基于 OpenCV 的视频图像处理应用研究*

郭 晖[1],陈 光[1,2]

(1.东华大学 信息科学与技术学院,上海 201620;
2.东华大学 数字化纺织服装技术教育部工程研究中心,上海 201620)

摘 要:以嵌入式 ARM 为硬件平台,以 ARM-Linux 为软件平台,在 QT/Qtopia 图形用户界面下,通过调用 OpenCV 图形处理库设计摄像头应用程序,最终实现把摄像头采集到的视频流数据显示在 Qtopia 图形用户界面窗体上。介绍了 QT 编程的基本原理,阐述了 OpenCV 图像处理库的工作机制与使用方法。

关键词:嵌入式系统;ARM-Linux;QT/Qtopia;OpenCV

中图分类号:TP319 **文献标识码**:A **文章编号**:1674-7720(2010)21-0014-03

Study of video stream data processing based on OpenCV

GUO Hui[1],CHEN Guang[1,2]

(1.College of Information Science and Technology,Donghua University,Shanghai 201620,China;
2.Engineering Research Center of Digitized Textile & Fashion Technology,Ministry of Education,Donghua University,Shanghai 201620,China)

Abstract: The design is based on embedded ARM hardware platform, and its OS is the ARM-Linux QT/Qtopia. By calling OpenCV (Intel® open source computer vision library) and QT library, we design the program of the camera which can display the data of the video stream collected by the camera on Qtopia graphical user interface. In addition, the paper also describes basic principle of QT programming, and normal working mechanisms of OpenCV.

Key words: embedded system; ARM-Linux; QT/Qtopia; OpenCV

随着计算机和微电子技术的迅速发展,嵌入式 ARM 及 ARM-Linux 操作系统已广泛应用于工业控制、通信、医疗仪器等各个领域。许多公共场所和居民小区等地点都安装了视频监控系统,因而视频监控与显示终端的应用越来越广泛。于是,如何以更高的效率和更低的成本设计视频监控设备的硬件和软件就成为广大研发人员关心的问题。

本文以嵌入式 ARM 作为硬件核心,在 ARM-Linux QT/Qtopia 图形操作系统下开发摄像头应用程序,实现摄像头对视频图像数据的采集、处理与显示。为了提高应用程序的开发效率,本设计采用了 OpenCV 图形处理库。摄像头视频显示的流程为:首先 ARM-Linux 通过摄像头驱动程序控制摄像头采集视频流数据,然后利用摄像头应用程序对采集到的视频流数据进行处理,最终使经过处理的视频流数据能够在 LCD 屏上显示。采用上述平台具有以下优点:(1)ARM-Linux 与 OpenCV 库同为开源的免费软件,开发者不仅可以根据需要修改源代码来提高软件开发的灵活性,而且可以节约开发成本;(2)OpenCV 库提供了许多视频图像处理的函数,因此开发者不需要花费大量的时间自己编写,可以提高软件的开发效率;(3)OpenCV 库中大部分函数都经过汇编优化,基于 OpenCV 的程序运行起来有更高的效率。

视频监控与显示系统的硬件和软件结构如图 1 所示。

硬件由三部分组成:(1)摄像头。负责采集原始视频流数据;(2)ARM 开发板。负责处理原始的视频流数据;(3)LCD 液晶显示器。负责图像数据的显示。软件部分也由三部分组成,这三部分运行在 ARM-Liunx 操作系统下:(1) 摄像头驱动程序;(2) 摄像头应用程序。控制在 ARM 板把摄像头采集到的视频流数据读入内存中,然后对内存中的图像数据进行处理,即把原始图像数据转化为可以在 ARM-Linux QT/Qtopia 操作系统下显示的

*基金项目:上海市自然科学基金项目(08ZR1400400)

图像数据;(3)LCD 显示驱动程序。本文将着重阐述运行在 ARM-Linux QT 下的摄像头应用程序。

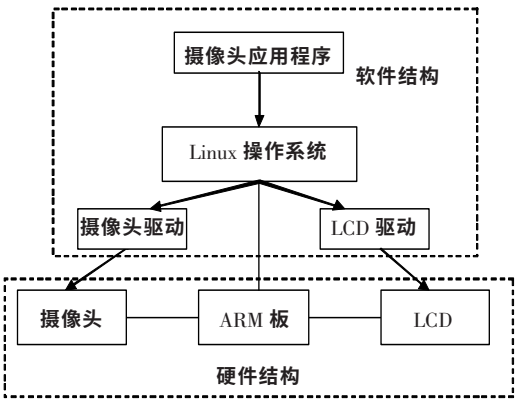

图 1 系统的硬件和软件结构

1 嵌入式系统应用程序开发方法

1.1 硬件平台

嵌入式系统开发平台由主机 PC 机和目标机 ARM 板组成。

主机 PC 要求 CPU 为 Pentium 4 或以上,拥有一个 25 针的并口、一个 9 针的 RS-232 串口和一个 20GB 的硬盘。ARM 板是由深圳市武耀博德信息技术有限公司生产的 270-S 平台。

1.2 软件开发平台

软件程序的开发是在 PC 机上完成的,PC 机上的开发环境是 Redhat Linux 9.0。Redhat 提供了许多与程序开发有关的工具,还要在 PC 机的 Linux 操作系统下安装 QT 和 OpenCV 软件工具包。

(1)QT 软件包。包括 QT/X11 2.3.2 库、QT/Embedded 2.3.2 库、Qtopia 1.7.0 库、uic 工具、qmake 工具、tmake 工具和 QT designer 工具等。

(2)OpenCV 软件包。包括 Libhighgui.so.0.9.7、Libhighgui.la、Libcxcore.so.0.9.7 和 Libcxcore.la 等主要的库。

在开发摄像头应用程序之前,要把 u-boot、ARM-Linux 操作系统和外部设备的驱动程序移植进入 ARM 目标板 270-S 中,这样主机 PC 上开发的各类应用程序软件才能在 ARM 目标板上运行。

2 摄像头应用程序的构架与关键技术

2.1 摄像头应用程序的结构

应用程序由两部分构成:

(1) 在 ARM-Linux QT/Qtopia 图形操作系统下的窗口界面设计(即人机界面的设计)。这部分是通过调用 QT/Embedded 库的各种库函数与窗口组件来完成的。

(2)对视频流数据进行处理,并把处理完成的图像数据显示在 QT/Qtopia 图形界面下。这部分设计是摄像头应用程序的核心,除了调用 QT/Embedded 库函数,还要调用 OpenCV 库函数。

摄像头应用程序结构图与库函数的调用关系如图 2 所示。

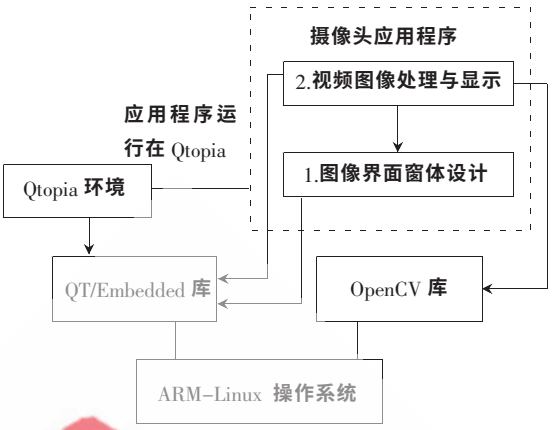

图 2 摄像头应用程序结构图

2.2 摄像头应用程序的关键技术

本设计应用程序以 OpenCV 库和 QT 库为核心,负责处理视频数据与图像显示。

2.2.1 OpenCV 简介

开放源代码的计算机图像处理库 OpenCV (Intel® Open Source Computer Vision Library) 是由一些 C 函数和 C++类所组成的库,用来实现图像处理及计算机图像算法。OpenCV 可以与英特尔公司所开发的图形处理库 IPL 兼容,所以它能够高效而充分地运行在 Intel 处理器上,主要用于对图像进行高级处理,例如特征检测与跟踪、运动分析及 3D 重建等。

2.2.2 嵌入式 QT 与 Qtopia 简介

QT 是跨平台 C++图形用户界面工具。由于 QT 采用面向对象开发,具有跨多平台、界面设计美观等特点,得到广泛应用。因为 KDE 等项目使用 QT 作为支持库,所以有许多基于 X-Windows 的 PC 机上的应用程序可以非常方便地移植到 QT 上。

Qtopia 是由 Trolltech 公司开发的基于 QT 库的消费电子设备综合应用平台。Qtopia 包含完整的应用层、灵活的用户界面、窗口操作系统、应用程序启动程序以及开发框架,并具有游戏和多媒体、工作辅助应用程序、同步框架、PIM 应用程序、Internet 应用程序等。本设计应用程序显示在 Qtopia 中。

3 QT 窗体的设计方法

在 QT 编程中,有两种设计程序窗体(即人机界面)的方法。第一种方法完全采用面向对象的 C++编程语言实现,开发者需要手工编写所有的代码;另一种是采用编写代码与 QT Designer 设计工具相结合的方法。QT Designer 工具会帮助开发者完成大部分绘制窗体的工作。本文摄像头应用程序的设计采用第二种方法。QT Designer 是 QT 系统专用的窗口界面开发工具,它不包含任何编译器,而仅仅提供一个可视化界面编辑器。QT Designer 将编辑完成的窗体界面通过 XML 保存为.ui 文

件，然后由专用的 uic 界面编译器将其转换为标准 C++ 的源文件。

4 视频图像处理与显示

视频图像处理与显示的过程如图 3 所示。

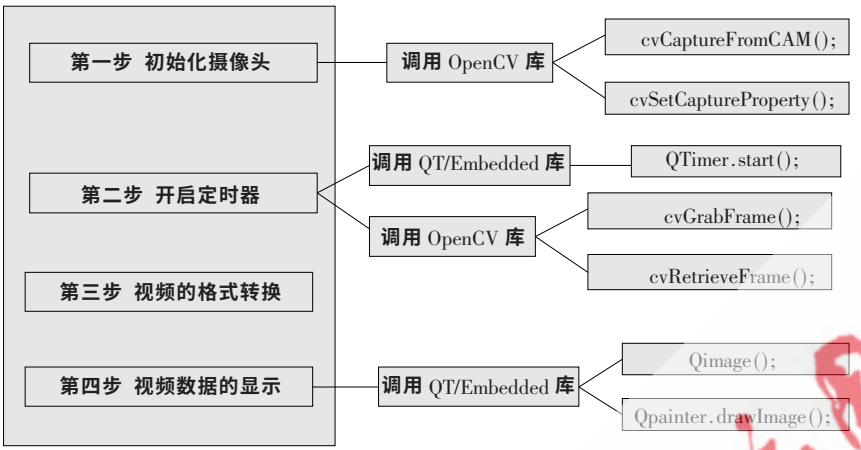

图 3 视频图像处理步骤

其过程主要由四步组成。

(1)初始化视频结构。关键代码：

```
CvCapture*capture=0;
capture=cvCaptureFromCAM(-1);
cvSetCaptureProperty
(capture,CV_CAP_PROP_FRAME_WIDTH,320);
cvSetCaptureProperty
(capture,CV_CAP_PROP_FRAME_HEIGHT,240);
```

在 OpenCV 应用程序中都要定义一个 CvCapture 类型的指针变量 capture。CvCapture 类是视频获取结构，它没有公共接口，各类图像数据存储位置的头地址都可以赋值给指针变量 capture。在 capture 指针被赋值之后，可以作为其他图像处理函数的参数使用，完成各种图像处理功能。

OpenCV 库中用 CvCapture*cvCaptureFromCAM (int index) 函数对摄像头分配视频图像数据流和初始化 CvCapture 结构。函数参数 index 为摄像头索引值。如果系统只有一个摄像头或者使用哪个摄像头都无所谓，则 index 的值为 -1。本设计开发板只连接一个摄像头，因此代码为 capture=cvCaptureFromCAM(-1)。

对视频数据结构 capture 设置参数。用到的 OpenCV 的库函数为 int cvSetCaptureProperty (CvCapture* capture, int property_id, double value)。参数 capture 指定哪个视频获取结构需要设置参数；property_id 为属性标识符，由几个固定值组成，用来决定设置哪个参数。

(2)开启定时器后抓取图像帧

关键代码：

```
QTimer CameraTimer->start(50,false);
int cvGrabFrame(capture);
IplImage*frame=cvRetrieveFrame(capture);
```

如果视频结构初始化成功，则开启由 QT 库提供的 QTimer 定时器。代码表示为：CameraTimer->start (50, false)。参数“50”表示 QT 定时器每隔 50 ms 触发一次，即发出一个内部信号调用一个槽函数，该槽函数负责从视频数据流中抓取一帧图像。

该槽函数抓取一帧图像的方法为：首先调用 OpenCV 库函数 int cvGrabFrame (CvCapture*capture)；从摄像头实时采集的视频流中快速抓取一帧图像数据，并且把这帧图像数据存入 ARM 板的缓存中，这帧图像数据对于用户是不可见的。采用这种机制，是因为 cvGrabFrame()可以把一帧图像数据以最快的速度存入缓存中[1]。

接下来，调用 OpenCV 库函数 cvRetrieveFrame ()。这个函数把刚刚通过 cvGrabFrame()抓取的一帧图像数据从内部缓存重新读取出来。具体代码为：IplImage*frame=cvRetrieveFrame(capture)。事实上在调用这个函数后，OpenCV 内部会完成多步复杂的图像处理的工作，例如解码等。

(3)视频格式的转化

关键代码：

```
for(int y=0;y<height;y++)
{  for(int x=0; x<width; x++)
   {  for(int i=0;i<3;i++)
        {*dst++=*frame->imageData++;}
      *dst++=0;}}
```

由于 cvRetrieveFrame()重新读取到的一帧图像数据是 IplImage 类型，IplImage 类型是 24 位真彩的三通道 BGR(BGR24)，而 QT 库内与图像处理与显示相关的函数只支持对 1 bit、8 bit 或者 32 bit 的位图进行处理[2]。因此为了使 IplImage 类型帧图像能够在 QT/Qtopia 图像界面中显示，又不降低视频图像质量，需要通过程序将 24 位(BRG24)帧图像转化为 32 位(BRG32)帧图像。

BGR32 每一个像素点除了拥有与 BGR24 相同的红绿蓝三种颜色，每种颜色 8 bit 外，要还在这三种颜色共 24 bit 的数据后面添加一组长度为 8 bit 的 0 数据。因此，图像格式转化的方法应该在原始的 24 位图像数据中每隔三个字节加入一个字节的 0。下列代码为 BGR24->BGR32 图像中一个点的转化程序，其中 frame->imageData 为原始图像的指针，dst 为转化后图像的指针。

```
for(int i=0;i<3;i++)
  {*dst++=*frame->imageData++;}
*dst++=0;
```

(4)将视频图像数据显示在 QT/Qtopia 图形界面

关键代码：

```
QImage image=QImage((uchar*)image32,frame->width,
      frame->height,32,NULL,0,QImage::LittleEndian);
```

```
QPainter display(picCamera);
display.drawImage(0,0,image);
```

首先调用 QImage 构造函数把上一步转换好的 32 位(BGR32)图像数据初始化为 QT 图像数据格式;然后调用 QT 的低水平绘制类 QPainter 的构造函数对主窗口的显示器组建初始化;初始化结束后将调用 QPainter 类的 drawImage 成员函数,把通过 QImage 类转换过的图像数据 image 绘制在主窗体的显示器中,代码为 QPainter.drawImage(0,0,image)。

通过以上步骤,应用程序最终将摄像头采集到的视频图像数据显示在嵌入式设备的 QT/qtopia 图形界面中。

ARM 平台的手持移动监控与显示终端设备已经广泛应用于社会的各个领域。OpenCV 图像处理库以其开源性、高效性、灵活性帮助开发者大幅度地缩减开发周期。ARM-Linux QT/Qtopia 与其他 ARM 端的图像界面操作系统相比较有免费、移植性好、内核精简、更加稳定的特点。本设计以 OpenCV 图像处理库为核心,在 ARM Linux QT/Qtopia 图形界面操作系统下实现摄像头显示的应用程序,有非常好的实用性,可以广泛应用于各类 ARM 终端设备中。

参考文献

[1] BRADSKI G, KAEHLER A. Learning openCV: computer vision with the OpenCV library. O'Reilly Media, 2008, 9.
[2] Trolltech 公司.Qtopia 和 Qt/Embedded 参考文档 [OL]. http://www.qiliang.net/qt/index.html, 2005.
[3] 韦东山.嵌入式 Linux 应用开发完全手册.北京:人民邮电出版社,2008.

(收稿日期:2010-07-06)

作者简介:
郭晖,男,1984 年生,硕士研究生,主要研究方向:嵌入式系统。