基于 OMAP 和 Gstreamer 的网络视频监控系统

陈 乐,吴 蒙
(南京邮电大学 通信与信息工程学院,江苏 南京 210003)

摘 要:随着人们生活水平的逐渐提高,人们对安全越来越重视,视频监控在很多领域逐渐普及。在 OMAP 平台上运用 Gstreamer 多媒体框架,实现了一个集视频采集、视频编码和视频传输的 C/S 结构网络视频监控系统。视频采集基于 V4L2 应用程序接口,视频编码采用 H.264 编码,网络传输承载于 RTP 协议族之上。系统包含服务端和客户端,基于插件的系统结构使该系统具有易维护、易扩展的特点。在 OMAP 平台上的测试结果表明,该系统整体延时在 1 s 以下,满足了实时监控需求,可应用于实际生产。

关键词:网络视频监控;Gstreamer;OMAP;实时传输协议;H.264
中图分类号:TP302 **文献标识码**:A **文章编号**:1673-629X(2016)09-0095-04
doi:10.3969/j.issn.1673-629X.2016.09.022

Network Video Surveillance System Based on OMAP and Gstreamer

CHEN Le, WU Meng
(College of Telecommunication & Information Engineering, Nanjing University of Posts and Telecommunications, Nanjing 210003, China)

Abstract: With the gradual improvement of people's living standards, people have paid more attention to security, and video surveillance is gradually spread around many fields. In this paper, a network video surveillance system with C/S structure including video capture, encoding and transmission is developed under OMAP platform based on Gstreamer multimedia framework. Video capture is based on V4L2 application program interface, video encoding is using H.264 coding standard, and network transmission is based on the RTP family. This system includes server and client, and it's easy to maintain and expand because of the plugin-based structure. Test results on the OMAP platform show that the overall system delay of about 1 seconds or less, can meet the needs of real-time monitoring, which can be used in actual production.

Key words: network video surveillance; Gstreamer; OMAP; RTP; H.264

0 引 言

视频监控以其直观方便、信息内容丰富而广泛应用于众多领域[1]。近年来,随着计算机网络和嵌入式技术的发展,视频监控也逐渐向网络化、嵌入式发展。文中设计了一种基于 C/S 结构的网络视频监控方案,在 OMAP4460 平台上运用 Gstreamer 框架,组建了集视频采集、视频编解码、网络传输于一体的视频监控系统,并对系统进行了测试分析。

1 系统硬件结构

系统采用基于 TI 高性能 OMAP4460 平台[2] 的 Pandaboard ES 开发板,平台核心搭载了一颗双核 ARM Cortex-A9 处理器,45 nm 制程,具有较高的能效,工作频率最高可达 1.5 GHz,支持对称多处理(Symmetrical Multi-Processing)技术,每个中央处理单元附带一个支持单指令多数据(Single Instruction Multiple Data)指令集的 Neon 协处理器;IVA 3 硬件加速模块支持多标准视频编解码器加速,最高分辨率达 1 080 p;图像信号处理器支持高质量图像和视频抓取;SGX530 图形加速器提供惊人的 2D/2D 图形表现。视频数据交由 DSP 核心及 GPU 处理,视频传输、逻辑控制处理交由 ARM 处理器,通过分工可以使工作效

收稿日期:2015-11-24 **修回日期**:2016-03-04 **网络出版时间**:2016-08-01
基金项目:国家"973"重点基础研究发展计划项目(2011CB302900);江苏省高校自然科学研究重点项目(10KJA510035);南京市科技发展计划重大项目(201103003)
作者简介:陈 乐(1991-),男,硕士研究生,研究方向为无线通信与信号处理;吴 蒙,教授,研究方向为无线通信与信号处理、无线网络安全与通信系统的信息安全。

率大大提高。摄像头采用基于 OV5640 芯片的 e-CAM51_44x 模块,500 万像素,支持自动对焦,支持多种帧率以及分辨率下的视频拍摄(720 p/60 fps,1 080 p/30 fps),还集成了符合 IEEE 802.11 b/g/n 标准的无线网卡,拓展了系统的无线组网能力。

2 Gstreamer 多媒体框架

Gstreamer[3]是一个基于插件和管道的开源多媒体应用框架。在 Gstreamer 下,开发人员可以很方便地创建各种类型的多媒体功能插件,例如:多媒体采集、多媒体回放、多媒体编辑等。再利用数据管道式处理的思想把各个功能插件的数据流串联起来组成插件管道就可以构成不同的多媒体应用。Gstreamer 开发有如下几个核心概念:

2.1 元 件

元件(element)是 Gstreamer 中最常见的概念,它是构成管道的基本单元,也是框架中所有可用组件的基础。可以把它看成一个具有独立功能的黑匣子,通过输入输出接口与其他元件进行通信。根据元件功能上的不同可以将其分为三类:数据源元件,只有输出端没有输入端,仅仅只是负责数据的生成,而不对数据进行任何处理,通常位于应用管道的首部;过滤器元件,既有输入端又有输出端,元件从管道上游的数据源获取数据流,并将数据流经过自身处理后传送到输出端;接收器元件,只有输入端没有输出端,负责数据的表现,往往处在应用多媒体管道的尾部[4]。

在 Gstreamer 中创建 GstElement 对象通常使用工厂方法。Gstreamer 提供了很多类型的元件工厂对象(GstElementFactory)对应不同的元件对象,工厂对象由 gst_element_factory_find(factory_name)函数获取,获得工厂对象后可以调用 gst_element_factory_create(factory,element_name)函数来创建元件,当元件不再使用时,需要对元件占用资源进行释放,调用函数 gst_element_unref(element)。

2.2 衬 垫

衬垫[5](pad)是元件间交流的接口。数据流流经一个元件需要通过元件的输入衬垫和输出衬垫。衬垫可以形象地理解为一个元件的插座,只有当两个衬垫允许通过的数据类型一致时,两个插座公母才能配对,元件之间的链接才被建立。可以看出,衬垫的功能决定了一个元件所能处理的媒体类型。利用函数 gst_element_get_pad(element,dst)和 gst_element_get_pad_list(element)可以获取元件的衬垫,利用函数 gst_pad_get_parent()可以获得衬垫对应的元件。

衬垫的功能通过 GstCaps 对象来描述,媒体类型由 GstStructure 对象描述,一个 GstCaps 对象通常包含一个或多个 GstStructure,表示该衬垫支持哪些媒体类型[6]。衬垫功能可以由函数 gst_pad_get_caps()获取。

2.3 箱 柜

箱柜是一个可以装载元件的容器。在一些功能较为复杂的管道中,经常使用箱柜将多个元件组合成一个逻辑单元,通过直接操作箱柜来改变元件状态,从而简化操作。同时箱柜会对数据流进行优化从而提高媒体性能。操作箱柜可以像操作普通元件一样,使用起来十分方便。

在 Gstreamer 中,箱柜有两种类型:GstPipeline 和 GstThread。Gstreamer 应用程序顶层必须为一个 GstPipeline 管道。GstThread 可以提供同步处理机制,在应用程序需要严格同步时可以使用。下面代码展示了使用工厂和特定函数产生箱柜的方法:

```
GstElement *thread, *pipeline;
thread=gst_element_factory_make ("thread",NULL);
pipeline=gst_pipeline_new("pipeline_name");
```

往箱柜中添加元件可以调用函数 gst_bin_add(),删除元件使用 gst_bin_remove()函数。箱柜没有自己独立的输入衬垫和输出衬垫,但是 Gstreamer 为它引入了精灵衬垫的概念。精灵衬垫映射到箱柜中元件的衬垫上,具有精灵衬垫的箱柜在行为上与元件是完全相同的,所以可以像操作元件一样操作箱柜。

3 相关技术

3.1 V4L2

V4L(Video for Linux)为 Linux 下视频设备的一个驱动程序框架,同时,它也为 Linux 应用程序提供了视频设备操作的应用程序接口(API)。V4L2[7](Video for Linux 2)是 V4L 的第二个版本,修正了 V4L 在一些设计上的缺陷。

V4L2 在结构上可以分为两层:上层是面向应用程序的 API,下层是面向硬件的视频设备驱动层。从应用层看,V4L2 屏蔽了底层硬件操作细节,使应用可以在统一的接口下操作不同的硬件设备。从驱动程序看,V4L2 提供了良好的内存管理和数据时序性管理,很好地适应了视频数据的特点[8]。基于 V4L2 的视频采集流程图如图 1 所示。

3.2 H.264

H.264 编码标准是由 ITU-T 和 MPEG 组成的联合视频组(Joint Video Team,JVT)开发制定,是目前代表最新技术水平的视频编码技术之一。与以往的标准如 H.263、MPEG-4 第二部分相比,H.264 可以在相同质量视频呈现的前提下节约一半或更多的带宽,计算复杂度增加幅度也比较小。H.264 提出了一个网络抽象层(NAL)概念,简化了编码内容的封装,对网络十分友

好。H.264 还提供多种档次的编码选择和参数定制,具有很强的应用灵活性,能适配各种带宽、分辨率条件,可以应用于各种场景和系统,如视频点播、可视电话、视频存储、广播等。

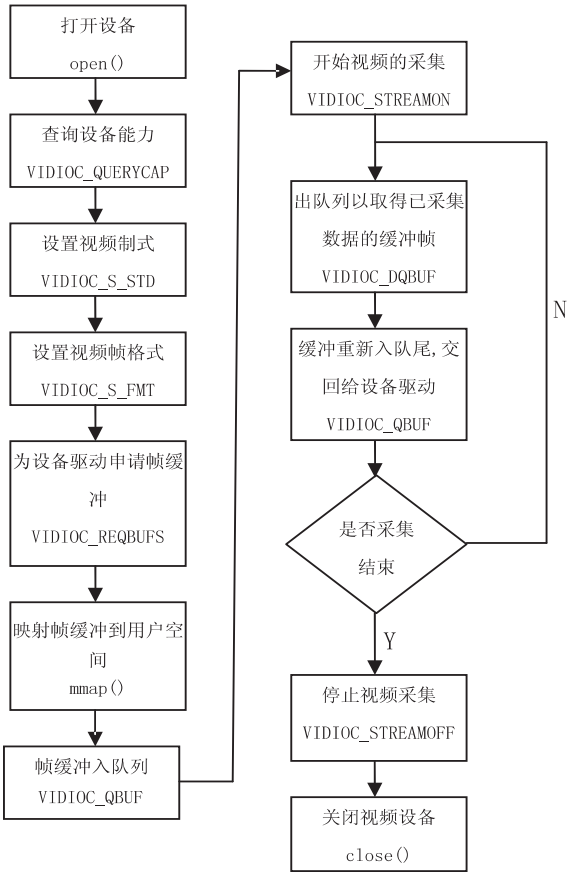

图 1　基于 V4L2 的视频采集流程图

目前,开源的 H.264 编解码器主要有:JM、T264 和 X264。JM 是 H.264 官方提供的编解码器参考模型,支持几乎所有 H.264 特性。但是 JM 程序只考虑了对所有特性进行实现却忽略了其实用性,导致编码复杂度非常高,实际应用起来编解码速度很慢,常常只用于学术研究[9]。T264 是由国内开源的 H.264 编解码器实现。吸收了 JM 和 X264 的一些优点,但是其编码码流不兼容 H.264,其他解码器无法解码 T264 的码流,限制了它的应用,目前 T264 已经停止开发。X264 是由网上自由组织联合开发的兼容 H.264 标准码流的编码器。它注重实用的特性,抛弃了 H.264 标准中一些对编码性能有贡献却异常复杂的编码特性,例如任意条带顺序 ASO、冗余条带、数据分割等。使得实际编码过程中在不明显降低编码性能的前提下,大幅降低了编码复杂度。与 JM 相比,在比特率相差不到 5% 的情况下,X264 的编码速度比 JM 快将近 50 倍[10]。因此,最终选定 X264 作为系统编码解决方案。

3.3　RTP

实时传输协议是一个网络传输协议,它是由 IETF 的多媒体传输工作小组 1996 年在 RFC 1889 中公布的。它包括两部分:实时传输协议(RTP)和实时传输控制协议(RTCP)。其中,RTP 负责包装媒体流、提供数据的实时阐述,通过提供时间戳信息来保持收发端的数据同步;RTCP 为 RTP 媒体流提供信道外控制,为 RTP 的服务质量(Quality of Service)提供反馈[11]。

RTP 是一个多播协议,主要承载于 UDP 协议之上[12]。一般的 RTCP 对 RTP 的 QoS 反馈场景为:会话开始后,发送端应用程序会周期性发送端报告(SR),接收端根据 SR 中的信息和自身已接收数据进行对比,计算生成接收端反馈报告,通过接收端报告(RR)发送给被反馈的发送端,发送方根据接收方服务质量反馈,动态调整发送速率以改善传输[13]。图 2 展示了 H.264 视频基于 RTP 的传输方案。

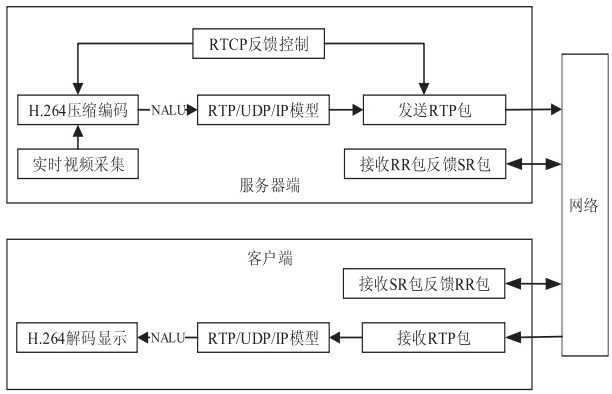

图 2　H.264 基于 RTP 的传输方案

4　基于 Gstreamer 的服务器端和客户端实现

4.1　服务器端

服务器端通过 v4l2 插件从摄像头采集原始视频流,根据摄像头模块的参数,设定分辨率为 1 280 * 720、帧率为 24 fps,采集到的原始视频流先送入数据队列插件 queue 进行缓冲,然后通过 videorate 插件将视频帧进行帧率调整,得到帧率稳定的视频数据。此时,视频数据的色彩空间还是 RGB 模式,而 x264 编码器只支持 YUV 模式,所以视频数据还需要通过 ffmpegcolorspace 插件进行色彩空间转换。经过色彩空间转换后,在 x264enc 插件中使用基本档次进行编码。编码得到的 NAL 单元通过 rtph264pay 插件进行 RTP 打包,之后进入 RTP 箱柜进行 RTP 协议封装。最后,利用 udpsink 插件将 RTP 数据包封装成 UDP 数据报发送到网络。服务器端 RTP 箱柜在生成 RTP 数据包的同时收送和周期性发送 RTCP 包,利用客户端发送的接收端报告动态地调节编码和传输速率,同时向客户端发送发送端报告。服务器端接收 RTCP 包的端口选择为 UDP 端口 5002。管道示意图见图 3。

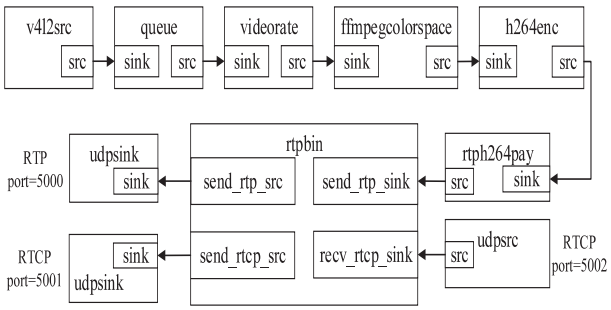

图 3 服务器端管道示意图

4.2 客户端

客户端使用 udpsrc 插件接收来自服务器端的 RTP 包和 RTCP 包,接收 RTP 包的端口设置为 UDP 端口 5000,接收 RTCP 包的端口设置为 UDP 端口 5002。UDP 数据包在 GStreamer 框架下以 Gstbuffer 的形式传递给 RTP 箱柜,完成对 RTP 数据包的解封装。接着在 rtph264depay 中将 RTP 包转化成 NAL 单元序列,NAL 单元序列在 h264dec 插件中解码,解码后的视频色彩空间为 YUV 模式,随后数据流流向 ffmpegcolorspace 插件,在 ffmpegcolorspace 插件中将视频色彩空间转换成 RGB 模式。最后在 ximagesink 插件中进行视频播放。客户端 RTP 箱柜在接收 RTP 数据包的同时收送 RTCP 包,利用 RTCP 接收端报告向服务器端反馈质量服务信息。管道示意图见图 4。

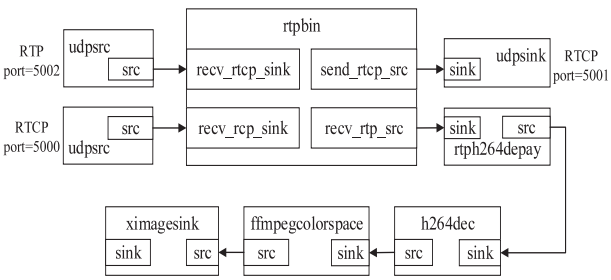

图 4 客户端管道示意图

5 系统测试

系统功能调试使用 Gstreamer 提供的调试工具 gst-lauch 进行。服务器端为 Pandaboard Es 开发板,IP 地址为 192.168.199.100,客户端为搭建好 Gstreamer 环境的 Linux 主机,IP 地址为 192.168.199.101。

服务器端执行调试指令:

```
$ gst-launch - v gstrtpbin name=rtpbin \
v4l2src device=/dev/video0 ! "video/x-raw-yuv, width=1280 height=720 framerate=24/1" ! \
videorate ! ffmpegcolorspace ! x264enc profile=baseline byte-stream=true ! rtph264pay ! \
rtpbin.send_rtp_sink_0 rtpbin.send_rtp_src_0 ! udpsink host=192.168.199.101 port=5000 \
udpsink host=192.168.199.101 port=5002 ! rtpbin.send_rtcp_sink_0 \
rtpbin.send_rtcp_src_0 ! udpsink port=5001
```

客户端执行调试指令:

```
$ gst-launch -v gstrtpbin name=rtpbin \
udpsrc caps="application/x-rtp,media=(string)video,clock-rate=(int)90000,\
encoding-name=(string)H264" port=5000 ! \
rtpbin.recv_rtp_sink_0 rtpbin.recv_rtp_src_0 ! rtph264depay ! ffdec_h264 ! ffmpegcolorspace ! ximagesink \
udpsrc port=5002 ! rtpbin.recv_rtcp_sink_0 rtpbin.send_rtcp_src_0 ! \
udpsink port=5001 host=192.168.199.100
```

经测试,视频能实时稳定传输,传输时延在 0.5 s 左右,满足了实时应用要求。视频传输实时性验证如图 5 所示。

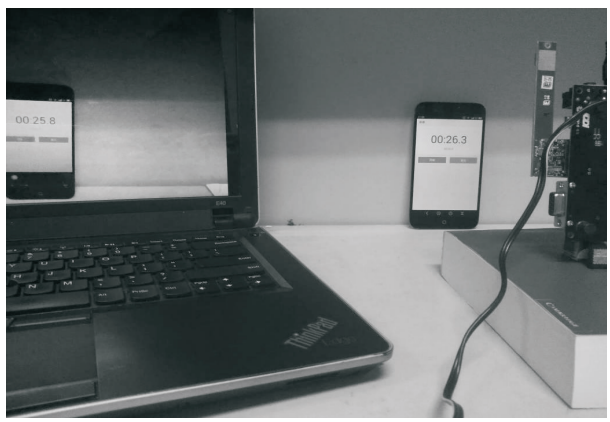

图 5 视频传输实时性验证

6 结束语

文中提出一种基于 OMAP4460 平台,在 Gstreamer 框架下实现的实时视频监控系统的方案。该方案具有高效率、低能耗、适应性强、易维护扩展等特点,为网络实时视频监控系统提供了一条新思路。

参考文献:

[1] 骆云志,刘治红. 视频监控技术发展综述[J]. 兵工自动化,2009,28(1):1-3.

[2] Texas Instruments. OMAP4460 multimedia device technical reference manual[M]. [s.l.]:Texas Instruments,2011.

[3] Taymans W, Baker S, Wingo A, et al. GStreamer application development manual[EB/OL]. 2008-10-31. http://gstreamer.freedesktop.org/data/doc/gstreamer/head/manual/manual.pdf.

[4] 洪承煜. 基于 GStreamer 嵌入式多媒体系统的设计与实现[D]. 成都:成都理工大学,2009.

[5] Boulton R J, Walthinsen E, Baker S, et al. GStreamer plugin writer's guide[EB/OL]. 2010-07-27. http://www.gstreamer.net/data/doc/gstreamer/head/pwg/pwg.pdf.

(下转第 103 页)

文中使用第 2 种方法来部署 Hadoop,使用的 Hadoop 版本是 2.5.2。

主要完成如下几步:

(1)下载基础系统镜像;

(2)使用 Dockerfile 的内建指令下载安装软件;

(3)使用 Dockerfile 内建指令加载对应的配置文件。

同时,由于 Docker 技术本身还在发展过程中,要使用 Docker 创建能够被外网访问的容器,需要进行一些额外设置。利用容器的底层实现原理,通过脚本动态地为每个容器创建对应的桥接网口。这样能够实现跨服务器间的容器间的互相访问,也更接近真实的应用环境。

通过使用 Dockerfile 文件,完成构建基于 Docker 的 Hadoop 环境。经过测试,Hadoop 能够正常使用。把其与安装在物理主机上的 Hadoop 进行比较,通过 time 命令测试两者运行时间,结果显示两者耗时相差不大,部分情况下 Docker 下的耗时才略高一些。虽然部署的 Docker 数量不多,但能够看出在 Docker 上部署的 Hadoop 在数据读写上的表现非常优异。

5　结束语

Hadoop 作为需要大量读写数据的云计算平台,部署在 Docker 下比部署在传统的虚拟机上有更好的性能表现。文中验证了在 Docker 上读写数据的性能比 KVM 的更高,同时,在 Docker 上部署 Hadoop 具有接近物理主机的资源利用率。并且,由于 Docker 把数据与运行环境进行了分离,所以可以把构建好的 Hadoop 平台作为镜像发布,方便随时添加或替换 Hadoop 节点,这能够提高部署 Hadoop 时的工作效率。Docker 作为容器技术的最佳实践,其性能优势使得其能够替换现有的虚拟化技术。现在国内已经有结合 Docker 与 OpenStack 的研究(如文献[13])和基于 Docker 的 PaaS 平台研究(如文献[14]),这将使得 IaaS 与 PaaS 的界限变得更加模糊,Docker 势必会引领下一场云计算技术的浪潮。

参考文献:

[1] 杨保华,戴王剑,曹亚仑. Docker 技术入门与实战[M]. 北京:机械工业出版社,2014.

[2] 肖德时. 深入浅出 Docker[EB/OL]. [2015-01-05]. http://www.infoq.com/cn/articles/docker-core-technology-preview?utm_source=infoq&utm_medium=related_content_link&utm_campaign=relatedContent_articles_clk.

[3] Compton D. Why Docker and CoreOS' split was predictable [EB/OL]. [2015-01-05]. http://danielcompton.net/2014/12/02/modular-integrated-docker-coreos.

[4] Lowy G. Application performance management enables DevOps ROI[EB/OL]. [2015-01-05]. http://www.apmdigest.com/application-performance-mangent-apm-devops-roi.

[5] Garber L. News briefs[J]. IEEE Security and Privacy,2011,9(6):9-11.

[6] Swan C. Docker: present and future[EB/OL]. [2015-01-05]. http://www.infoq.com articles/docker-future.

[7] Kavis M. Blurring the line between PaaS and IaaS[EB/OL]. [2015-01-05]. http://www.forbes.com/sites/mikekavis/2014/06/02/blurring-the-line-between-paas-and-iaas/.

[8] Shalom N. Do I need OpenStack if I use Docker[EB/OL]. [2015-01-05]. http://pensource.com/business/14/11/do-i-need-openstack-if-i-use-docker.

[9] Dua R,Raja A R,Kakadia D. Virtualization vs containerization to support PaaS[C]//Proc of IEEE international conference on cloud engineering. Boston,MA:IEEE,2014:610-614.

[10] 黄　刚,徐小龙,段卫华. 操作系统教程[M]. 北京:人民邮电出版社,2009.

[11] Bica M,Bacu V,Mihon D,et al. Architectural solution for virtualized processing of big earth[C]//Proc of IEEE international conference on ICCP. Cluj Napoca: IEEE,2014:399-404.

[12] Turnbull J. The Docker book[M]. [s. l.]: Amazon Digital Services,Inc.,2014.

[13] 张忠琳,黄炳良. 基于 OpenStack 云平台的 Docker 应用[J]. 软件,2014,35(11):73-76.

[14] 鞠春利,刘印锋. 基于 Docker 的私有 PaaS 系统构建[J]. 轻工科技,2014(10):80-80.

(上接第 98 页)

[6] 刘兴民,赵连军. 基于 GStreamer 的远程视频监控系统的关键技术研究[J]. 计算机应用与软件,2011,28(5):243-246.

[7] 徐　家,陈　奇. 基于 V4L2 的视频设备驱动开发[J]. 计算机工程与设计,2010,31(16):3569-3572.

[8] 刘登诚,沈苏彬,李　莉. 基于 V4L2 的视频驱动程序设计与实现[J]. 微计算机信息,2011,27(10):56-58.

[9] 刘喜龙,石中锁. 基于 H264 的嵌入式视频服务器的设计[J]. 微计算机信息,2005,21(1):133-134.

[10] 李世平. H.264 三大开源编码器之评测报告[R/OL]. 2005. http://blog.csdn.net/sunshine1314/archive/2005/06/19/397895.

[11] 刘　浩,胡　栋. 基于 RTP/RTCP 协议的 IP 视频系统设计与实现[J]. 计算机应用研究,2002,19(10):140-143.

[12] 刘尚麟,刘　军. GStreamer RTP 插件的改进及应用[J]. 信息安全与通信保密,2009(1):91-92.

[13] 孙彦景,李世银,董　杨. 基于 RTP 的嵌入式网络化视频采集压缩系统[J]. 计算机工程与设计,2006,27(16):2939-2942.