基于 V4L2 的远程视频采集系统设计与实现

李珊珊，王绪国
（武汉理工大学信息工程学院，武汉 430063）

摘要：本文探讨了一种基于 ARM 的视频采集与传输系统的实现方案。针对市面上普遍应用的中星微摄像头芯片，在 Linux 操作系统下对 V4L2 视频采集流程进行了深入的研究。系统以 S3C2440 高性能微处理器为核心，外围扩展 USB 接口、网口，实现 USB 摄像头实时捕获视频图像并通过网络传送给上位机的功能。系统结构简单，性能可靠，运行良好，在视频监控中具有广泛的应用前景。

关键词：视频采集；V4L2；H.264；格式转换
中图分类号：TN919.81

Design and Implementation of Remote Video Capture System Based on V4L2

LI Shanshan, WANG Xuguo
(School of Information Engineering, Wuhan University of Technology, Wuhan 430063)

Abstract: This article explores an implementing method of video acquisition and transmission based on ARM. It researches deeply on Linux kernel driver V4L2 in the process of video collection and chiefly aims at Vimicro camera chip which has been widely used. The core of this system is high-performance processor S3C2440, together with peripheral components, USB port and network port, to achieve the function of capturing real time video images by USB camera and transmitting them to the host computer. This system is simple structured and running well, it has reliable performance and will have a broad foreground in video monitoring.

Key words: video capture;V4L2;H.264;format conversion

0 引言

视频的实时采集与压缩传输是视频监控和视频通信等应用开发的必要步骤。由于嵌入式 Linux 具有低成本、多种硬件平台支持、优异的性能和良好的网络支持[1]以及对各种文件系统完备支持等特点，被广泛应用于各种嵌入式设备的开发中。结合 V4L2 驱动的 USB 接口的视频捕捉设备作为 Linux 系统下的一种通用视频架构得到了广泛的支持和应用。

USB 摄像头采集数据的流程为：传感器——图像信号处理——图像压缩——USB 传输。一般 USB 摄像头使用动态分配 USB 带宽技术，为了让 DSP 采集的图像在 USB1.1 上传输效率提高，对传感器采集的图像都作了硬件压缩，直接输出 JPEG 图像。在视频压缩编码方面，新一代的视频编解码标准 H.264，具有优异的压缩性能，良好的网络亲和性，能够在传输过程中节省带宽，让视频实时传输并流畅播放。

本文使用的是开源编码器 T264 对采集的数据进行编码，而编码器要求原始数据为 YUV 格式，这样就涉及到 JPEG 数据格式到 YUV 格式之间转化问题。本文对此进行了深入的探讨。

作者简介：李珊珊（1986-），女，主要研究方向：信息采集、传输与处理. E-mail: lishanshan-033@163.com

1 系统总体方案介绍

1.1 硬件结构

系统开发基于 ARM9 系列开发板，其中主控芯片采用 S3C2440，主频高达 400MHz，16Kb 指令和数据 Cache，具有 MMU 管理单元、外部存储器、控制器以及 64MB NAND Flash 和 64MB SDRAM 存储单元。系统主要涉及串口、网口和 USB 接口等外部器件。整体硬件结构如图 1 所示。

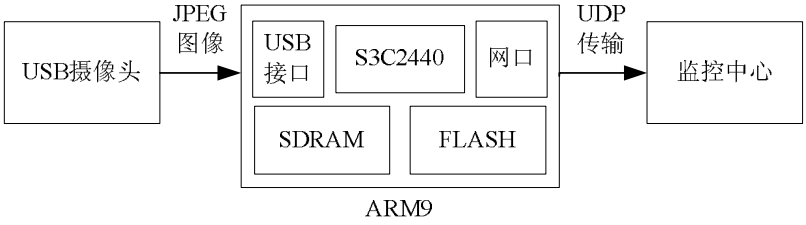

图 1 系统硬件结构

Fig.1 hardware structure of the system

1.2 软件流程

Gspca 是 Linux2.6 内核中自带的万能 USB 摄像头驱动，能支持市面上绝大多数 USB 摄像头。V4L2（全称 Video For linux 2)作为内核针对视频设备的驱动，为视频应用程序提供统一的 API 接口，通过标准的系统调用即可操作各种不同的视频捕捉设备。本文采用带 JPEG 硬压缩的 zc301 摄像头进行视频采集与压缩传输，系统的软件流程图如图 2 所示。

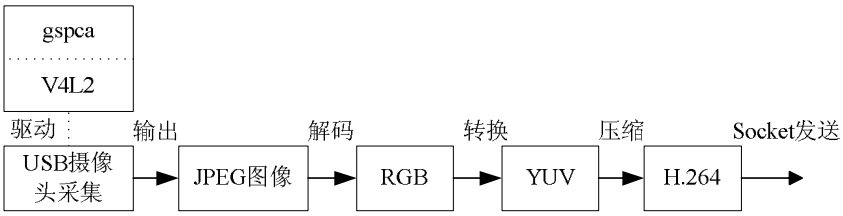

图 2 系统软件流程

Fig.2 software process of the system

2 各模块设计与实现

2.1 V4L2 视频采集

V4L 是 Linux 系统中关于视频设备的应用程序接口，它分为驱动和编程接口两层。驱动与内核、硬件设备打交道，而编程接口为应用程序提供函数调用。由于在早期版本的 Linux 内核中，V4L 有很多的缺陷，Bill Dirks 等人对其进行了重新设计。作为 V4L 的升级版 V4L2，两者并不兼容，但是 V4L2 为 Linux 下视频应用程序开发提供更好的可扩展性和灵活性。在 Linux 中，将视频设备看作普通的设备文件，可以对其进行读写操作。对整个采集流程本文做如下说明：

1. 摄像头作为一个设备文件被打开后，可以读取设备信息；查看设备具备哪些功能；选择视频输入；设置视频制式等，本文省略描述。

在设置视频帧格式时，可以查看内核关于视频设备的头文件 Videodev2.h 中，对 RGB24、RGB565、RGB32、YUV420、YUV422、YUYV、JPEG、MPEG、MJPEG 等多种格式都作了定义，表明驱动是认识这些数据格式的。但也需要根据摄像头芯片自身所支持的格式来设定，比如 ov511 摄像头一般输出格式为 RGB24/YUV420，本文选用的摄像头输出格式为

JPEG，宽和高设为 320*240。

2. 在视频采集前期还需要做一些准备工作：

(1) 为采集的视频数据申请内存（帧缓冲），申请 req.count 个缓冲区（一般不超过 5）。

(2) 把分配到的缓冲区转换为物理地址，获取这些地址。我们知道 V4L2 采集的数据是存放在内核空间的，而内核空间的数据，应用程序无法访问。我们利用 mmap 函数将这些地址映射到用户空间。这样程序可以直接对缓冲区的数据操作，简单而高效，省去了 read，write 繁琐的步骤，极大的缩短了数据交互的时间。

(3) 将 req.count 个缓冲区放入队列，等待驱动向缓冲区填入视频数据。数据缓存采用 FIFO 的方式，当应用程序调用缓存数据时，缓存队列将最先采集到的视频数据缓存送出，并重新采集一张新的视频数据。每个缓冲区存放一帧数据。

3. 开始采集，从下图我们得知，帧缓冲出列，实际上是驱动将一帧视频数据的处理权利交给应用程序，是应用程序可以对缓冲区进行操作的唯一时间。此时可以取出一帧视频数据做格式转换、压缩、传输等处理。处理完毕后，将空的缓冲区再次放进传入队列。

视频采集流程的主要步骤如图 3 所示。

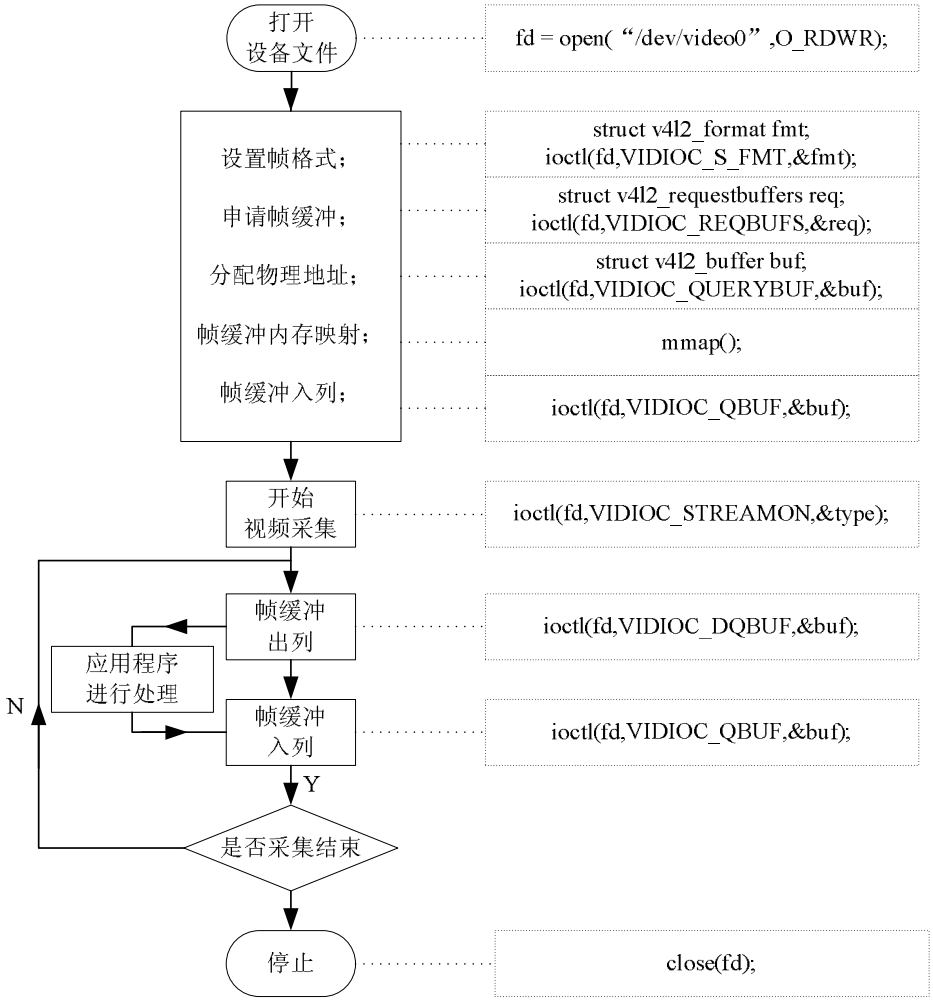

图 3 V4L2 视频采集

Fig.3 video capture using V4L2 method

2.2 图像格式转换

2.2.1 JPEG 解码得到 RGB 分量

一个 JPEG 文件大体上可以分为两个部分：标记码（Tag）和压缩数据，标记码由两个字节构成，其前一个字节是固定值 0xFF，在每个标记前可以添加数目不等的填充字节 0xFF，标记码一般由 8 个部分组成，如表 1 所示。

表 1 JPEG 定义的标记
Tab. 1 JPEG Tag

符号	*标记代码*	*含义*
SOI(Start of Image)	0xD8	图像开始
APP0	0xE0	JFIF 应用数据块
APPn	0xE1~0xEF	其他数据块(n=1～15)
DQT(Define Quantization Table)	0xDB	量化表
SOF(Start of Frame)	0xC0	帧图像开始
DHT(Define Huffman Table)	0xC4	霍夫曼表
SOS(Start of Scan)	0xDA	扫描开始
EOI(End of Image)	0xD9	图像结束

对 JPEG 图像解码需要 libjpeg 库的支持。libjpeg 是一个广泛使用在 Unix 类系统下的 JPEG 压缩/解压缩函数库，通过 libjpeg 库，应用程序可以每次从 JPEG 压缩图像中读取一个或多个扫描线（scanline）。所谓扫描线，是指由一行像素点构成的一条图像线条。

在一帧图像的二进制数据中，找到 0xFFD8，它是图像开始标记。从此处开始将后面的数据写入一个文件 infile，利用 libjpeg 库中的函数 jpeg_create_decompress 对 infile 进行解码，再利用 jpeg_read_scanlines 函数将解码后的数据一行一行的扫描取出，得到各个像素的 RGB 分量。每个分量用 1 个字节表示，取值范围从 0 到 255，则一个像素点需要 3 字节的空间存储。这三个字节以 B、G、R 的顺序构成扫描线上的一个像素点。

简要程序如下：

```
……
    jpeg_create_decompress(&cinfo);
    jpeg_stdio_src(&cinfo, infile);
    jpeg_read_header(&cinfo, TRUE);
    jpeg_start_decompress(&cinfo);

while (cinfo.output_scanline < cinfo.output_height) {
    jpeg_read_scanlines(&cinfo,&buffer, 1);
    buffer++;
}
    jpeg_finish_decompress(&cinfo);
    jpeg_destroy_decompress(&cinfo);
……
```

利用 libjpeg 库解压后为 RGB24 的格式。其中 buffer 是一个数组型指针变量，数组大小是 320*3。无疑以 320*240*3Byte 的数据量发送一张图片，占大量带宽，非常不划算。

2.2.2 RGB 颜色空间到 YUV 颜色空间的转换

YUV 是被欧洲电视系统所采用的一种颜色编码方法。其中“Y”表示明亮度，也就是灰阶值，“U”和“V”表示的则是色度，是构成彩色的两个分量。

研究发现人眼对色度的敏感程度要低于对亮度的敏感程度[2]。正是如此，在我们的视频存储中，没有必要存储全部颜色信号。YUV420 相比 RGB 体现出极大的优越性：它的色度采样在每条横向扫描线上及扫描线的条数上，都只有亮度的一半。比如每张图像的像素为 320*240，那么亮度信号是 320*240，色度信号只有 160*120，需要存储空间 320*240*1.5Byte。这样 YUV420 的存储空间只需要 RGB24 存储空间的一半，但视觉效果感觉不到明显下降。

RGB 到 YUV 转换的公式如下（RGB 取值范围是 0-255）：

Y= 0.299R + 0.587G + 0.114B

U= -0.147R - 0.289G + 0.436B = 0.492(B - Y)

V= 0.615R - 0.515G - 0.100B = 0.877(R - Y)

2.3 基于 H.264 视频压缩编码

H.264 是一种高性能的视频编解码技术。目前国际上制定视频编解码技术的组织有两个，国际电联（ITU-T）制定的标准有 H.261、H.263、H.263+等；国际标准化组织（ISO）制定的标准有 MPEG-1、MPEG-2、MPEG-4 等[3]。

H.264 既保留了以往压缩技术的优点和精华又具有其他压缩技术无法比拟的许多优点：低码流、高质量的图象、较强的容错能力和网络适应性。最大的优势在于具有很高的数据压缩比，在同等图像质量的条件下，经过 H.264 压缩的视频数据，在网络传输过程中所需要的带宽更少，压缩比是 MPEG-2 的 2 倍以上，是 MPEG-4 的 1.5～2 倍[4]。特别在无线网络传输的方面表现的更加出色。

本文所用的 T264 视频编解码器是国内视频编码自由组织合力开发的 264 编解码程序，它遵循 H.264 视频编码标准，并吸收了 JM，X264 和 Xvid 三个源码的优点。在 Linux 下完成压缩编码，通过对 avc-src-0.14 版本的源码进行编译，在 T264/avc 编码器下生成一系列的*.obj 文件，应用程序可以直接使用这些目标文件提供的功能函数对 YUV 格式视频进行编码，其流程如图 4 所示。

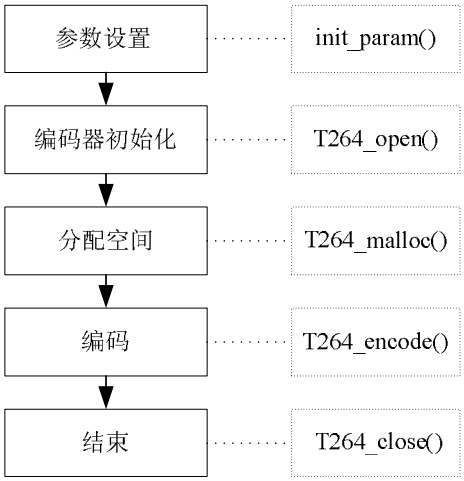

图 4 T264 压缩编码

Fig.4 compression coding using T264

1. 设置参数

修改 avc-src-0.14 源码库的 enconfig.txt 文件中各项参数，根据实际的应用需求（编码效率优先或图像质量优先）对参数进行设置，如数据帧数量、采样图像长宽、I 帧间距、参考帧数目等等。

init_param(&m_param, paramfile);

2. 用以上参数为编码器初始化

m_t264 = T264_open(&m_param);

3. 为编码分配内存空间

m_pDst = T264_malloc(m_lDstSize, CACHE_SIZE);

m_lDstSize 的大小即一帧 YUV 图像所占的内存大小，一般是高*宽*1.5。

4. 开始编码

T264_encode(m_t264, yuv_buf, m_pDst, Pixels);

yuv_buf 是指向 YUV 数据的字符型指针，length 是一帧图像的像素。

T264_encode 函数是实现压缩编码的关键函数，这个函数对一帧图像进行初始化，根据 t->slice_type 分辨是 P 帧、B 帧或 I 帧，然后分别作相应处理，然后调用 T264_encode_frame 函数对一帧图像编码，完成图像编码并返回实际编码长度值。

5. 释放资源，关闭编码器。

2.4 基于 UDP 协议的视频实时传输

TCP/IP 协议可以提供面向连接的、可靠的数据传输服务，但对于时延有高要求的多媒体数据，其重传机制和拥塞控制机制将严重影响其传输质量[5]。由于 UDP 不属于连接型协议，不提供输入包顺序及出错报文，具有资源消耗小，处理速度快的优点，所以通常运用在音频、视频和普通数据的传送，因为即使偶尔丢失一两个数据包，也不会对接收结果造成很大影响。面向无连接的 UDP 协议完全能够胜任在实时性要求大于准确性的环境下。

本设计利用 socket 进行网络编程。socket 也称为“套接字”，是网络通信的 API。它有流式套接字、数据报套接字和原始套接字三种类型。本系统采用了数据报套接字进行设计，数据报套接字使用 UDP 协议，数据通过相互独立的报文进行传输。视频服务器与客户端之间的通信过程如图 5 所示。

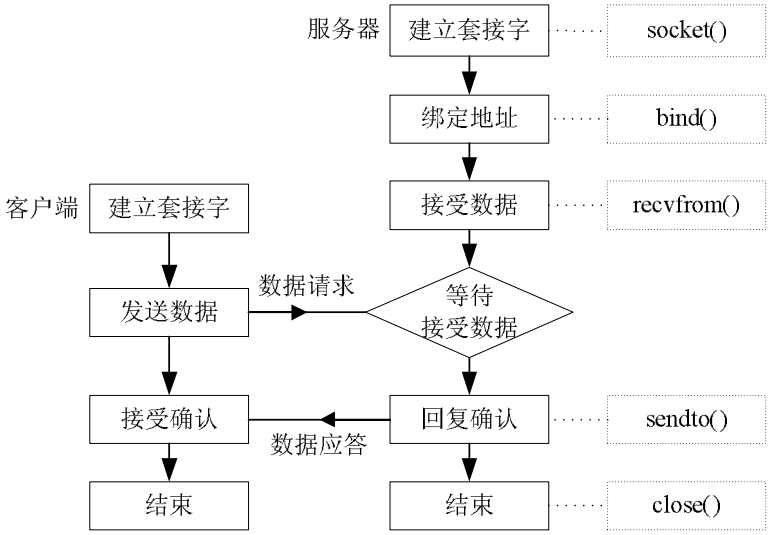

图 5 基于 UDP 协议的 socket 通信

Fig.5 the socket communication based on UDP protocol

按图示方法建立套接字，绑定本地地址，等待对方的数据请求，最后用 sendto 函数将 H.264 编码后的视频流发送给指定的客户端地址。

3 总结

本文给出了一种在嵌入式系统上基于 V4L2 的视频采集与传输设计方案，针对 zc3xx 系列的 USB 摄像头进行数据的采集与 H.264 压缩编码。经测试，摄像头采集的 JPEG 图像能够正确解码成 RGB24 格式的数据，并进一步转化成 YUV 格式给 H.264 编码器编码传输，在客户端能够流畅观看视频。

[参考文献] (References)

[1] 赵苍明，穆煜. 嵌入式 Linux 应用程序开发教程[M]. 北京：人民邮电出版社，2009.
[2] 赵勇，袁誉乐，丁锐. DAVINCI 技术原理与应用指南[M]. 南京：东南大学出版社，2008.
[3] 毕厚杰. 新一代视频压缩编码标准--H.264/AVC[M]. 北京：人民邮电出版社，2005.
[4] 姜璐. 基于 ARM 的嵌入式移动视频监控的设计[D]. 上海：华东师范大学，2008.
[5] 侯靳勇. 无线多媒体跨层协议传输系统研究与设计[D]. 长沙：国防科学技术大学，2009.
[6] 张辉. 基于 V4L2 的嵌入式视频驱动程序开发与实现[D]. 合肥：安徽大学，2010.
[7] 汪明虎，欧文盛. ARM 嵌入式 Linux 应用开发入门[M]. 北京：中国电力出版社，2008.
[8] 杨水清，张剑，施云飞，等. ARM 嵌入式 Linux 系统开发技术详解[M]. 北京：电子工业出版社，2008.