

由于分类器的识别效果易受光照的影响，而光照情况通常和时间关联，所以本文首先从时间维度对分类器的可信度进行划分，在不同的时间分类器的识别结果具有不同的可信度。用X_t表示在t时间内，分类器的可信度，如果将一天划分为ω个时间段，则t的范围是$1\sim\omega$。

同时，一些非人的物体很容易被分类器误判为人，例如椅子。假如在某些场景中存在一些非人并且易被误判为人的物体，该物体在场景的位置为d，假如该场景在某段时间内保持不变，则分类器很容易误判d位置识别有人，造成分类器的识别结果在该位置的可信度很低。所以，本文从空间的维度对分类器的可信度进行划分，在场景中的不同位置，分类器的识别结果具有不同的可信度。例如，用X表示分类器的可信度，则X_d表示分类器的识别结果在d位置的可信度。假如把整个图片分成θ个区域，则d的范围为$1\sim\theta$。

从时间和空间的维度对分类器的识别结果进行划分，则对应一个场景，分类器共有$\theta * \omega$个可信度。X_{td}则表示分类器的识别结果在t时间，d位置的可信度。

2) 时间位置复合可信度的计算方式

时间位置复合可信度是根据用户的反馈进行动态更新的，根据用户的反馈，如果分类器的判断正确，则可信度上升；如果分类器的判断错误，则可信度下降。用户不断反馈，则可信度不断更新。

这里给出时间位置复合可信度的计算公式：

$$X_{td} = \frac{T_{td}}{T_{td} + \lambda F_{td}} \tag{4-7}$$

其中，T_{td}表示分类器在t时间，d位置处结果判断正确的次数，初值为 10。F_{td}表示在t时间，d位置处判断错误的次数，初值为 0。λ为调节参数，可以根据不同的应用场景调节错误反馈在可信度中的影响。当$\lambda = 1$时，X_{td}趋近该时间位置组合的实际正确率。

4.3.2 方法组成

本方法并不针对原有行人检测算法做修改，而是在其基础上添加了基于时间位置可信度的决策部分和反馈部分，所以本方法主要有以下四个部分构成：

1) 训练部分：收集样本图片（包括正样本和负样本），计算这些样本的梯度方向直方图（HOG），训练得到图片分类器。

2) 识别部分：计算需要识别的图片的 HOG 特征，将结果作为输入传入分类器进

行判断。判断输入图片中是否有行人存在，如果存在行人可以识别出其位置和大小。

3) 决策部分：决策部分根据时间位置复合可信度，对分类器得出的有人侵入的结果进行可信度判断，综合识别模块和决策部分的识别结果，最终得出是否有人侵入。

4) 反馈部分：将决策部分判定后的图片，发送给用户，用户进行反馈，分类器是否有误判。如果没有误判，则提升相应的时间位置复合可信度。如果发生误判，则保存该负样本图片并降低相应的时间位置复合可信度。

4.3.3 基于可信度的自适应人员入侵检测方法

为了进一步阐明本方法的目的和技术方案，本小节将介绍本方法的详细步骤：

步骤 1：针对摄像头场景准备样本集合，包括正样本集和负样本集。正样本集为多姿态多尺度不同衣着的人的截图，负样本是该场景中所包含的各种背景。

步骤 2：计算正样本集和负样本集的 HOG 特征，使用 SVM 分类器对特征数据进行计算，得到行人检测分类器，并初始化各个系统参数。

步骤 3：截取视频流的单帧图片，计算图片的 HOG 特征，使用步骤 2 中得到的行人检测分类器进行计算，判断图片中是否有人，如有没有，则继续捕捉下一帧图片进行相应操作；如果有则转步骤 4。

步骤 4：根据步骤 3 中所识别到有人的时间 t 和人的位置 d，查找相应的可信度 X_{td},根据X_{td}确定步骤 3 的识别结果是否可信。如果可信则转步骤 5；否则，转步骤 2。其判断方式为：我们用伪随机数模拟概率性决策过程，首先，将系统当前时间作为输入，通过伪随机数生成器生成伪随机数R，并让R对 100 取余，得到$R_1 = R \ mod \ 100$。若$X_{td} < R_1$结果可信，否则不可信。

步骤 5：经过步骤 4 之后，对用户进行人员入侵检测提醒，同时由用户判断步骤 4 的识别结果是否正确，并对系统进行反馈。根据公式(4-8)可知，如果此次判断正确，则对应的可信度上升；否则，下降。

基于可信度的自适应人员入侵检测方法的示意图如图 4.3 所示。

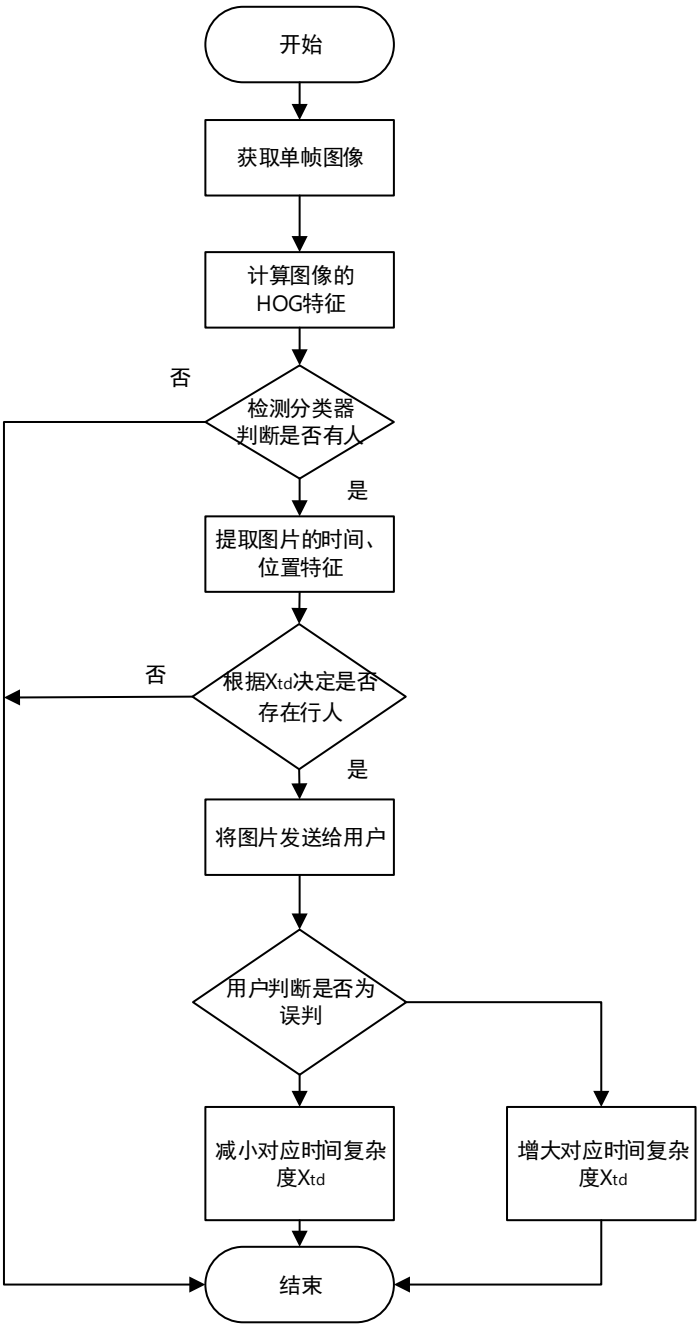

图4.3 基于可信度的自适应人员入侵检测方法示意图

4.3.4 方法分析

本小节将对本方法对误报率的影响以及可信度公式中λ的取值进行分析。

1) 误报率

本算法可以降低 OpenCV 自带行人识别算法（以下称为原算法）的误报率，此处将对此进行证明。误报率与正确率之和为 1，正确率的升高必然导致误报率的降低，本文通过证明本算法可以提升原算法的正确率，进而证明可以降低原算法的误报率。

由 4.1 节分析可知，由于光照条件以及位置的变化，原算法在不同时间段不同位

置的误报率也不相同，亦即算法的正确率不相同。将一天分为ω个时间段，将图片分为θ个区域，由于原算法在不同时间段不同位置有不同的正确率，则共有$\theta * \omega$个正确率，为了方便证明，定义$num = \theta * \omega$。定义Z_i表示第 i 个正确率，T_i为正确判断的数量，F_i为错误判断的数量，当T_i和F_i足够大时，$Z_{i=}T_i/(T_i + F_i), i \in [1, num]$。定义$C_{i=}T_i + F_i$，则判断的图片总量为$\sum_{i=1}^{num} C_i$，正确判断的图片总量为$\sum_{i=1}^{num} C_i * Z_i$，平均正确率可表示为：

$$\frac{\sum_{i=1}^{num} C_i * Z_i}{\sum_{i=1}^{num} Ci} \tag{4-8}$$

使用本方法，对算法在不同时间段不同位置的判断结果赋予可信度，则共有num个可信度，$X_{i=}T_i/(T_i + \lambda F_i)$，$i \in [1, num]$，此时算法的平均正确率为：

$$\frac{\sum_{i=1}^{num} C_i * Z_i * X_i}{\sum_{i=1}^{num} C_i * X_i} \tag{4-9}$$

因此只需证明式(4-10)大于等于式(4-9)，亦即证明：

$$\frac{\sum_{i=1}^{num} C_i \frac{T_i}{T_i + F_i} \frac{T_i}{T_i + \lambda F_i}}{\sum_{i=1}^{num} C_i \frac{T_i}{T_i + \lambda F_i}} > \frac{\sum_{i=1}^{num} C_i \frac{T_i}{T_i + F_i}}{\sum_{i=1}^{num} C_i} \tag{4-10}$$

只需证明：

$$(\sum_{i=1}^{num} C_i \frac{T_i}{T_i + F_i} \frac{T_i}{T_i + \lambda F_i})(\sum_{i=1}^{num} C_i) \geq (\sum_{i=1}^{num} C_i \frac{T_i}{T_i + \lambda F_i})(\sum_{i=1}^{num} C_i \frac{T_i}{T_i + F_i}) \tag{4-11}$$

即：

$$\sum_{j=1}^{num} \sum_{i=1}^{num} C_i C_j \frac{T_j}{T_j + F_j} \frac{T_j}{T_j + \lambda F_j} \geq \sum_{j=1}^{num} \sum_{i=1}^{num} C_i \frac{T_i}{T_i + F_i} C_j \frac{T_j}{T_j + \lambda F_j} \tag{4-12}$$

可转化为证明：

$$\begin{aligned} &\frac{1}{2}(\sum_{j=1}^{num} \sum_{i=1}^{num} C_i C_j \frac{T_j}{T_j + F_j} \frac{T_j}{T_j + \lambda F_j} + \sum_{i=1}^{num} \sum_{j=1}^{num} C_j C_i \frac{T_i}{T_i + F_i} \frac{T_i}{T_i + \lambda F_i}) \\ &\geq \frac{1}{2}(\sum_{j=1}^{num} \sum_{i=1}^{num} C_i \frac{T_i}{T_i + F_i} C_j \frac{T_j}{T_j + \lambda F_j} + \sum_{i=1}^{num} \sum_{j=1}^{num} C_j \frac{T_j}{T_j + F_j} C_j \frac{T_i}{T_i + \lambda F_i}) \end{aligned} \tag{4-13}$$

如果对于展开后的每一项，有：

$$\begin{aligned} &C_iC_j\frac{T_j}{T_j+F_j}\frac{T_j}{T_j+\lambda F_j}+C_jC_i\frac{T_i}{T_i+F_i}\frac{T_i}{T_i+\lambda F_i} \\ &\geq C_i\frac{T_i}{T_i+F_i}C_j\frac{T_j}{T_j+\lambda F_j}+C_j\frac{T_j}{T_j+F_j}C_j\frac{T_i}{T_i+\lambda F_i}) \end{aligned} \tag{4-14}$$

则原式得证，因此只需证明：

$$\frac{T_j}{T_j+F_j}\frac{T_j}{T_j+\lambda F_j}+\frac{T_i}{T_i+F_i}\frac{T_i}{T_i+\lambda F_i}\geq\frac{T_i}{T_i+F_i}\frac{T_j}{T_j+\lambda F_j}+\frac{T_j}{T_j+F_j}\frac{T_i}{T_i+\lambda F_i} \tag{4-15}$$

通分展开可得：

$$\begin{aligned} &T_i^2(T_j+F_j)(T_j+\lambda F_j)+F_j^2(T_i+F_i)(T_i+\lambda F_i) \\ &\geq T_iT_j\left[(T_i+\lambda F_i)(T_j+F_j)+(T_i+F_i)(T_j+\lambda F_j)\right] \end{aligned} \tag{4-16}$$

将左右两边式子展开，做差可得结果为：

$$\lambda(T_iF_j-T_jF_i)^2\geq 0 \tag{4-17}$$

式(4-18)大于等于 0，式(4-17)得证，所以原式得证。

通过该证明过程可证明，本方法可以提高原算法的准确率，进而证明可以降低原算法的误报率。

2) λ取值分析

λ是调节负反馈（即误报图片数量）影响程度的参数，当λ较小时负反馈对可信度的影响较弱，当λ较大时负反馈对可信度的影响较强。负反馈调节参数λ主要有以下两个重要功能：

①调节反馈对可信度的影响程度，影响程度与λ的值呈正相关。

②调节可信度稳定后趋向的终值，该终值与λ的值呈负相关。

当λ较大时，负反馈对可信度的影响较大，使得可信度快速的下降，可信度稳定后趋向的终值较小。

当λ较小时，负反馈对可信度的影响较小，使得可信度在负反馈的调解下更加平缓的变化，对可信度进行微调；可信度稳定后趋向的终值较大。

λ应取不同的值以适应不同的场景。例如：当某路视频流来自重要的监控地点，并且此地点一般情况下应该无人或者人迹罕至，对此应该降低λ的值以削弱反馈部分的作用，人为增加该路视频的可信度，使得系统不会因为决策部分的概率性筛选而错过重要的警示信息。又例如：当某路视频流来自人流较大或者经常有人路过的地带，对此应该略增加λ的值以增强反馈部分的作用，使得可信度水平整体下降，忽略

更多的信息以缓解工作人员的压力。

综上，引入负反馈调节参数λ提高了系统的灵活性和适应性，合理的调节λ使得系统能够应用于更多的场景。

4.4 本章小结

本章针对目前常用的行人检测算法，通过分析影响其误报率的因素，提出两种降低其误报率的方法，基于贝叶斯推断的人员检测结果修正方法以及基于可信度的自适应人员检测方法，并对两种方法进行了详细介绍。

第五章　系统设计与实现

本章基于前两章给出的计算架构和相关方法，设计实现了面向大规模监控系统的视频计算平台。本章将对视频计算平台的设计与实现部分做详细介绍，包括整体设计、客户端设计、中心节点设计和边缘节点设计等；并对前两章给出的方法进行验证；在本章最后，将给出视频计算平台的系统效果展示。

5.1 系统详细设计

基于第三章给出的视频计算架构，本节将对视频计算平台进行详细设计，包括整体设计、中心节点设计、边缘节点设计和客户端设计。

5.1.1　总体设计

视频计算平台主要包括三部分，边缘节点、中心节点和客户端。边缘节点部署在各个区域，负责本区域内视频流的采集和处理。它接收中心节点下发的计算任务，并将计算结果返回给中心节点。中心节点作为中央枢纽，负责管理、调度边缘节点，它接收客户端提交的视频计算任务，然后下发给边缘节点，并存储、处理边缘节点的计算结果，将计算结果反馈给用户。客户端负责与用户交互，提供给用户可视化的任务提交和结果反馈界面。视频计算平台总体结构如图 5.1 所示。

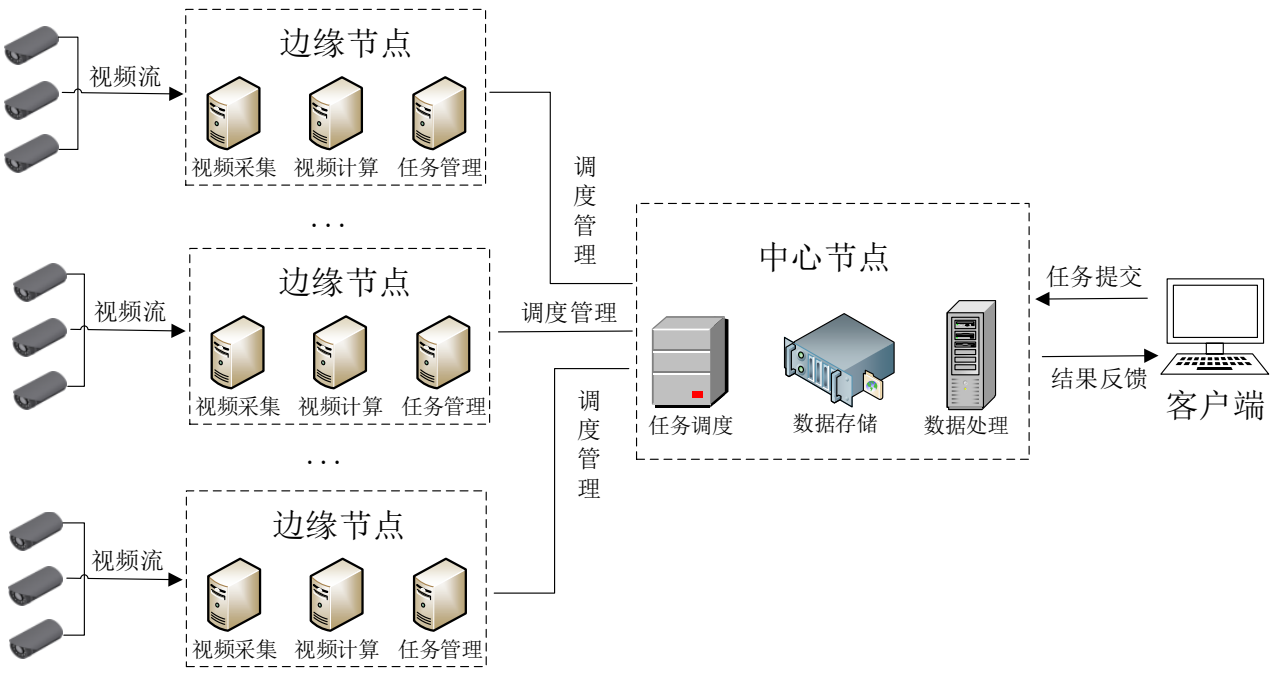

图5.1 视频计算平台结构图

5.1.2 中心节点设计

中心节点是整个视频计算平台的中央枢纽，负责存储、处理边缘节点的计算结果，为上层应用系统提供支撑。同时，中央节点承担着所有边缘节点的管理工作，其整体框架如图 5.2 所示。

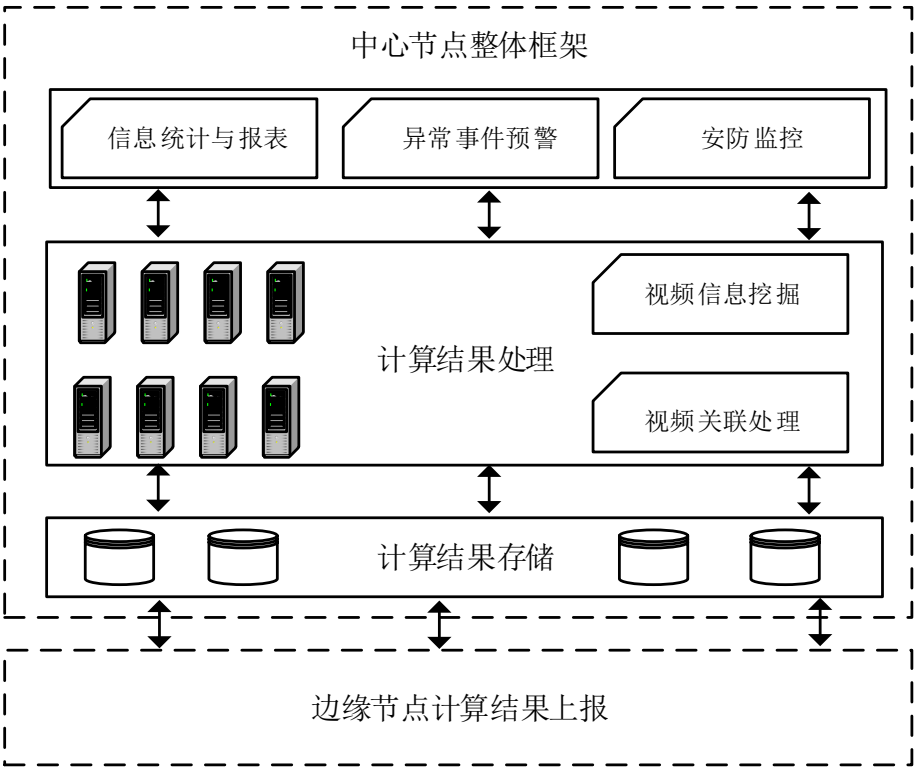

图5.2 中心节点整体框架图

中心节点主要有以下功能：计算任务调度、边缘节点与客户端管理、数据管理和信息整合。第三章针对计算任务调度策略进行了阐述，本小节将详细设计其余功能。

1) 数据管理设计

中心节点需持久化存储很多信息，包括监控地点信息、视频设备信息、用户信息、计算结果等。本文采用开源的关系型数据库 Mysql 对系统信息进行存储。

①数据表设计

平台中包含大量信息，本小节主要针对比较重要的视频设备信息表、用户信息表、节点信息表进行介绍。

设备信息表：平台需要提供给用户查看实时视频功能，所以需要存储相关视频设备的信息，包括 IP 地址、端口、用户名和密码，其表结构如表 5.1 所示。

表5.1设备信息表

字段	类型	主键	外键	描述	其他
id	int(5)	是	否	自增 ID	auto_increment
ip	char(15)	否	否	IP 地址	无
port	int(5)	否	否	端口	无
username	varchar(30)	否	否	用户名	无
password	varchar(30)	否	否	密码	无
placeID	int(4)	否	是	所属区域 ID	无

用户信息表：存储平台合法用户的相关信息，包括用户名、密码以及用户拥有的权限等。用户使用用户名密码，通过客户端登录平台，用户的权限等级代表用户是否有权限查看视频、提交计算任务等，其表结构如表 5.2 所示。

表5.2用户信息表

字段	类型	主键	外键	描述	其他
id	int(5)	是	否	自增 ID	auto_increment
username	varchar(30)	否	否	用户名	无
password	varchar(30)	否	否	密码	无
authority	int(2)	否	否	权限等级	无

节点信息表：为了保证边缘节点的合法性，给每一个边缘节点分配用户名和密码，只有使用正确用户名和密码的边缘节点才可以注册到中心节点上。同时边缘节点可以通过数据库得到中心节点的相关信息，其表结构如表 5.3 所示。

表5.3节点信息表

字段	类型	主键	外键	描述	其他
id	int(5)	是	否	自增 ID	auto_increment
username	varchar(30)	否	否	节点名	无
password	varchar(30)	否	否	密码	无
placeID	int(4)	否	是	所属区域 ID	无
centerID	int(4)	否	是	所属中心节点 ID	无

区域地点信息表：保存区域地点的信息，包括该地点名称，地址，经纬度等，其表结构如表 5.4 所示。

表5.4区域地点信息表

字段	类型	主键	外键	描述	其他
id	int(5)	是	否	自增 ID	auto_increment
name	varchar(30)	否	否	该区域名称	无
address	varchar(30)	否	否	该区域地址	无
lon	double(10)	否	否	经度	无
lat	double(10)	否	否	纬度	无

②数据库访问设计

平台中很多模块和组件需要访问数据库，比如边缘节点需要通过数据库得到中心节点的信息、客户端需要得到视频设备的信息等。这些模块和组件可能基于不同平台，运行在不同的系统环境下，因此数据库的访问设计需要具有跨平台性。

基于以上考虑，本文采用 Web Service 的方式提供数据库的访问服务。在 Web Service 的服务端将对数据库的各种操作抽象成各种接口，提供给不同模块使用。各个模块作为 Web Service 客户端使用服务端提供的各个接口即可完成对数据库的操作。

部分 Web Service 接口如表 5.5 所示。

表5.5 WebService 接口说明表

名称	含义
GetPlace	获取监控点相关信息
GetDevice	获取视频设备相关信息
GetCameras	获取视频设备的通道信息
ValidateNode	验证边缘节点信息是否正确
GetUser	获取用户相关信息
GetCoreNode	获取核心节点相关信息
AddNode	添加边缘节点
AddDevice	添加视频设备
AddPlace	添加监控点
AddCoreNode	添加核心节点

③数据库维护设计

为了便于对数据库进行维护，本文实现了基于 B/S 结构的数据库维护系统。Web 后台使用 Struts2 框架实现，前端用 JSP，并采用 JavaScript 与后台程序进行交互，消息使用 JSON 格式封装，其组成结构如图 5.3 所示。

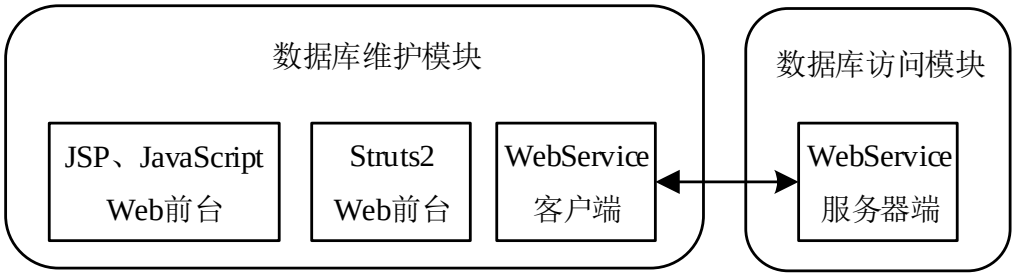

图5.3 数据库维护模块组成结构图

2) 边缘节点与客户端管理

中心节点统一管理所有的边缘节点和客户端，包括边缘节点的认证、删除、客户端的认证。只有通过认证的边缘节点才能注册到系统中，供中心节点调度使用；只有通过认证的客户端才能向中心节点提交视频计算任务。同时，当不再使用某些边缘节点时，需要将其从系统中删除。

①边缘节点与客户端认证

本文采用“单向认证”机制，即中心节点对边缘节点或者客户端进行单方面认证。中心节点为每个边缘节点和客户端分配名称和密码。边缘节点和客户端凭密码向中心节点发出认证请求，其认证过程如下（以边缘节点认证为例）：

步骤 1：边缘节点发送自己的节点名称给中心节点，请求认证服务。

步骤 2：中心节点返回随机字符串 S 给边缘节点。

步骤 3：边缘节点将自身密码的 MD5 校验值，加上字符串 S 重新计算 MD5，产生字符串 D_1，并将字符串 D_1 发送给中心节点。

步骤 4：中心节点采用同样的方法计算 D_2，比较 D_1 与 D_2。若两者相同则认证成功；否则认证失败。

②边缘节点删除

当不再使用某些边缘节点时，边缘节点向中心节点发出注销请求，由中心节点在系统中将其信息删除。其消息格式格式为：{NodeID=?，Type=2，TimeStamp=?，Sign=?}，每条数据以“{}”作为起始字符，“{}”内多个条目以“，”分隔，其具体参数说明如表 5.6 所示。

表5.6 注销请求数据包参数说明

属性	参数	说明
节点标识	NodeID	边缘节点标识号
数据包类型	Type	Type=2 代表此数据包为注销请求包
时间戳	TimeStamp	Unix 时间戳
校验值	Sign	对各个参数的 MD5 校验值，32 位的大写字符串

由于注销请求消息并不涉及隐私数据，所以不对消息内容进行加密。为了防止恶意节点仿冒边缘节点，在消息中增加校验值 *Sign* 字段，为各个参数的 MD5 校验值，其计算方式如下：

$$Sign = MD5(Node = ?\&Type = 2\&TimeStamp = ?\&Key = ?)$$

Key 为边缘节点密码，只有中心节点和该边缘节点拥有此密码。中心节点收到此消息时，采用相同的方式计算 *Sign* 值和消息中的 *Sign* 值比对，如果比对失败则直接丢弃此条消息。通过这样设计，既保证了边缘节点不被恶意冒充，同时避免了消息被篡改。

TimeStamp 代表产生计算结果的时间，为了避免重放攻击，中心节点对 *Sign* 值验证通过后，对 *TimeStamp* 进行验证，将本机时间值与 *TimeStamp* 值做差，如果结果大于阈值 *T*，则丢弃此条消息。其中阈值 *T* 需要根据具体网络情况确定，取值较大时有利于包容网络传输延迟，取值较小时有利于防重放攻击。

③心跳协议设计

各个边缘节点按固定的周期向中心节点发送心跳信息，而中心节点按固定周期检查心跳。通过心跳机制，中心节点判断各个边缘节点是否还在正常提供服务，以便及时改变处理策略以及通知用户。

设计心跳协议时，有三个关键参数需要考虑：心跳包的发送时间间隔T_S,心跳包接收检查的时间间隔T_c，以及心跳失效的超时时间 *Timeout*。通常心跳包发送和接收应该一一对应，所以T_S与T_c应该相等。而具体值，需要根据实际情况进行确定，过大则无法较快知道对方情况；过小，一方面会增加网络负担，另一方面考虑到网络延迟，容易发生误判的情况。同理，由于网络延迟的存在，*Timeout* 的值不宜设置过小。假设连续发送三个心跳包，P_1、P_2、P_3。其中P_1和P_3按时到达，P_2网络延迟较大。接收端收到P_1后，需要经过$2T_S$收到P_3，此时确认错过心跳包P_2，所以将 *Timeout* 设置为$2T_S$较为合适。

心跳包协议数据格式：{NodeID=?,Type=0,ID=?},每条数据以“{}”作为起始字符，“{}”内多个条目以“,”分隔，其具体参数说明如表 5.7 所示。

表5.7 心跳包数据参数

属性	参数	说明
节点标识	NodeID	边缘节点标识号
数据包类型	Type	0 代表此数据包为心跳包
序列号	ID	心跳包序列号

5.1.3 边缘节点设计

边缘节点是整个架构设计中的计算节点，是架构的核心。它接收中心节点分配的视频计算任务，采集前端摄像头的传来的视频数据进行分析处理，并将计算结果发送给中心节点。架构中存在多组边缘节点，考虑到带宽以及计算能力的压力，每组边缘节点负责某一区域具体的视频计算工作。每组边缘节点由多台服务器组成，负责视频采集和计算。边缘节点调度计算资源协作完成中心节点下发的计算任务，其整体结构如图 5.4 所示。

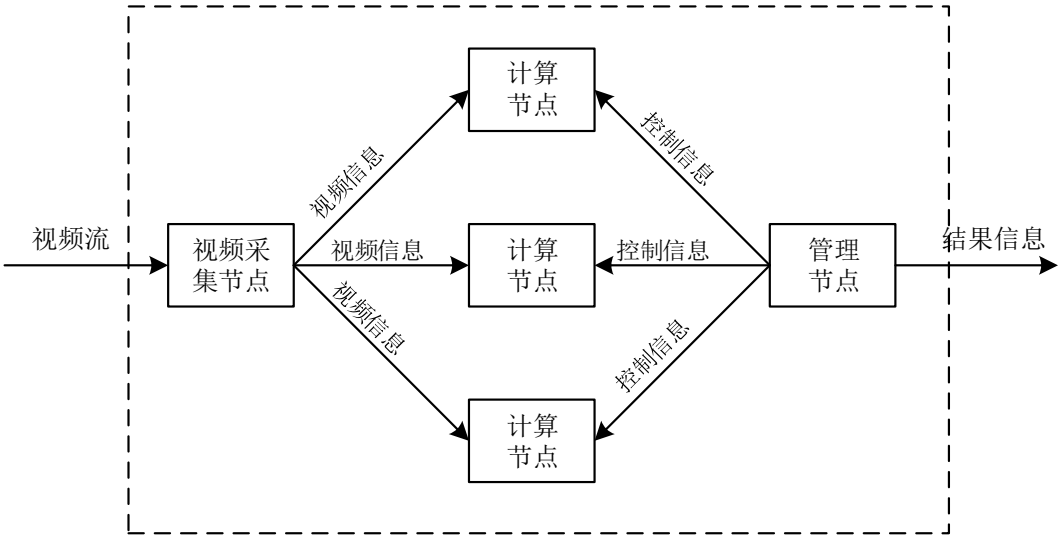

图5.4 边缘节点整体结构图

1) 视频采集节点

由于摄像头的处理能力有限，当某一摄像头被多路并发访问时，会给摄像头造成巨大压力；同时，由于不同的监控系统常常使用不同厂家的摄像头，因此摄像头的访问方式也不相同。如果计算节点直接访问前端设备，不仅会造成前端设备压力过大而且计算节点面对不同监控系统，需要使用复杂多样的访问方式，增加系统的复杂性。

本文使用流媒体服务器完成视频流的导入工作。流媒体即使用流的方式在网络上传输音频或者视频数据，流媒体服务器是流媒体应用的核心系统。流媒体服务器从前端摄像头取流，然后进行多路转发，这样，流媒体服务器的一路访问操作，便可满足多路并发访问需求；同时流媒体服务器提供统一的访问接口，降低了系统的复杂性。通过流媒体服务器分担摄像头压力，提高了系统的整体可靠性，使整体网络的负载较为均衡。

本文采用 Live555 作为流媒体服务器，Live555 是一个开源的流媒体项目，它使用 C++语言实现了对 RTSP、RTP、RTCP 以及 SIP 等标准流媒体传输协议的支持，支持接收和发送实时的媒体数据以及多种音视频编码格式的文件进行流化。

2) 管理节点

①基于最小堆的计算资源调度算法实现

由第三章介绍可知，本文采用“低负载、先调度”的方式对计算资源进行调度，即当有任务到达时，将任务优先分配给具有最低负载的计算节点执行。本小节使用基于最小堆的优先级队列对该资源调度算法进行实现，具体方式如下：

步骤 1：定义优先级队列用数组 Q 表示，元素Q_0、$Q_1 \ldots Q_N$表示。其中每一个元素均有两个属性组成，计算节点 ID 与该节点的负载值。初始时，将所有元素的负载值设为 1。

步骤 2：根据每个元素的负载值，使用堆排序算法将数组 Q 建成最小堆，则 Q 的首元素的节点 ID 即为负载最低的计算节点。

步骤 3：当有任务到达时，将任务分配给数组 Q 的首元素所代表的计算节点。

步骤 4：当收到计算节点上报的负载时，更新相应元素的负载值，此时 Q 已不满足最小堆性质，因此重新对队列 Q 执行建堆操作。

②计算结果安全上报协议

管理模块承担着与中心节点通信，上报计算结果的任务，在前边章节已经介绍了心跳协议的设计，本小节介绍计算结果上报协议。由于计算结果包含了敏感信息，例如产生计算结果的时间，计算类型等，在实现时对结果上报进行 DES 加密。明文格式设计为: {Node=?,Type=1,TaskID=?,TimeStamp=?,Sign=?},各属性达标含义如表 5.8 所示。

表5.8 结算结果上报协议

属性	参数	说明
节点表示	Node	节点的唯一标识
消息类型	Type	Type=1 代表此数据包为计算结果
任务类型	TaskID	代表不同的计算任务 ID
时间戳	TimeStamp	产生计算结果的时间，类型为 Unix 时间戳

视频计算任务的结果信息，通常只关心产生计算结果的时间和地点，例如入侵检测任务，当检测到有入侵发生时，将发生的时间和地点发送给用户；人脸匹配任务，当检测到目标人物时，将检测到的时间和地点展示给用户即可。在此明文消息中，时间戳代表了产生计算结果的时间，而中心节点可以通过节点标识查询到该边缘节点的地理位置，进一步确定产生该计算结果的地点。

对该明文使用 DES 加密算法进行加密产生密文，密钥为计算节点的密码，只有

中心节点和该边缘节点拥有此密码。中心节点收到密文后使用密钥进行解密，如果解密失败则丢弃该报文。

其中 *TimeStamp*，既作为产生计算结果的时间，又承担了防止重放攻击的作用，其原理已在 5.1.2 进行介绍。

3) 计算节点

本文使用计算机视觉库 OpenCV 对监控视频流进行处理，为每一路摄像头分配一个进程，每个进程包含三个线程，分别为视频流切片线程，视频帧处理线程以及数据上报线程。

视频流切片线程，使用 OpenCV 核心模块的 VideoCapture 函数，不断地从视频采集节点取流，然后将视频流切片保存在链表中。视频流由连续的视频帧组成，且相邻视频帧的图像内容基本相同，如果对所有视频帧进行处理，则处理压力过大。本文实现时，默认对视频流每 500 毫秒取一帧视频帧保存在链表中，如果触发丢帧策略，则取帧数量相应减少。

视频帧处理线程，使用 OpenCV 图像处理模块中的相应函数，从视频帧链表中取出视频帧进行相应处理，将计算结果发送给数据上报线程，由其发送给管理节点。同时，视频帧处理线程需要记录每一张图片的处理时间，计算平均负载，发送给数据上报线程，由其发送给管理节点。

数据上报线程，与管理节点通信，实时上报计算节点负载率以及计算结果。

5.1.4 客户端设计

客户端是用户与计算平台进行交互的重要途径，用户通过客户端向平台提交计算任务，浏览监控点视频，同时客户端还需要展示计算平台的计算结果。

本文基于 Qt 实现客户端，Qt 是一个基于 C++语言的跨平台的应用编程和 UI 编程框架，具有良好的封装机制，同时提供多种组件，允许组件式编程。通过 Qt 框架，开发者可以轻松创建图形用户界面。客户端也需要从数据库中获取信息，因此客户端需要调用 Web Service 接口，本文使用 gSoap 实现 C++平台的 Web Service 客户端程序。

客户端软件的中心节点组成结构如图 5.5 所示。

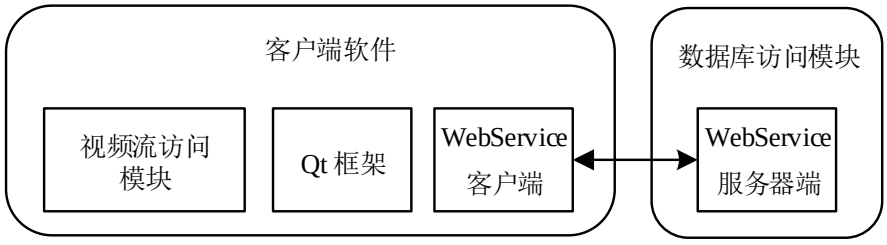

图5.5 客户端软件组成结构图

5.2 系统实验环境

面向大规模监控系统的视频计算平台作为一个复杂系统需要多种硬件的支持，本系统包括三处环境，中心节点、边缘节点和客户端。中心节点和边缘节点服务运行在 Ubutntu-Server 上，客户端软件运行在 Windows7 上，测试环境所需要用到的设备如表 5.9 所示。

表5.9 测试环境所需设备表

设备型号	配置参数	数量	操作系统	用途
浪潮 NF5270M4 2U 机架	CPU:Intel E5-2609 内存：32GB DDR4 存储：硬盘接口 SAS 容量 2T SATA 网卡：集成英特尔双端口千兆以太网卡	4	Ubuntu-Server 16.10	三台工作站用做边缘节点，负责视频计算工作。一台工作站用做中心节点，负责任务下发和信息整合
惠普 HP Pro3380 MT 台式电脑	CPU: 英特尔 第三代酷睿 i5-3470 @ 3.20 GHz 内存：4GB 海力士 DDR3 存储：希捷 ST500DM002-1BD142 (7200 转/分) 网卡：瑞昱 RTL8168E PCI-E Gigabit Ethernet NIC	1	Windwos7 专业版 32 位	运行客户端，与计算平台进行交互，负责任务提交和结果展示。
海康摄像头	支持分辨率：1280x960 帧率： 25fps 支持协议：RTSP,TCP/IP,DHCP,HTTP	若干		负责视频的采集

本平台同样需要多款软件支撑，目前使用的软件包括 Mysql5.5，OpenCV3.0，Tomcat 等。

5.3 方法测试

本节将对第三章提出的计算能力动态调整策略，第四章提出的基于贝叶斯推断的人员检测修正方法以及基于可信度的自适应人员检测方法进行测试验证。

5.3.1 计算能力动态调整策略测试

视频计算是一项计算密集型的任务，对计算能力提出了巨大挑战。随着计算任务的增加会造成计算能力不足，导致系统不稳定甚至崩溃。本文提出计算能力动态调整策略，当计算节点首次出现计算能力不足的情况时，触发视频帧丢弃策略，使有限的计算资源向重要摄像头倾斜，同时保证系统的稳定。然后通过增加计算节点数量，提升计算能力，保证所有计算任务得到充足的计算资源。

为了评估计算能力动态调整策略，本文设计使用两台服务器作为计算节点进行摄像头分析任务。首先使用一台服务器分析一定数量的监控视频，并逐步增加监控视频的路数，观察系统分析一帧图片所消耗的平均时间 *AverageTime*。设定阈值 500 毫秒，当 *AverageTime* 超过阈值之后，触发视频帧丢弃策略，观察 *AverageTime* 变化。之后，通过将另一台服务器加入系统，观察 *AverageTime* 的变化。

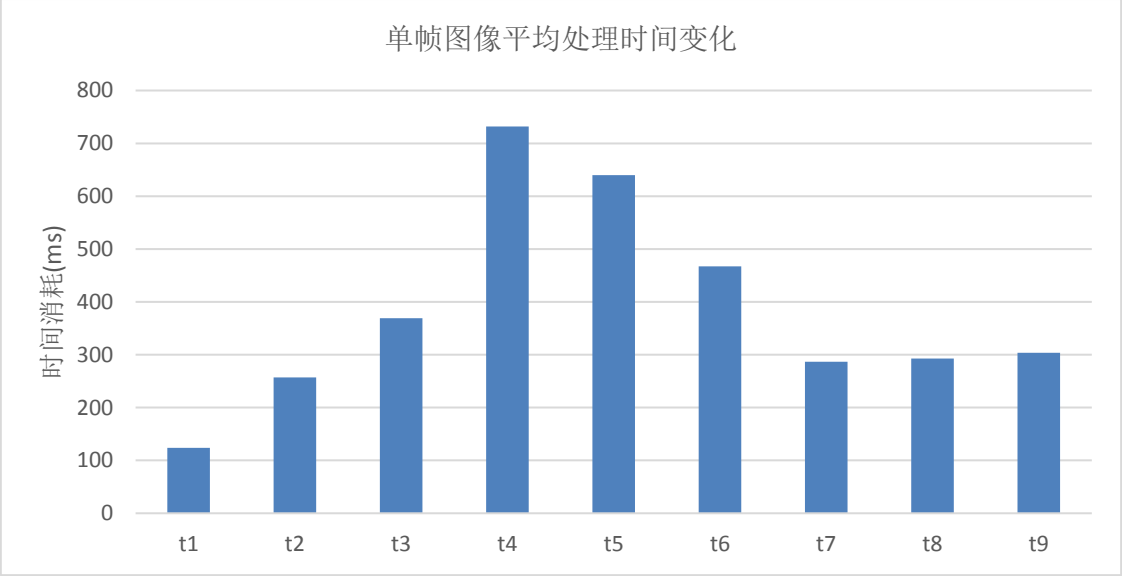

图5.6 单帧图像平均处理时间变化图

如图 5.6 所示，在计算任务刚开始的一段时间内，逐步增加计算任务，*AverageTime* 不断升高。在 t4 时刻，*AverageTime* 大于阈值 500 毫秒，触发视频帧丢弃策略，*AverageTime* 不断下降，在 t6 时刻降到阈值以下。在 t6 时刻之后，将另一台服务器加入系统，*AverageTime* 降到初始水平。由此可以表明，本文提出的计算能力动态调整策略可以使视频计算任务稳定的进行，同时可以通过增加计算节点数量来横向扩展计算能力。

5.3.2 基于贝叶斯推断的人员检测结果修正方法测试

针对当前人员检测算法在一些复杂场景下误报率较高的问题，本文提出了两种降

低误报率的方法，基于贝叶斯推断的人员检测修正方法和基于可信度的自适应人员检测方法，已在前文详述，本章将对两种方法进行测试。

本文采用某小区的停车棚作为测试环境，该停车棚背景复杂，棚内光照条件变化显著。截取一千张该停车棚的图片，设置图片中感兴趣区域大小为 800x575，时间范围为早上八点到晚上二十点。实验测试主要观察两个参数，误报率和漏检率，漏检是指有人的图片未被算法识别。本文所提算法的目标是基本不影响漏检率的情况下，降低误报率。

本小节分别采用 OpenCV 自带算法和基于贝叶斯推断的人员检测结果修正方法对各个时段采集的图片进行测试，采用贝叶斯推断的方法时，使用时间和位置两个特征属性对算法识别结果进行划分，将时间分为 13 个时间段，从 8 点到 20 点，则$t \in \{8,9,\dots 20\}$；将背景图片分为 12 个区域，则人员位置范围为$d \in \{1,2,\dots 12\}$。人工对算法的识别结果进行二次判断，计算贝叶斯修正分类器所需要的各个参数，如$P(C=0)$，$P(C=1)$等，详细步骤已在第四章介绍，使用修正分类器对算法计算结果进行修正，其对比结果如图 5.7 所示。

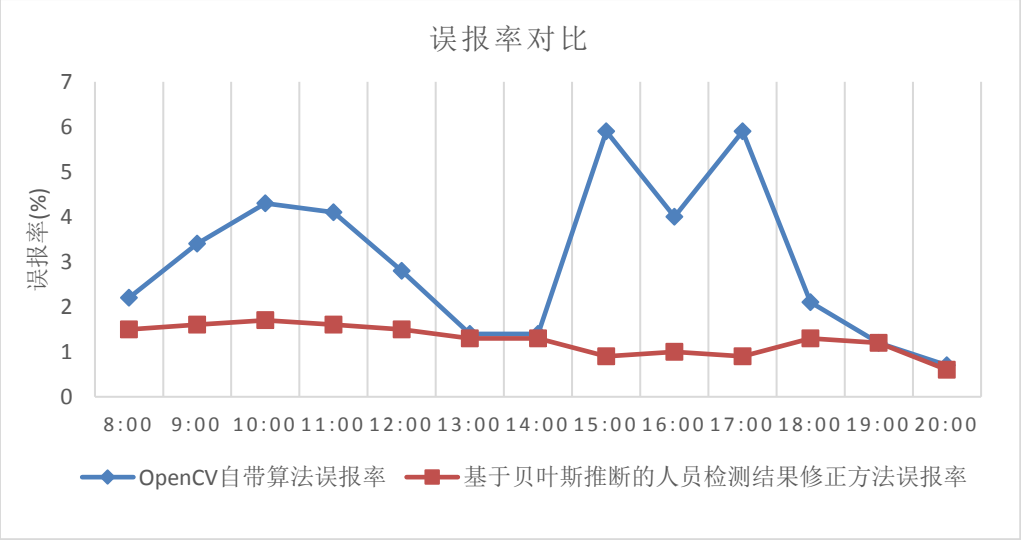

图5.7 贝叶斯结果修正算法与 OpenCV 自带算法误报率对比图

从图中可以看出随着光照条件逐渐变好，OpenCV 自带算法误报率逐渐降低，在 13 点、14 点基本达到最佳。同时可以看到，算法在 19 点，20 点算法误报率很低，分析原因为晚上摄像头开启夜间模式，采集到的图片只有黑白两色，色彩对比度较低，造成识别率降低，同时导致误报率也降低。在 15 点、16 点和 17 点误报率很高，查看算法在这三个时间段的识别结果如图 5.8 所示。

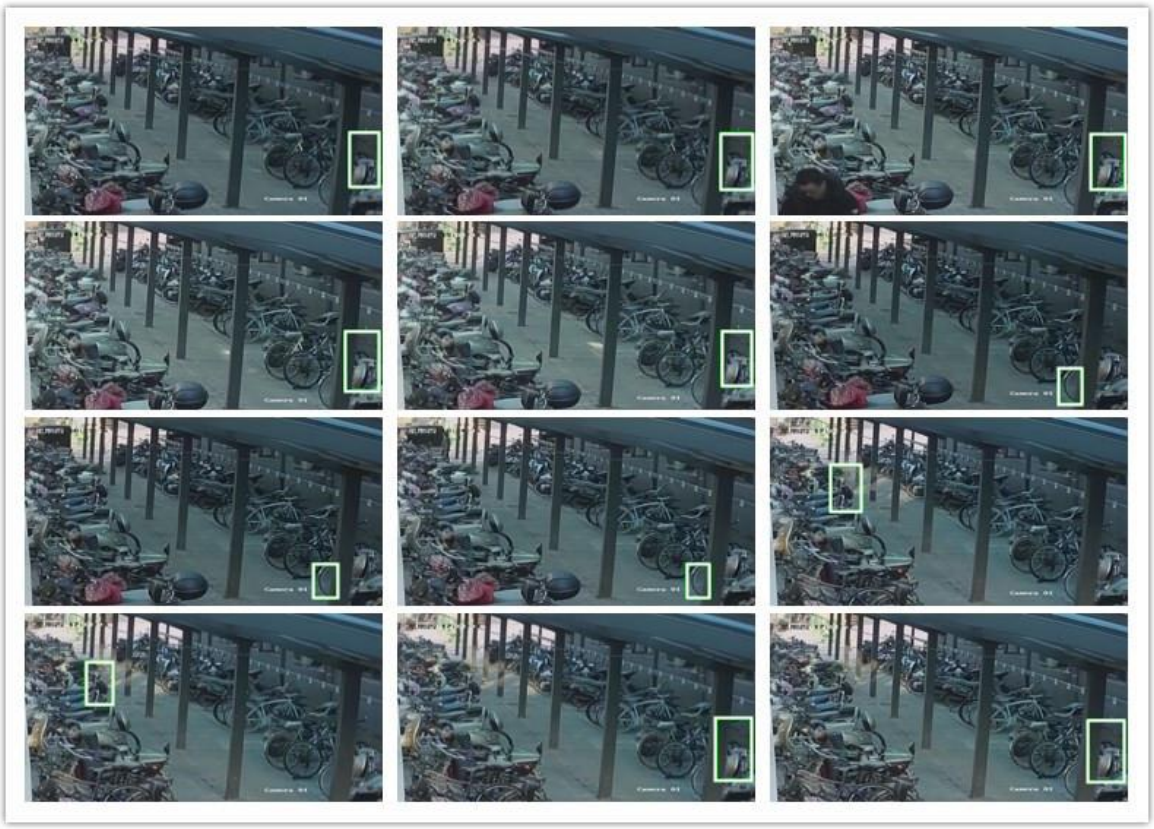

图5.8 OpenCV 算法识别结果

如图 5.8 所示，每幅图中方框区域为算法识别到的人员位置。由图可以看出，随着光线和背景的变化，在这三个时间段频繁将场景右下角识别为人，造成算法在这三个时间段误报率较高。

由于该停车棚非人员较多的场景，在一千张图片中仅有八十九张出现行人，自带算法识别出八十五张，有四张漏检。

采用基于贝叶斯推断的人员检测人员检测修正方法测试，由于原本在$t=16$、$t=16$ 和 $t=17$ 时，在 $d=8$ 和 $d=12$ 处，误报次数较多，造成$P(t=15|C=1)$，$P(t=16|C=1)$，$P(t=17|C=1)$，$P(d=8|C=1)$和$P(d=12|C=1)$值偏高，因此在此三个时间段修正效果显著。进行修正后造成八张图片漏检，与原本的漏检率相比，影响较小。修正分类器各个条件概率值如表 5.10 所示。

表5.10 修正分类器各条件概率值

概率参数	值（保留十位小数）	概率参数	值（保留十位小数）
$P(C=0)$	0.6016096579	$P(C=1)$	0.3983903421
$P(t=8\|C=0)$	0.0902675585	$P(t=8\|C=1)$	0.0555555555
$P(t=9\|C=0)$	0.0735451505	$P(t=9\|C=1)$	0.0858585858

$P(t=10\|C=0)$	0.0568227424	$P(t=10\|C=1)$	0.1085858585
$P(t=11\|C=0)$	0.0601672240	$P(t=11\|C=1)$	0.1035353535
$P(t=12\|C=0)$	0.08190635451	$P(t=12\|C=1)$	0.0707070707
$P(t=13\|C=0)$	0.10531772575	$P(t=13\|C=1)$	0.0353535353
$P(t=14\|C=0)$	0.10531772575	$P(t=14\|C=1)$	0.0353535353
$P(t=15\|C=0)$	0.03006688963	$P(t=15\|C=1)$	0.1489898989
$P(t=16\|C=0)$	0.06183946488	$P(t=16\|C=1)$	0.1010101010
$P(t=17\|C=0)$	0.03006688963	$P(t=17\|C=1)$	0.1489898989
$P(t=18\|C=0)$	0.09361204013	$P(t=18\|C=1)$	0.0530303030
$P(t=19\|C=0)$	0.10866220735	$P(t=19\|C=1)$	0.0303030303
$P(t=20\|C=0)$	0.11702341137	$P(t=20\|C=1)$	0.0176767676
$P(d=1\|C=0)$	0.11177644710	$P(d=1\|C=1)$	0.1127764471
$P(d=2\|C=0)$	0.11627906976	$P(d=2\|C=1)$	0.0029960079
$P(d=3\|C=0)$	0.02325581395	$P(d=3\|C=1)$	0.0079840319
$P(d=4\|C=0)$	0.02325581395	$P(d=4\|C=1)$	0.0029960079
$P(d=5\|C=0)$	0.02325581395	$P(d=5\|C=1)$	0.0259481037
$P(d=6\|C=0)$	0.04651162790	$P(d=6\|C=1)$	0.0998003992
$P(d=7\|C=0)$	0.06976744186	$P(d=7\|C=1)$	0.0029960079
$P(d=8\|C=0)$	0.02325581395	$P(d=8\|C=1)$	0.2994011976
$P(d=9\|C=0)$	0.04651162790	$P(d=9\|C=1)$	0.0818363273
$P(d=10\|C=0)$	0.02325581395	$P(d=10\|C=1)$	0.0029960079
$P(d=11\|C=0)$	0.02325581395	$P(d=11\|C=1)$	0.0029960079
$P(d=12\|C=0)$	0.04651162790	$P(d=12\|C=1)$	0.3632734530

由于修正分类器各条件概率参数基于历史判断结果，因此该方法适用于环境条件规律性变化的场景，当变化规律（例如光照条件的变化规律）发生较大变化时，需要重新采集数据训练修正分类器。

5.3.3 基于可信度的自适应人员检测方法测试

根据第四章的介绍，从时间和空间维度对行人识别分类器的可信度进行划分，$t \in \{8,9,\dots 20\}$，$d \in \{1,2,\dots 12\}$，可信度的计算公式为：$X_{td} = \frac{T_{td}}{T_{td}+\lambda F_{td}}$。

分别取$\lambda = 1, \lambda = 0.8, \lambda = 0.6$测试，其对比结果如图 5.9 所示。

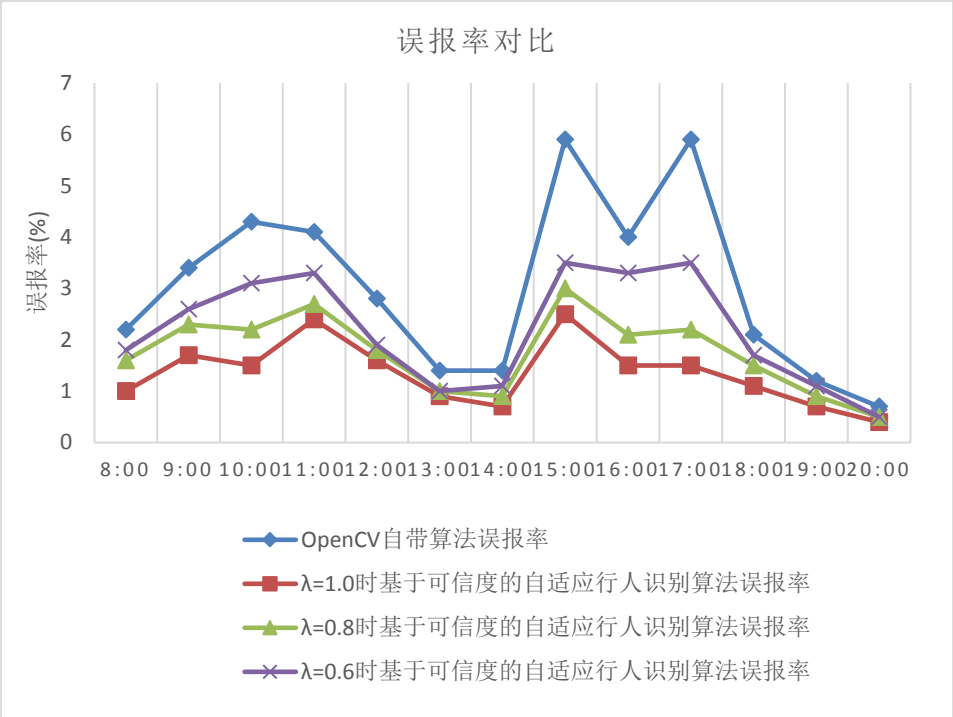

图5.9 基于可信度算法和 OpenCV 自带算法误报率对比图

当$\lambda = 1$时漏检 17 张，$\lambda = 0.8$时漏检 12 张，当$\lambda = 0.6$漏检 8 张。

由此可见，随着λ的升高，算法的误报率逐渐降低，但是漏检率逐渐升高。因此λ的取值需根据场景判定。误报较少，行人出现频繁的场景，λ宜取较低值；误报较多，行人出现较少的场景，λ宜取较高值。

5.4 系统功能验证

本系统统一接入各区域摄像头，并对各个监控地点视频进行实时分析，及时高效地对异常事件进行分析、预警，并提供给用户查看实时视频、录像回放、提交计算任务、查看计算结果以及数据管理等功能。为了验证本文所提出的架构设计及设计方案的实际效果，本节将对系统功能进行测试，验证前述架构、方法以及设计方案的合理性与正确性。

为方便数据管理，本文开发了基于 B/S 结构的数据库维护系统，提供给用户方便、简洁的数据库维护系统。如图 5.10 所示为视频计算凭条数据管理系统管理员界面，从管理员界面可以看出，数据管理主要包括管理监控地点信息、视频设备信息、用户信息等。当有新区域加入到系统中时，管理员通过数据库维护系统，添加监控地点，完善该地点的信息，包括该地点地理位置，所包含视频设备等；当有新用户加入到系统中时，系统为新用户分配用户名、密码以及相关权限。

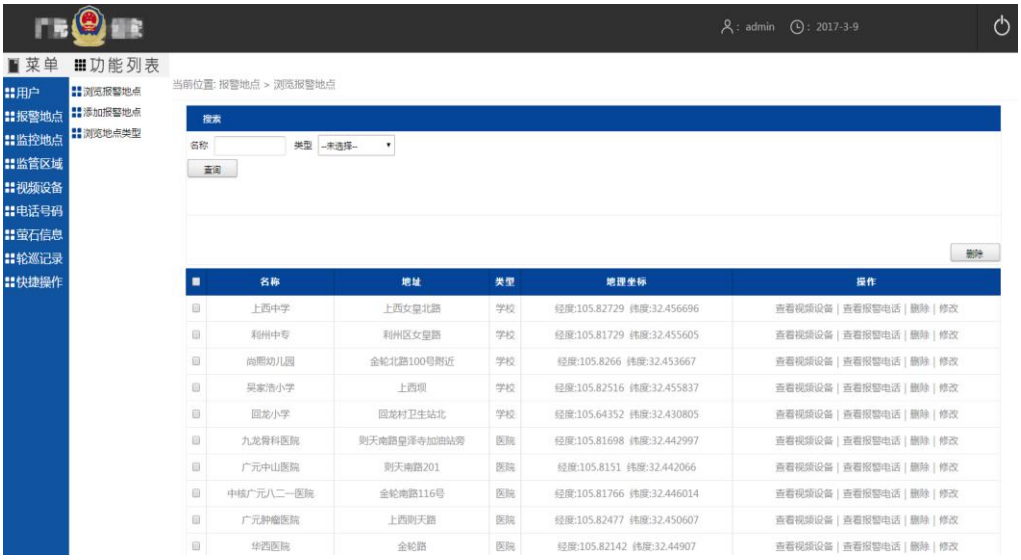

图5.10 数据管理界面

客户端是用户与平台交互的桥梁，本文开发了基于 QT 的客户端界面，主要包括登陆界面、主界面、任务提交界面等。如图 5.11 所示为客户端登录界面，用户使用平台分配的用户名和密码进行登录，系统后台对用户的输入进行验证，验证通过后才可进入到客户端主界面。

图5.11 客户端登录界面

用户登录成功后即可进入到客户端主界面，客户端提供给用户多项功能，包括实时视频浏览、历史视频查看、抓图浏览、信息配置等多种功能，如图 5.12 所示。同时可以通过客户端针对特定区域提交视频计算任务，例如人员入侵检测、行人徘徊检测等，任务提交窗口如图 5.13 所示。

图5.12 客户端主界面

图5.13 任务提交界面

系统的计算结果最终通过客户端展示给用户，以上述人员入侵任务为例，当边缘节点检测到人员入侵事件时，将相关信息发送给中心节点，由中心节点发送给客户端，客户端以右下角弹窗的方式提醒用户，如图 5.14 所示

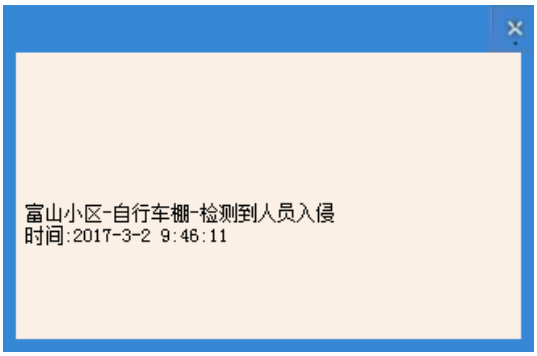

图5.14 计算结果提示窗口

用户收到系统提示后，可以查看该地点的实时视频以及历史视频，并做出相应处置，如图 5.15 所示。

图5.15 视频查看窗口

5.5 本章小结

本章主要描述了系统的设计与实现。首先针对第三章提出的视频计算架构进行了详细设计，包括整体设计、客户端设计、中心节点以及边缘节点设计。然后介绍了系统的实验环境，并对前两章提出的计算能力动态调整策略、基于贝叶斯推断的人员检测结果修正方法以及基于可信度的自适应人员检测方法进行了测试。最后给出系统的效果展示，验证了系统的可用性。

第六章　总结与展望

在“平安城市”的建设过程中，监控系统发挥着无法取代的作用。然而目前的监控系统以人工监控为主，存在消耗大量人力、易使人产生疲劳、可扩展性差等问题。尤其随着监控系统规模的不断扩大，以人工监控为核心的监控系统越来越无法满足需求。智能监控系统的概念应用而生，它使用计算机强大的计算能力代替人工，利用视频计算技术对监控视频进行处理与分析，做到智能监控、自动预警。

本文针对大规模监控系统带来的数据传输压力以及计算压力问题，提出边缘化、分布式的视频计算架构，围绕数据部署计算资源，同时部署中心节点对所有计算资源进行统一管理调度，计算节点只需将计算后的信息片段发送到中心节点，从而减低通信带宽的消耗，减小数据传输压力。同时提出一种计算能力动态调整策略，以解决监控系统规模扩大时带来的计算压力，保证充足的计算资源以及系统稳定。图像分析算法存在易受环境因素影响的问题，本文针对当前常用的行人检测分析算法，提出两种降低其误报率的方法：基于可信度的自适应行人检测方法和基于贝叶斯推断的行人检测结果修正方法。在本文最后，对视频计算架构进行了实现，同时对系统方法进行了测试，验证了系统以及方法的可用性。

本文虽然解决了大规模监控系统带来的数据传输压力以及计算能力问题，但是仍然有很多领域需要研究，包括安防数据挖掘、监控视频关联计算等问题。本文针对行人检测方法提出了两种降低误报率的方法，然而行人检测算法只是众多图像分析算法里的一小部分，如何保证所有图像算法高效运行，不受环境因素影响，还有待进一步研究。如果上述问题都得到解决，相信监控系统将得到长足的发展。

参考文献

[1] 马昌义. 上海平安城市建设与应用趋势[J].中国公共安全.2013.(Z1): 74-77.

[2] 马晓东. 平安黑龙江 和谐黑土地——黑龙江城市报警与监控系统建设与规划[J]. 中国公共安全 (综合版), 2011, 9: 008.

[3] 肖珂, 高冠东. 基于智能监控技术的平安城市建设系统研究[J]. 农业网络信息, 2013 (12): 20-22.

[4] 建新.智能视频监控技术的演变对比及主要优势[J].金卡工程,2006.4:69-71.

[5] Held C, Krumm J, Markel P, et al. Intelligent video surveillance[J]. Computer, 2012, 45(3): 83-84.

[6] Diamantopoulos G, Span M. Event detection for intelligent car park video surveillance[J]. Real-Time Imaging,2005,11(3):233-243.

[7] Wang W, Zhang D, Zhang Y, et al. Robust spatial matching for object retrieval and its parallel implementation on GPU[J]. Multimedia, IEEE Transactions on, 2011, 13(6): 1308-1318.

[8] Ahrendt P, Gregersen T, Karstoft H. Development of a real-time computer vision system for tracking loose-housed pigs[J]. Computers and Electronics in Agriculture,2011, 76(2): 169-174.

[9] Poudel P, Shirvaikar M. Optimization of computer vision algorithms for real time platforms[C]//System Theory (SSST), 2010 42nd Southeastern Symposium on. IEEE, 2010: 51-55.

[10] Chen Q, Qiu Q, Li H, et al. A neuromorphic architecture for anomaly detection in autonomous large-area traffic monitoring[C]//Computer-Aided Design(ICCAD), 2013 IEEE/ACM International Conference on. IEEE, 2013:202-205.

[11] Xu R, Tsai C Y, Kender J R. An adaptive anchor frame detection algorithm based on background detection for news video analysis[C]//Audio, Language and Image Processing (ICALIP), 2016 International Conference on. IEEE, 2016: 743-748.

[12] Micheloni C, Foresti G L, Snidaro L. A network of co-operative cameras for visual surveillance[J]. IEE Proceedings-Vision, Image and Signal Processing, 2005, 152(2): 205-212.

[13] Snidaro L, Niu R, Varshney P K, et al. Automatic camera selection and fusion for outdoor surveillance under changing weather conditions[C]//Advanced Video and Signal Based Surveillance, 2003. Proceedings. IEEE Conference on. IEEE, 2003: 364-

369.

[14] Christmas J. Theoretical motion functions for video analysis, with a passive navigation example[C]//Neural Networks (IJCNN), 2016 International Joint Conference on. IEEE, 2016: 4001-4008.

[15] Wang G, Tao L, Di H, et al. A scalable distributed architecture for intelligent vision system[J]. IEEE transactions on Industrial Informatics, 2012, 8(1): 91-99.

[16] Hong K, Voelz M, Govindaraju V, et al. A distributed framework for spatio-temporal analysis on large-scale camera networks[C]//Distributed Computing Systems Workshops (ICDCSW), 2013 IEEE 33rd International Conference on. IEEE, 2013: 309-314.

[17] Saini M, Wang X, Atrey P K, et al. Adaptive workload equalization in multi-camera surveillance systems[J]. IEEE Transactions on Multimedia, 2012, 14(3): 555-562.

[18] Geng Chen-yao. Distributed Video Processing Platform Based on Map Reduce[J]. Computer engineering，2012: 280-283.

[19] 李招昕. 基于流式计算的大规模监控视频分析关键技术研究[D]. 上海大学, 2015.

[20] 方权亮, 余谅. 基于云计算的智能高清视频监控系统研究[J]. 微型机与应用, 2013, 32(3): 90-92.

[21] Sanchez N, Menéndez J M. Video analysis architecture for enhancing pedestrian and driver safety in public environments[C]//Image Analysis for Multimedia Interactive Services, 2009. WIAMIS'09. 10th Workshop on. IEEE, 2009: 288-291.

[22] 曹海宾. 海量视频的分布式协作处理与检索研究[D]. 中国科学技术大学, 2013.

[23] Bansal R, Sehgal P, Bhasin V, et al. Multi-agent system for intelligent watermarking of fingerprint images[C]//Fuzzy Systems (FUZZ), 2013 IEEE International Conference on. IEEE, 2013: 1-8.

[24] 李立仁，李绍军，刘忠岭.智能视频监控技术综述[J].中国安防，2009，（4）：89-95.

[25] Lipton A J, Fujiyoshi H, Patil R S. Moving target classification and tracking from real-time video[C]//Applications of Computer Vision, 1998. WACV'98. Proceedings., Fourth IEEE Workshop on. IEEE, 1998: 8-14.

[26] 李刚, 邱尚斌, 林凌, 等. 基于背景差法和帧间差法的运动目标检测方法[J]. 仪器仪表学报, 2006, 27(8): 961-964.

[27] Horn B K P, Schunck B G. Determining optical flow[J]. Artificial intelligence, 1981, 17(1-3): 185-203.

[28] McKenna S J, Jabri S, Duric Z, et al. Tracking groups of people[J]. Computer Vision and Image Understanding, 2000, 80(1): 42-56.

[29] Smeulders A W M, Chu D M, Cucchiara R, et al. Visual tracking: An experimental survey[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2014, 36(7): 1442-1468.

[30] Isard M, Blake A. Contour tracking by stochastic propagation of conditional density[C]//European conference on computer vision. Springer Berlin Heidelberg, 1996: 343-356.

[31] Yilmaz A, Li X, Shah M. Object contour tracking using level sets[C]//Asian Conference on Computer Vision. 2004.

[32] Seitz S M, Kutulakos K N. Proceedings of the IEEE International Conference on Computer Vision[C]//IEEE. 1998.

[33] Babenko B, Yang M H, Belongie S. Visual tracking with online multiple instance learning[C]//Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on. IEEE, 2009: 983-990.

[34] Kalal Z, Matas J, Mikolajczyk K. Pn learning: Bootstrapping binary classifiers by structural constraints[C]//Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on. IEEE, 2010: 49-56.

[35] Santner J, Leistner C, Saffari A, et al. PROST: Parallel robust online simple tracking[C]//Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on. IEEE, 2010: 723-730.

[36] Csurka G, Dance C, Fan L, et al. Visual categorization with bags of keypoints[C]//Workshop on statistical learning in computer vision, ECCV. 2004, 1(1-22): 1-2.

[37] Su S Z, Li S Z, Chen S Y, et al. A survey on pedestrian detection[J]. Dianzi Xuebao(Acta Electronica Sinica), 2012, 40(4): 814-820.

[38] Dalal N, Triggs B. Histograms of oriented gradients for human detection[C]//Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on. IEEE, 2005, 1: 886-893.

[39] 于仕琪，刘瑞祯译. 学习 OpenCV[M]. 北京：清华大学出版社，2009.10，1-16,351~355.

[40] Li M and Baker M. The Grid CoreTechnologies[M] . Chichester, England: John Wiley & Sons,2005.

[41] Dong F, Akl S G. Scheduling algorithms for grid computing: State of the art and open

problems[R]. Technical report, 2006.

致谢

随着论文的完成，二十年的求学生涯也即将结束。在西安电子科技大学度过的这七年里，在科研和生活上都学到了很多。西安电子科技大学为学生提供了优良的学习环境和科研条件，学校里浓厚的科研氛围对我产生了深深的影响。通过在西安电子科技大学的学习，我的科研能力和技术水平得到了飞快的提升，知识体系得到了极大丰富，为我步入社会、走向工作岗位奠定了坚实的基础。

本篇论文是在导师沈玉龙教授的悉心指导下完成的，沈老师具有严谨的治学态度，渊博的知识，在科研方面为我们树立了榜样，指明了方向。在本篇论文完成的过程中，沈老师耐心指导，帮助我理清思路，不厌其烦地和我讨论，指出问题所在，为论文的最终完成指明了方向。沈老师不仅在学术上认真指导、耐心交流，在生活方面也对学生细心呵护，关爱有加。非常感谢沈老师在我研究生生涯对我的帮助。

感谢研究生阶段一起学习生活的胡俊、祝幸辉、徐荣茂、司旭、刘茜和张英等实验室成员，在科研方面当我遇到问题时，你们帮我翻阅文献、排忧解难，在生活上你们对我关心、照顾。在这里对你们的陪伴表示衷心的感谢。

此外还要感谢实验室的陈博闻、睢悦、徐真真和林旭等师弟和师妹们，你们的勤奋工作为我分担了很多科研任务和压力，和你们的讨论与交流让我学习到了很多东西，课外的各项娱乐活动，使我在西电的求学生活充满了乐趣。

最后感谢我的家人在我遇到困难时对我的鼓励和支持，感谢你们这么多年来对我的教育和培养，在生活上对我的支持。你们是我不断奋斗、勇往直前的动力

作者简介

1. 基本情况

田朝会，男，河北石家庄人，1991 年 10 月出生，西安电子科技大学计算机学院计算机系统结构专业 2014 级硕士研究生。

2. 教育背景

2010.08～2014.07 西安电子科技大学，本科，专业：计算机科学与技术；
2014.08～2017.07 西安电子科技大学，硕士研究生，专业：计算机系统结构。

3. 攻读硕士学位期间的研究成果

3.1 申请（授权）专利

[1] 沈玉龙,林旭,田朝会等. 专利名称:一种基于可信度的自适应人员入侵检测方法 中国, 201710145116.8[P]. 2017.03.13.

[2] 沈玉龙,徐真真,田朝会等专利名称:一种基于贝叶斯推断的人员入侵检测结果修正方法 中国, 201710145115.3[P]. 2017.03.13.

3.2 参与科研项目及获奖

[1] 陕西省科技统筹创新工程计划项目，物联网综合接入与服务提供平台及示范应用，2016.01-2018.12，在研，参与软件开发。

3.3 申请软件著作权

[1] 软著名称：平安城市智能协同支撑平台 PC 客户端软件。
受理号：2017R11S059157。

[2] 软著名称：平安城市地理信息协同系统软件。
受理号：2017R11S059158。

[3] 软著名称：平安城市智能协同支撑平台移动客户端软件。
受理号：2017R11S059156。