

文章编号:1007-2780(2013)06-0883-06

基于 ARM11 的全方位移动图像检测报警系统的设计与实现

李 锋, 李小彪

(江苏科技大学 电信学院, 江苏 镇江 212003, E-mail: 547369225@qq.com)

摘 要: 基于 B/S 架构设计了全方位运动图像检测报警系统。将 Linux 系统移植于 ARM11 平台上作为操作系统, 通过 USB 摄像头采集前端图像, 将采集到的图像传送到运动目标检测模块, 采用图像序列帧差法来检测是否有移动物体存在, 动态更新背景参考帧图像以防止虚警的出现; 将图像数据送到 H. 264 压缩模块进行压缩, 然后将压缩后的图像数据传输到客户端进行解码显示; 客户端利用 CGI 动态网页技术来有效控制监控端行为。实验结果表明: 本监控系统对运动目标能够准确检测, 并且图像清晰, 云台控制流畅, 能够完全满足用户日常要求, 可广泛应用于如交通、学校、银行、超市、楼宇以及一些重要赛事中。

关 键 词: ARM11; Linux; B/S 架构; CGI 技术; 图像序列帧差法; H. 264; 云台

中图分类号: O753.2 **文献标识码:** A **DOI:** 10.3788/YJYXS20132806.0883

Design and Realization of Omni-Directional Mobile Detection Alarm System Based on ARM11

LI Feng, LI Xiao-biao

(College of Electronic Information, Jiangsu University of Science and Technology,
Zhenjiang 212003, China, E-mail: 547369225@qq.com)

Abstract: Based on B/S architecture, a full range of motion image detection and alarm systems is designed. Linux system is translated into ARM11 platform as the operating system platform. Via USB camera capture front-end image, the collected image were sent to moving target detection module. Using image sequence frame difference method to detect whether a moving object exists, in order to prevent the emergence of false alarms. The background reference frame were update timely. Then the image data were compressed by H. 264 compression module, and the compressed image data is transmitted to the client for decoding display; and client end uses CGI dynamic web technology to control monitor behavior. Experimental results show that the monitoring system can accurately detect the moving target, and the image is clear, smooth PTZ control, which can fully meet the user's daily requirements, and can be widely used in transport, schools, banks, supermarkets, buildings and some important events.

Key words: ARM11; Linux; B/S architecture; CGI technology; image sequence frame difference method; H. 264; cloud terrace

1 引 言

随着计算机科学技术的日新月异, 多媒体技

术和网络通信技术得到了飞速发展, 带动了视频监控技术的飞速发展, 随着最近几年人工智能、模式识别、专家系统与计算机视觉的发展, 以及大量

收稿日期: 2013-04-11; 修订日期: 2013-10-25

作者简介: 李锋(1970—), 男, 陕西商洛人, 博士, 副教授, 研究方向: 光电子技术。

使用视频处理算法的出现,视频由此进入全数字化的智能时代,亦第三代网络视频监控。最近几年随着监控设备的成本日益下降,在国内外都掀起了智能视频监控技术的新热潮,国外等研究机构在运动物体行为的理解,基于视觉的手势识别,人体运动的人机交互技术以及导弹制导等发面都做了大量研究,目前国内虽然起步较晚,但对此也做了大量的研究如水下目标探测跟踪,人群跟踪、人和车的数量统计、遗留物体检测等方面都做了大量的研究。本文主要在视频图像的压缩算法、移动物体的识别与报警、人机交互以及云台控制等方面做了深入的分析。

2 系统总体架构与原理

本监控系统采用 B/S 架构,既视频采集监控端,与 PC 监控端。采用飞凌公司 OK6410 作为硬件核心,该硬件配备 ARM11 的高速处理器, MFC 硬件编解码模块、GPIO 接口、UART 接口、DM9000 网卡。以及外设有 PELCO-D 云台协议解码器、数字云台、中星微的 ZC301 型号的 usb 摄像头、报警器件等。采用 Linux 操作系统对现场进行全方位的监控。由视频采集、移动图像检测、图像处理压缩、图像传输、云台控制、以及客户端图像解压显示,等几部分构成。系统总体框架图如图 1 所示。

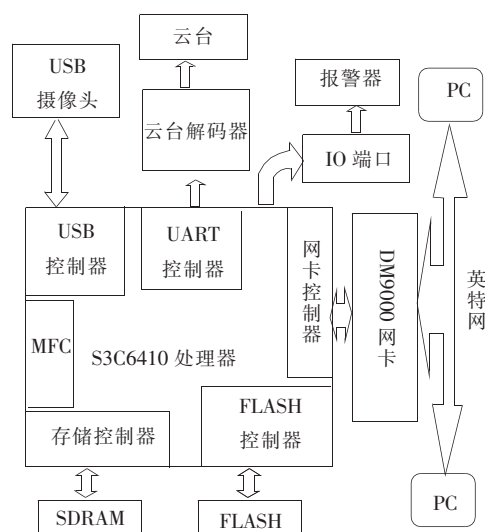


图 1 系统总体框架图

Fig. 1 Overall system architecture diagram

2.1 系统工作原理

视频采集以及移动监测模块,主要负责模拟图

像采集转化为数字图像并比较两幅图像的亮度差,分析监控范围内运动场运动点及有无超过临界范围,若无,继续处理后继的视频序列;若有,通过数字 I/O 口向报警器发出报警信号;然后将采集到的图像交给硬件的 MFC 模块对其进行 H. 264 格式的压缩;视频传输模块等待接受客户端的请求然后将压缩后的 H. 264 图像经过 internet 网络传递给客户端;客户端视频显示模块对接收到的图像进行解压缩并显示;云台控制模块接受客户端的 CGI 请求对命令进行解码控制云台的转动。

3 系统实现流程与方法

3.1 视频采集系统框架

本监控系统采用的是中星微 ZC301 的摄像头,属于 OV511 芯片类型,因此在移植 Linux 内核时要把 OV511 芯片的驱动加载上。系统中的视频子系统 Video For Linux(简称 V4L)^[1]为视频应用程序提供了一套统一的 API,视频应用程序通过标准的系统调用即可操纵各种不同的视频捕获设备。视频采集流程图如图 2 所示。

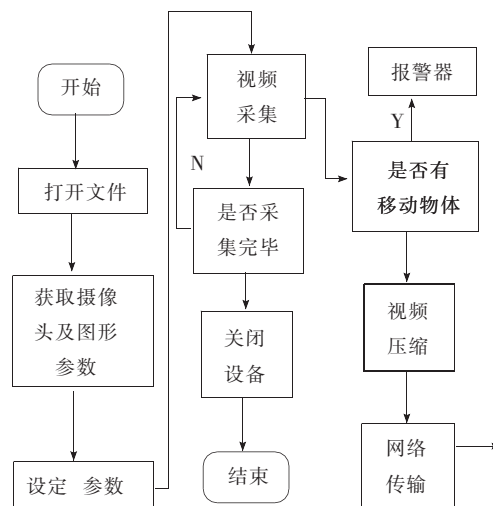


图 2 视频采集流程图

Fig. 2 Video capture process

3.2 视频采集流程

(1) 首先打开设备文件

```
fd=open("/dev/video",O_RDWR);
```

(2) 读取设备信息

```
ioctl(fd,VIDIOCGETCAP,&(vd->capability))
```

成功后可读取 `vd->capability` 各成员变量的值,得到设备的具体信息。`vd` 为视频采集结构的指针。

(3) 读取设备图像信息

```
ioctl (fd, VIDIOCGPICT, &(vd-> picture));
```

(4) 设置图形各分量的值

首先为 `vd->picture` 结构分量赋新值然后调用 `VIDIOCSPICT` 命令将新值生效

```
ioctl(fd,VIDIOCSPICT,&(vd->picture));
```

(5) 初始化 channel

```
for(i=0;i<vd->capability.channels;i++)
{vd->channel[i].channel=i;
ioctl (fd, VIDIOCSCHAN, &(vd-> channel[i]));
}
```

(6) 视频采集

视频采集有两种方法:一种是 `read` 方法:直接读取,但是每次都要系统调用,对于视频采集这种需要频繁的读取的情况,使用 `read` 方法效率低下。另一种方法是 `mmap` 方法:它是把设备文件映射到用户进程空间,这样用户就像读取一个指针指向的地址中的数据一样,也不要频繁的系统调用,大大调高了读取的速度。本采集系统使用的就是 `mmap` 方法。

`vd->map(unsignedchar *) mmap(0, vd->mbuf.size, PROT_READ | PROT_WRITE | MMAP_SHARED, fd, 0)` 将设备文件映射到用户空间,然后用视频截取命令, `ioctl(fd, VIDIOCMCAPTURE, &(vd->mmap))` 截取一帧图像,利用 `VIDIOCSYNC` 判断一帧图像是否截取完成。

`ioctl (fd, VIDIOCSYNC, &vd-> frame)`, `frame` 表示采集的是哪一帧;本采集系统使用的是双 buffer 的方式,这样处理一帧的同时可进行另一帧的采集,在内存中的地址可以通过 `vd->map+vd->mbuf.offsets.[vd->frame]` 得到。

3.3 移动图像检测报警

检测移动图像流程图如图 3 所示。

首先设定图像变化阈值 $T^{[2]}$,在不同时刻采取一幅图像,求出这些图像两两之间的差值的平均值, T 的选取介于图像的最大像素与这个平均值之间,对于要求检测比较小的物体 T 设置小一些,对于比较大的物体 T 相对取得大一些。建立参考帧图像,将当前采集到的图像与参考帧的图像作比较,若超过设定阈值 T 时发出报警信号,并将当前该帧图像作为参考帧图像,若不是继续采集下一帧图像。这样做的好处是周围的环境变

化时,避免因固定背景参考帧图像带来的副作用:当周围的背景图像变化后,即使没有移动图像出现,图像变化也会出现超过阈值的情况。经过反复设置及试验最终将 T 改为 2000。

申请一个动态内存空间作为参考帧图像的存储空间。

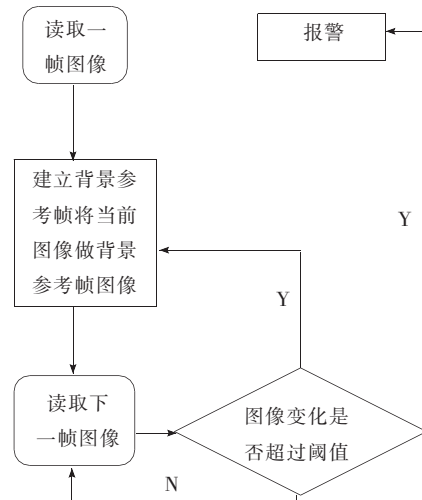


图 3 检测移动图像流程图

Fig. 3 Moving image detection process

```
ckmap = (char *) malloc (size of (char) *
HIGHT * WIDTH * BPP/8);
```

其中 `WIDTH` 为图像的宽度 `HIGHT` 为图像的高度 `BPP` 为像素值;

将图像采集缓冲区中的图像的第一帧图像复制到参考帧缓冲区:

```
p = (char *) vd-> map + vd-> mbuf. [vd->
frame];
for(i=0;i<HIGHT * WIDTH * BPP/8;i++)
{
    ckmap[i]=p[i];
}
```

由于本采集系统采用双 buffer 的结构,缓冲区中有两帧图像则将第二幅图像与背景参考帧图像进相比较,若有变化则将当前帧图像复制到背景参考帧缓冲区。

```
p = (char *) vd-> map + vd-> mbuf. [vd->
frame];
for(i=0;i<HIGHT * WIDTH * BPP/8;i++)
{
```

```
    Diff += abs(ckmap[i]-p[i]);
```

```

if(Diff>T)
{
    beep(2);
    p=(char *)vd->map+vd->mbuf.[vd->
frame];
    for(i=0;i<HIGHT*WIDTH*BPP/8;i++)
    {
        ckmap[i]=p[i];
    }
}
}

```

3.4 视频压缩

如果要将一帧采集到的未经压缩原始图像传输到客户端,假设按照 320×240 每帧的像素来计算 $320 \times 240 \times 2/1024 = 150k$ 假设每秒传输 20 P,则 20×150 约等于 3M,显然这样的传输量,对网络是极大的挑战。所以对其进行压缩很关键,本系统采用 H.264^[3] 压缩,H.264 是视频编码技术标准之一,H.264 压缩具有高压缩比能达到 102:1,低码率、容错能力强、网络适应性强、图像质量好等优点。本开发板带有 MFC 模块可以对视频数据进行 H.264 格式的硬编码,提高了编码的速度,采用 MFC 将视频采集缓冲区中的数据进行了 H.264 编码压缩。压缩流程如图 4 所示。

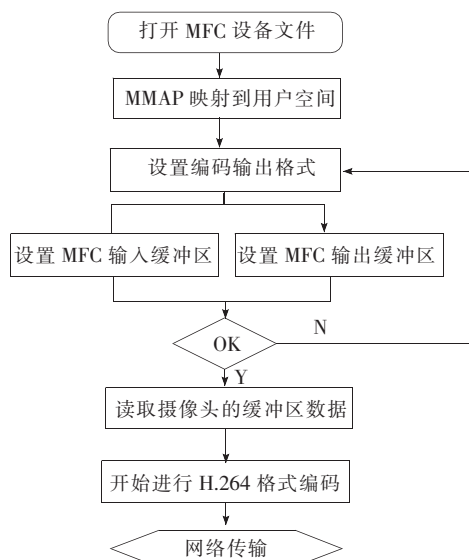


图 4 图像数据进行 H.264 格式压缩

Fig. 4 Image data compression of the h.264 format

编码流程为:

(1) 首先打开 MFC 设备文件

```

mfc_fd = open(MFC_DEV_NAME, O_RDWR |
O_NDELAY);

```

(2) 将 MFC 设备文件利用 MMAP 方法映射到用户空间

```

Addr=(char *)mmap(0,BUF_SIZE,PROT_
READPROT_WRITE,MAP_SHARED,mfc_fd,
0);

```

(3) 用 IOCTL_MFC_H264_ENC_INIT 命令设置编码参数

```

enc_init.in_width = out_width;

```

```

enc_init.in_height = out_height;

```

```

enc_init.in_frameRateRes=atoi(argv[2]);

```

```

enc_init.in_frameRateDiv = 0;

```

```

enc_init.in_bitrate= atoi(argv[3]);

```

```

enc_init.in_gopNum=atoi(argv[4]); ioctl(mfc_
fd,IOCTL_MFC_H264_ENC_INIT, &enc_init);

```

(4) 调用 ioctl 命令其中的命令参数是 IOCTL_MFC_GET_FRAM_BUF_ADDR 命令设置输入缓冲区地址

```

get_buf_addr.in_usr_data= (int)addr;

```

```

ioctl(mfc_fd,IOCTL_MFC_GET_FRAM_BUF_
ADDR, &get_buf_addr);

```

```

in_buf=(char *)get_buf_addr.out_buf_addr;

```

(5) 调用 ioctl 命令其中的命令参数是 IOCTL_MFC_GET_LINE_BUF_ADDR 命令 MFC 输出缓冲区地址

```

ioctl(mfc_fd,IOCTL_MFC_GET_LINE_BUF_
ADDR, &get_buf_addr);

```

```

out_buf=(char *)get_buf_addr.out_buf_addr;

```

(6) 循环接收摄像头数据到 MFC 的输入缓冲区

```

read(cam_fp, in_buf, (out_width * out_height *
3/2));

```

(7) 用 IOCTL_MFC_H264_ENC_EXE 命令对 MFC 输入缓冲区中的 YUV 图像进行 H.264 编码

```

ioctl(mfc_fd,IOCTL_MFC_H264_ENC_EXE,
&enc_exe);

```

(8) 将编码后的图像数据经过网络发出

```

send(net_fd, &enc_exe.out_encoded_size, 2,
0);

```

```

send(net_fd, out_buf, enc_exe.out_encoded_
size, 0);

```

3.5 云台控制

云台是安装在摄像头底座上的工作台,通过命令可以实现云台在水平和垂直方向上的转动,

云台通过云台解码器来控制其转动,本系统采用的云台,遵循 PELCO-D 云台协议。PELCO-D 云台协议^[4]如表 1 所示。

表 1 云台协议
Table 1 PTZ control protocol

Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
同步字	地址码	命令字 1	命令字 2	数据 1	数据 2	校验码

其中所有的值都用十六进制表示。同步字一般都设为 0xFF,地址码是与服务器进行通信的那台设备的逻辑地址这里设为 0x01,命令字 1 和命令字 2 用于设置摄像头的打开关闭、云台的上下左右转动、以及云台的停止、光圈的打开与关闭、焦距的拉近拉远、这里只用到了云台的上下左

右转动以及关闭。数据 1 表示镜头水平方向的速度,数值依次从停止(0x00)到高速(0x3F)和最高速(0xFF)。数据 2 表示镜头垂直方向的移动速度,其值从停止(0x00)到最高速(0x3F)。它们的命令如表 2 所示。其中所有的值都用十六进制表示。

表 2 云台命令字节
Table 2 PTZ command byte

命令	同步字	地址码	命令字 1	命令字 2	数据 1	数据 2	校验码
上	FF	01	00	08	00	FF	08
下	FF	01	00	10	00	FF	10
左	FF	01	00	04	FF	00	04
右	FF	01	00	02	FF	00	02
停止	FF	01	00	00	00	00	01

依据 PELCO-D 云台协议定义一维字符型数组 stop、left、right、up、down;
up [7] = { 0xff, 0x01, 0x00, 0x08, 0x00, 0xff, 0x08};
down [7] = { 0xff, 0x01, 0x00, 0x10, 0x00, 0xff, 0x10};
left [7] = { 0xff, 0x01, 0x00, 0x04, 0xff, 0x00, 0x04};
right [7] = { 0xff, 0x01, 0x00, 0x02, 0xff, 0x00, 0x02};
stop [7] = { 0xff, 0x01, 0x00, 0x00, 0x00, 0x00, 0x01};
并且预定义: 0 为 up、1 为 down、2 为 left、3 为 right、4 为 stop。

当客户端通过浏览器发来请求云台转动信号后,web 服务器调用相应的 moving.cgi 程序对请求进行解析,判断命令是“上”“下”“左”“右”“停止”中的哪个命令后接着做后续的工作,如下:

(1) 打开串口在 Linux 下串口文件是位于/dev 目录下,串口 1 为/dev/devttyS0
fd=open(“/dev/ttyS0”,O_RDWR);
(2) 判断是哪个命令然后往串口写入响应的命令
switch(direction)
{
 case 0:
 //当接收到 0 时执行 up 命令
 write(fd,stop,7);
 close(fd);
 Break;
 case 1:
 //当接收到 1 时执行 down 命令
 write(fd,up,7);
 close(fd);
 break;
 //以此类推

```

.....
.....
default;
break;
}

```

(3) 云台解码器^[5]通过读取串口的命令后进行解码,转化为电信号后就可以控制云台的转动了。

3.6 CGI 技术

所谓 CGI^[6]即通用网关接口,用户可以通过填写表单将一个嵌入式应用提交数据到 Web 服务器,服务器调用相应的 CGI 脚本程序来处理信息使得 Web 服务器与浏览器与之间具有交互性。如图 5 所示。

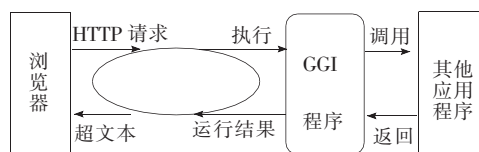


图 5 CGI 的工作流程

Fig. 5 CGI flow chart of work

本系统需要编写两个 cgi 程序分别是: video.cgi 用于开始和停止摄像头的采集; moving.cgi 用于控制云台的转动;要使 cgi 程序可以工作必须移植 boa 服务器作为 web 服务器,boa 服务器由于其非常小巧所以比较适合用于嵌入式产品,这种资源空间比较有限的嵌入式系统上;所以要先对 boa 服务器进行移植以及配置,完成后就可以接受浏览器发过来的请求调用相应的 cgi 程序来读取相应的 cgi 环境变量值,就可以对相应的请求进行操作了。

参 考 文 献:

- [1] 杨颖,陈之龙,黄杰,等. 基于 USB 摄像头的嵌入式远程视频监控系统 [J]. 微计算机信息(嵌入式与 SOC),2007,23(22):2-2.
- [2] 周瑞春. 基于 ARM+Linux 网络视频检测与报警系统的研究 [D]. 柳州:广西工学院硕士学位论文,2011.
- [3] 王洋. 基于 ARM9 的视频识别系统的研制 [D]. 武昌:湖北工业大学硕士学位论文,2011.
- [4] 张秋林,夏靖波,邱婧,等. 视频监控系统中云台控制模块的设计与实现 [J]. 电子技术应用,2011,73(9):43-46.
- [5] 谢永昌,许少凡,狄红卫. 基于 DM642 的运动目标实时跟踪监控系统的设计 [J]. 光学技术,2010,36(3):469-473.
- [6] 张军卫. 基于嵌入式 WEB 服务器的视频监控系统的的设计 [D]. 秦皇岛:燕山大学硕士学位论文,2011.
- [7] 胡峰俊,王广鹏,赵燕伟. 基于 Active X 技术构建 B/S 模式的交通视频监控系统 [J]. 交通与计算机,2005,5(23):103-104.

4 客户端测试

在客户端浏览器安装 Active X 控件^[7] ActiveX 控具有接收解码显示的功能。在浏览器地址栏输入开发板地址后,就可以出现开发板的主页了,点击开始按钮就可以浏览监控实时画面。经测试监控画面清晰,当移动目标超过给定阈值时蜂鸣器会发出报警声,云台控制流畅。丢帧率控制到 0.5% 以下,帧传输速率达到 25 P/s。如图 6 所示。

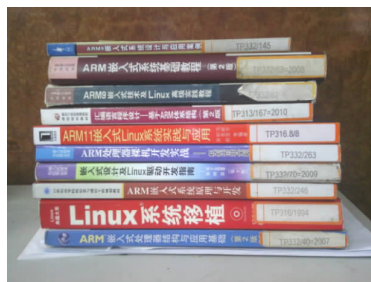


图 6 客户监控画面

Fig. 6 Client monitoring picture

5 结 论

采用基于 ARM11 处理器的飞凌公司的 OK6410 开发板上,移植 Linux 系统作为其片上操作系统,采用 B/S 架构,用新一代视频编码技术 H.264 对视频图像进行压缩大大降低网络带宽的使用,采用帧差法对移动目标做出检测并报警,并通过云台控制扩大了监控范围,缩小了监控死角。此监控系统可广泛用于家庭学校以及企事业单位,同时本文也是往智能监控方向做了一个有益的探索。