文章编号：2095-2163(2020)03-0196-06 中图分类号：TP277 文献标志码：A

基于边缘计算的视频监控报警系统研究

殳雪城，童晓斌，刘天宏，刘 媛
（上海工程技术大学 机械与汽车工程学院，上海 201620）

摘 要：随着人们对安防意识的不断增强，视频监控报警变得越来越重要，但传统的设计方法暴露出时延高的缺点。本文结合边缘计算的热点，将视频监控与边缘计算技术结合起来，设计了一种基于边缘计算的视频监控报警系统。系统包括边缘计算网关，以 EAI610 为核心，采用帧间差分法处理视频图像数据，运用 Docker 虚拟化容器技术，分布式部署边缘计算框架。通过实验对比的方法，选定帧间差分法的关键参数，二值化阈值为 10，在此基础上，测试分析了整个系统的最大报警平均时延为 187.7 ms，最小报警平均时延为 135.5 ms，实验结果表明，基于边缘计算的视频监控报警系统符合要求，且大大地降低了时延，为视频监控报警系统提供了新的思路，具有很高的应用价值。

关键词：边缘计算；网关；视频监控系统；帧间差分法

Research on video monitoring and alarm system based on edge computing

SHU Xuecheng, TONG Xiaobin, LIU Tianhong, LIU Yuan
(School of Mechanical and Automotive Engineering, Shanghai University of Engineering Science, Shanghai 201620, China)

【Abstract】 With the increasing awareness of security, video monitor and alarm become more and more important, but the traditional design method exposes the shortcomings of high delay. In this paper, the video monitor and alarm system based on edge computing is designed, which combines video monitor and alarm with edge computing. The system includes an edge computing gateway, which takes EAI610 as the core, uses temporal difference to process video image data, and uses Docker virtualization container technology to deploy a distributed edge computing framework. By comparing the experimental methods, the key parameters of temporal difference method are selected and the threshold value of binarization is 10. On this basis, the maximum average alarm delay of the whole system is 187.7 ms and the minimum average alarm delay is 135.5 ms. The experimental results show that the video monitor and alarm system based on edge computing meets the requirements. It reduces the time delay, which provides a new idea for video monitor and alarm system, and has high application value.

【Key words】 edge computing; gateway; video monitor and alarm; temporal difference

0 引 言

随着生活水平的不断提高，人们对安全的防范意识在不断增强。作为安防系统的一个重要组成部分，视频监控报警系统在安防系统中扮演的角色越来越重要，同时也吸引了越来越多的关注[1-2]。视频监控系统最初的应用场景包括交通安全、超市监控和大型会场的安保等公共场所，近年来，则从公共场所逐渐转向了家庭安防，从而推动了视频监控系统的快速发展。传统的模拟监控系统受限于距离的限制。远程视频监控系统采用数字监控系统，传输距离不收限制。且将采集到的数据上传至云平台，在云平台上处理，控制，暴露出时延比较高的缺点。

边缘计算[3-4]是在靠近数据源头的网络边缘侧，融合网络、计算、存储及应用等核心能力的开放平台，近提供边缘智能数据处理服务，以满足网络敏捷连接、实时业务、数据优化等应用需求。可以缓解云平台的数据处理的负担，提高了数据处理的效率。同时由于传感器与边缘端更贴近，极大地降低了时延。

本文提出了一种基于边缘计算的视频监控报警系统，系统主要包括感知传感器的节点的部署，边缘计算网关和软件平台。研究的目的为搭建视频监控报警平台，部署感知传感器，添加边缘计算的规则，实现视频监控异常报警功能，降低报警时延。

1 系统组成

本文采用 EdgeX Foundry 作为视频监控报警系统的核心框架，系统组成如图 1 所示，包含罗技摄像头 270、EdgeX Foundry 边缘计算平台和物联网应用平台。具体流程如图 2 所示。

作者简介：殳雪城(1994-)，男，硕士研究生，主要研究方向：边缘计算、物联网应用。
通讯作者：童晓斌 Email：tongxiaobin1994@163.com

收稿日期：2019-10-11

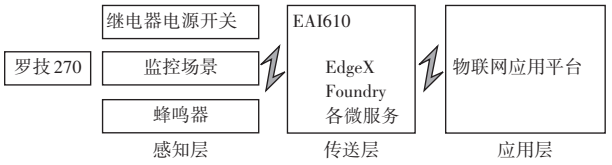


图 1　系统组成

Fig. 1　System composition

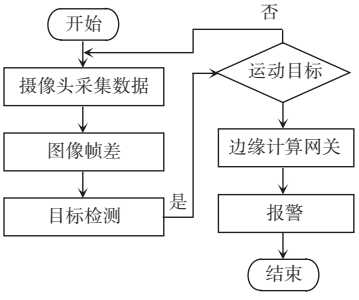


图 2　流程图

Fig. 2　Flow chart

首先通过罗技摄像头对监控区域进行数据图像的采集，然后将获得数据图像通过帧差法进行处理判定，将判定后的结果通过 MQTT 通讯协议发送给 EdgeX Foundry 边缘计算平台，边缘计算网关根据制定的规则逻辑，通过规则引擎和通知模块完成事件的通知和报警。规则引擎模块根据制定的规则，对报警传感器进行联动，执行报警操作，通知模块在满足规则逻辑的情况下，通过邮件发送的方式，向指定的用户发送报警通知邮件，最终实现视频监控报警功能。

2　移动目标检测算法

移动目标检测的目的是为了有效地提取到视频中变化的部分，这是智能图像处理的基本部分，是人工智能和机器视觉发展的基础。在视频监控报警系统中进行移动目标检测是安防系统的一个重要特征。移动目标检测算法主要有光流法、背景差分法和帧差法。本文通过对比分析，选择帧差法作为本文所采用的移动目标检测算法。

帧差法[5]又称图像序列差分法，是将相邻的 2 帧图像进行差分，并将所得到的的结果与所设定的阈值函数进行比较，如果差分的结果大于阀值函数，则设定二值差分图像的值为 1，与之相反，则设定为 0，其实现原理可写为如下数学公式：

$$f_d(x,y,t_1,t_2)=f(x)=\begin{cases}1, & \text{if } |f(x,y,t_1)-f(x,y,t_2)|>T;\\ 0, & \text{其他.}\end{cases} \tag{1}$$

其中，T 为阈值，当差分结果大于 T 时，则认定有物体闯入，当差分结果小于 T 时，则认为不存在运动目标，即无物体闯入。相比于光流检测法和背景差分法，帧间差分法的优点在于，对光线变化的敏感性较低，不易受到气候条件变化的影响。

3　边缘计算平台

3.1　EdgeX Foundry 框架

边缘计算要具备时效性、安全性、计算能力，并且部署在边缘侧，对工程大小、内存占用、CPU 消耗等都有严格的要求，目前全球各大公司都加入了边缘计算平台系统的开发中，例如：Apache Edgent、OpenStack、EdgeX Foundry、微软的 Azure IoT Edge、Google Cloud IoT 和亚马逊的 AWS Greengrass 等[6]。

本文选取的边缘计算平台 EdgeX Foundry[7]作为边缘计算网关，EdgeX Foundry 是由 Linux 基金会运营的厂商中立的开放源码项目，框架由 2017 年开始被人们所熟知，旨在为“物联网边缘计算”创建公共开放的框架。EdgeX Foundry 的边缘计算系统框架如图 3 所示。

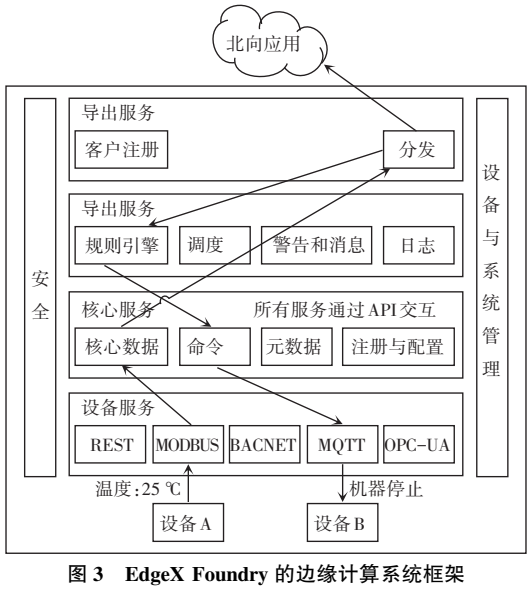


图 3　EdgeX Foundry 的边缘计算系统框架

Fig. 3　Edge computing system framework of Edgex Foundry

由图 3 可看出 EdgeX Foundry 框架从南到北依次为：设备服务层、核心服务层、支持服务层、导出服务层。框架分为南北两侧。南侧包括：在物理领域内的所有物联网对象，以及与这些设备、传感器、执行器和其他物联网对象直接通信并从中收集数据的网络边缘，统称为“南侧”。北侧包括：将数据收集、存储、聚合、分析并转换为信息的云（或企业系统），以及与云通信的网络部分，称为网络的“北侧”。

4 个服务层从南侧到北侧分别为：设备服务层、核心服务层、支持服务层、导出服务层。对此可做阐释分述如下。

（1）设备服务层（DS）：设备服务层负责与南向设备交互。

(2)核心服务层(CS):核心服务层分隔了边缘的北侧和南侧层。核心服务包括以下组件:核心数据,命令,元数据,注册和配置。

(3)支持服务层(SS):支持服务层包含广泛的微服务,该层微服务主要提供边缘分析服务和智能分析服务。

(4)导出服务层(ES):北向应用可以在网关注册,并获取其想获得的南向设备的数据;设置数据发送的方向;设置数据传输的格式。

在此基础上,研究得到 2 个增强的基础系统服务分别为:系统管理,安全基础设施。这里给出概述如下。

(1)设备与系统管理:提供 EdgeX Foundry 微服务的安装、升级、启动、停止和监视, 以及 BIOS 固件、操作系统和其他与网关相关的软件。

(2)安全:EdgeX Foundry 内外的安全元件保护由 EdgeX Foundry 管理的设备、传感器和其他物联网对象的数据和命令。

3.2 规则引擎

3.2.1 规则引擎作为导出微服务客户端

规则引擎微服务提供了一种边缘事件触发机制。规则引擎服务接受传感器传入的数据,并触发设备,执行联动。因此,规则引擎在网络边缘处或附近提供“智能”,以加快响应时间。

该实现在其核心使用一个 Drools 规则引擎。Drools 是一个开源规则引擎。这种微服务能够被第三方提供的许多其他边缘分析功能所取代或增强。

规则引擎是一个自动注册为导出微服务的客户端。当规则引擎微服务启动时,将会自动调用导出客户端注册微服务,将自己注册为从核心数据中输出的所有设备和传感器读数的客户端。作为导出微服务的客户端,通过导出微服务分发数据,规则引擎接收所有事件和读数。基于接受到的数据,执行制定的规则,并且规则引擎通过核心命令微服务,触发对设备的任何驱动,实现设备间的联动,其流程图如 4 所示。

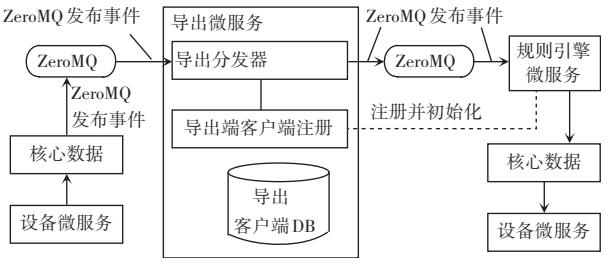


图 4 规则引擎作为导出微服务客户端

Fig. 4 Rule engine as export microservice client

3.2.2 规则引擎直接连接核心数据

在对时间敏感的生产环境中,大量数据是由连接的传感器生成的,将规则引擎微服务直接连接到核心数据,从核心数据微服务接受数据。其流程图如图 5 所示。

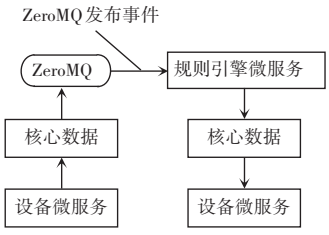


图 5 规则引擎直连核心数据

Fig. 5 Rule engine directly connected to core data

3.2.3 规则引擎客户端高级交互

规则引擎微服务附带了一个 RESTful 服务,可以添加和删除新的规则。RESTfulAPI 允许在 JSON 中定义新规则,通过 REST POST 动态添加到规则引擎中。微服务将提供的 JSON 数据转换为 Drools 规则文件(.drl 文件)。每个规则必须与一个唯一的名称相关联,该名称用于标识规则和保存规则的 Drool 文件。还可以按名称请求删除规则。其交互图如图 6 所示。

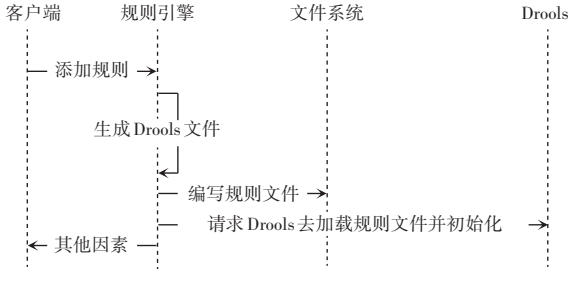


图 6 规则引擎客户端高级交互图

Fig. 6 Advanced interaction diagram of rule engine client

3.3 通知引擎

3.3.1 通知引擎模块

通知具有信息性,而警报通常具有更重要、关键或紧急的性质,可能需要立即采取行动。其系统图如图 7 所示。

图 7 显示了警报和通知的高级体系结构。在左侧,API 是为其他微服务、盒内应用提供的,这些 API 可以是 REST、AMQP、MQTT 或任何标准应用程序协议。在右侧,通知接收器可以是云端或个人服务器或应用程序系统。通过调用订阅 RESTful 接口,订阅特定类型的通知,当事件发生时,接收者通过定义的接收通道获取适当的通知。接收通道包括短消息、电子邮件、REST 回调、AMQP、MQTT 等。

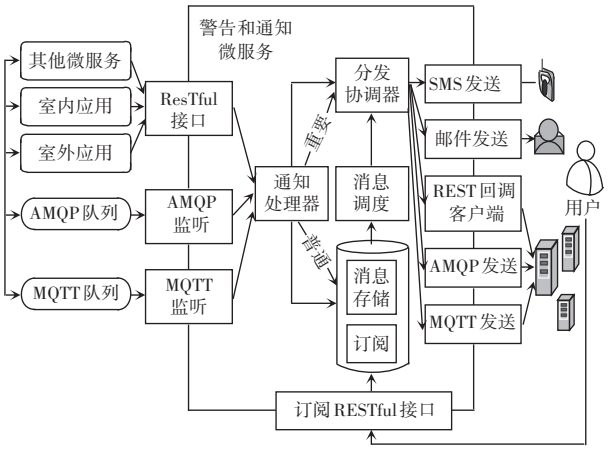


图 7　通知引擎模块图

Fig. 7　Notification engine module diagram

当警报和通知从任何接口接收通知时，通知将在内部传递给通知处理器。通知处理器首先保持接收通知，如果通知是关键的（severity = “critical”），则立即将其传递给分发协调器。对于普通通知（severity = “normal”），即等待消息调度程序批量处理。

当分发协调器收到通知时，首先查询订阅，以获取需要获取此通知及其接收通道信息的接收者。根据通道信息，分发协调器将此通知传递给相应的通道发送者。然后，通道发送者向订阅的接收者发送通知。

3.3.2　通知引擎交互

当接收到一个关键通知时，通知首先需要持续，并立即触发分发过程，更新通知状态后，通知将响应客户端以指示已接受通知，其高级交互图如图 8 所示。

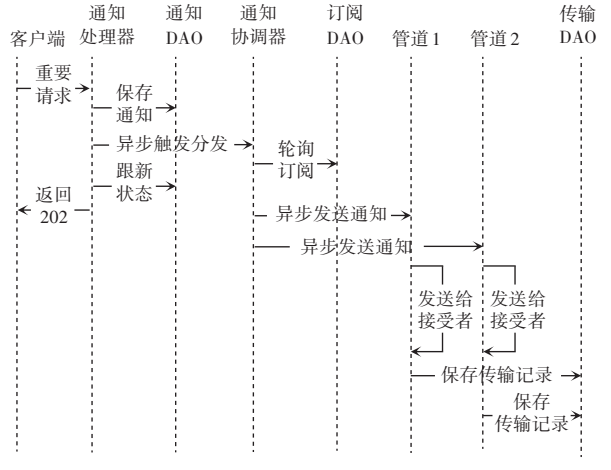


图 8　通知引擎交互图

Fig. 8　Notification engine interaction diagram

4　实验实施与结果分析

本文采用的是罗技 270 摄像头传感器，嵌入式开发板采用的是 openailab 公司的 EAI610-PO[8]，具体参数见表 1，安装 64 位的 Ubuntu16.04 操作系统，配置 Python+OpenCV 环境。EdgeX Foundry 是微服务架构，采用的是 golang 高并发语言设计，本文在开发板上配置 golang 环境，安装 Docker，docker-compose，运用 Docker[9] 虚拟化容器技术部署 EdgeX Foundry 各微服务。

表 1　EAI610-PO 硬件规格表

Tab. 1　EAI 610-PO hardware specification

Soc	RK3399
CPU	ARM 6 核 64 位处理器，基于 big.little 大小核架构 双核 Cortex-A72，最高 1.8 GHz
GPU	ARM Mail-T860 MP4 4 核 GPU 支持 OpenGL ES 1.1/2.0/3.0，OpenCL 1.2、DirectX11.1 支持 AFBC（帧缓冲压缩）
运行内存	双通道 LPDDR3（64-bit）4 GB
内置存储	16 GB 高速 emmc

4.1　视频采集处理模块

视频处理模块的核心是对运动物体的检测和跟踪[10]。视频处理流程如图 9 所示，进行运动物体检测和跟踪的算法的主要步骤包括：

（1）帧差计算：将相邻的两帧进行帧差计算，这是图像处理的基础，后续的步骤在此基础上进行。

（2）灰度图像转换：采集的视频图像是彩色图像，彩色图像包含红绿蓝（RGB）三个通道，图像的彩色对运动物体的检测与跟踪没有任何影响，但是却会增加图像处理系统的复杂度，所以将彩色图像转换为灰度图像，减小系统冗余。

（3）高斯模糊：指定高斯核的宽和高（必须是奇数），以及高斯函数沿 X，Y 方向的标准差。本文指定 2 个标准差都为默认值 0，函数会根据核函数的大小自行计算。高斯模糊的目的是为了将运动检测的目标的高斯滤波过滤掉，避免出现干扰。

（4）二值化图像：在图像转换成灰阶后，为了使图像更加简洁，经过二值化后的图像只剩黑白两色。

（5）腐蚀膨胀：腐蚀运算，即局部最小值运算，用于消除图像中不相关的细节。腐蚀即是将图像与核进行卷积，求出局部最小值，减小图像中的高亮区；膨胀与腐蚀是相反操作，膨胀运算，即局部最大值运算，用于桥接细节裂缝，其目的是为了前后两帧对比更加明显，有利于找到前景的轮廓。

（6）轮廓的寻找：使用 OpenCV 中的 cv2.findCountours 函数寻找轮廓，此函数可以将轮廓上的冗余去掉，节省了系统内存的开支。

(7)运动物体标记:计算轮廓的尺寸,并绘出外框将运动的物体标记,并保存图片,输出信息。

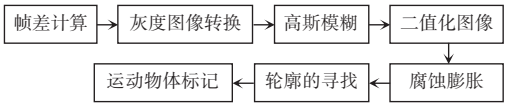


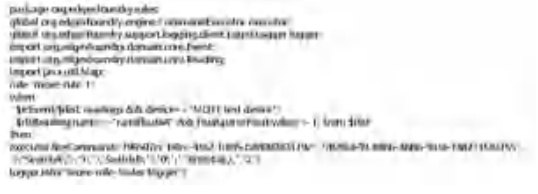
图 9 视频处理流程

Fig. 9 Video processing flow

4.2 报警模块

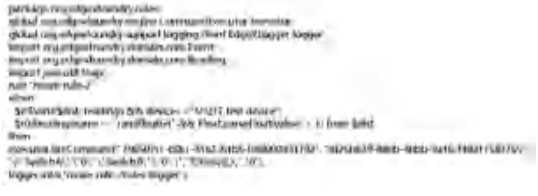
报警模块是基于 EdgeX Foundry 框架,将此框架编译成 ARM 架构,运用 Docker 虚拟化容器技术部署在 EAI610 开发板上,启动各个模块微服务,通过软件平台,进行设备的注册,本文选用 MQTT[11] 通讯协议,视频处理模块输出的信息通过 MQTT 传送至 EdgeX Foundry 边缘计算网关,信息存储至核心数据库,核心数据库通过 ZeroMQ[12] 发布数据,规则引擎微服务订阅数据,判断规则,通过核心命令微服务,向 Modbus 微服务发送命令,来控制继电器开关的闭合,与此同时向通知微服务发送命令,通知微服务将报警信息以邮件的方式发送给用户。

搭建好视频监控报警系统后,编写继电器电源开关的设备文件,在软件平台上添加设备,本文通过 Postman 的 post 命令上传规则,设定当视频处理模块返回的信息为 2 时,则触发边缘计算,继电器电源开关设定为打开(即设置值为 1),并向用户邮件发送报警信息;当视频处理模块返回的信息为 0 时,继电器电源开关设定为关闭(即设置值为 0)。在通知微服务里面注册通知的邮箱。生成了 2 个规则文件,如图 10 所示。



(a) 继电器开

(a) Relay on



(b) 继电器关

(b) Relay off

图 10 规则文件

Fig. 10 Rule file

4.3 实验结果

实验场景设置为:EAI610 开发板距离实验室的大门大概 5 m 的距离,人从门口走向设备,在此过程中,对人这个运动物体进行实时的检测与跟踪。

4.3.1 阈值对系统的影响

如图 11~图 14 所示,最小运动检测区域都取 5 000,有灯光照射下,阈值分别为 2,10,15,20,对比实验结果和对应的阈值处理的结果。

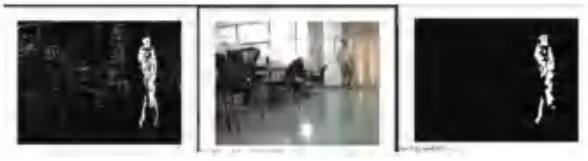


图 11 二值化阈值为 2

Fig. 11 Binary threshold is 2

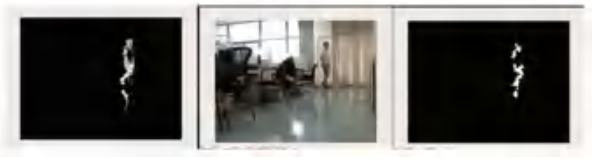


图 12 二值化阈值为 10

Fig. 12 Binary threshold is 10

图 13 二值化阈值为 15

Fig. 13 Binary threshold is 15

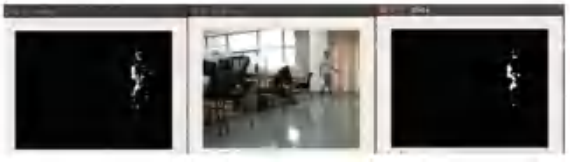


图 14 二值化阈值为 20

Fig. 14 Binary threshold is 20

从 4 幅图中可以看出,当阈值取 2 时,由于实验室噪声过多,一些被二值化后的噪声也被识别成了检测对象,造成白点过多。当二值化的阈值取值为 10 时,白点明显减少,且检测效果在 4 幅图中最好。当阈值过大时,检测效果开始变差,效果不是那么好。综上所述,在最小检测区域为 5 000 时,二值化阈值为 10,检测效果最好。

4.3.2 整体系统的时延测试

整体系统时延包括 2 个部分,分别为视频处理模块处理的时间 T_1, 报警模块包括 2 个部分,分别为触发蜂鸣器报警时间 T_{21} 与用户接收到报警邮件的时间 T_{22}, 最小报警时间为 $T_1 + \min(T_{21}, T_{22})$, 最

大报警时间为 $T_1 + \max(T_{21}, T_{22})$。本文做了 10 次实验见表 2，平均最大报警时延为 187.7 ms，平均最小报警时延为 135.5 ms，短信通知邮件如图 15 所示。

表 2　实验结果

Tab. 2　Experimental results

实验次数	视频处理模块时间 T_1/ ms	蜂鸣器报警时间 T_{21}/ ms	邮件报警时间 T_{22}/ ms	最大报警时间为 $T_1 + \max(T_{21}, T_{22})$ / ms	最小报警时间为 $T_1 + \min(T_{21}, T_{22})$ / ms
1	53	129	78	182	131
2	45	125	80	170	125
3	56	132	76	188	132
4	54	130	81	184	135
5	58	134	79	192	137
6	62	129	76	191	138
7	64	136	79	200	143
8	60	128	84	188	144
9	56	137	81	193	137
10	58	131	75	189	133

图 15　邮件通知

Fig. 15　Email notification

5　结束语

本文将视频监控报警技术与边缘计算框架结合起来，采用微服务架构，虚拟化容器部署，设计了基于边缘计算的视频监控报警系统。针对不同的参数，进行对比实验，分析出最适合的参数。相对于其他的视频监控报警系统，本系统没有采用 GPRS 短信模块，运用边缘计算框架中的规则引擎与通知微服务，实现了边缘端的规则处理，减轻了云端的压力，同时降低了时延，微服务架构更加灵活，部署起来更加方便。实验结果充分证明了系统的稳定性与可靠性，为视频监控报警系统的设计提供了一个新的思路，与此同时为边缘计算的实施提供了参考，具有一定的参考价值。

参考文献

[1] 刘楠. 基于 Android 平台的远程视频监控报警系统研究[D]. 长春：吉林大学，2016.

[2] 孟庆博. 基于 Android 平台视频实时监控系统的设计与实现[D]. 长春：吉林大学，2016.

[3] SATYANARAYANAN M. The emergence of edge computing[J]. Computer, 2017, 50(1):30.

[4] SHI W, CAO J, ZHANG Q, et al. Edge computing: Vision and challenges[J]. IEEE Internet of Things Journal, 2016, 3(5):637.

[5] 吴双. 基于 MFC+OpenCV 的视频监控区域入侵检测系统设计与实现[D]. 济南：山东师范大学，2017.

[6] 施巍松，孙辉，曹杰，等. 边缘计算：万物互联时代新型计算模型[J]. 计算机研究与发展，2017，54(5)：907.

[7] NA W, LEE Y, DAO N N, et al. Directional link scheduling for real-time data processing in smart manufacturing system[J]. IEEE Internet of Things Journal, 2018,5(5):3661.

[8] 佚名. 瑞芯微首发 ARM 架构人工智能开发平台[J]. 智能城市，2018，4(17)：9.

[9] 李东光，刘智平，姜雨菲. 蚁群优化算法的 Docker 集群调度策略[J]. 西安工业大学学报，2019，39(3)：330.

[10] 韩宇，张磊，吴泽民，等. 基于嵌入式树莓派和 OpenCV 的运动检测与跟踪系统[J]. 电视技术，2017，41(2)：6.

[11] 胡存，骆德汉，童怀. 基于 Modbus 与 MQTT 融合工业能耗网关系统设计[J]. 物联网技术，2019，9(4)：49.

[12] 崔晓佳，张云，刘锦锋. 基于 ZeroMQ 的教学资源存储系统的设计[J]. 现代计算机(专业版)，2018(28)：96.

(上接第 195 页)

[16] 常向魁. 视频运动目标跟踪算法研究[D]. 开封：河南大学，2007.

[17] 程爱灵，黄昶，李小雨. 运动目标检测算法研究综述[J]. 信息通信，2017(1)：12.

[18] 彭艳芳. 视频运动目标检测与跟踪算法研究[D]. 武汉：武汉理工大学，2010.

[19] MISHRA S K, BHAGAT K S. Human motion detection and video surveillance using MATLAB [J]. International Journal of Scientific Engineering and Research (IJSER), 2014, 3(7): 154.

[20] CHO H, RYBSKI P E, BAR-HILLEL A, et al. Real-time pedestrian detection with deformable part models [C]//2012 IEEE Intelligent Vehicles Symposium (IV). Alcala de Henares, Spain: IEEE, 2012: 1035.

[21] 朱宏. 基于视频序列的运动目标检测与跟踪技术研究[D]. 成都：西南交通大学，2008.

[22] KANKANHALLI M S, MEHTRE B M, WUR K. Cluster-based color matching for image retrieval[J]. Pattern Recognition, 1996, 29(4):701.

[23] LIANG Shen, RANGAYYAN R M, LEO DESAUTELS J E. Application of shape analysis to mammo-graphic calcification [J]. IEEE Transactions on Medical Imaging, 1994, 13(2):263.