使用 OpenCV 技术实现的交通检测系统①

冯云鹏[1]，张 娜[1]，马 融[2]

[1](大连东软信息学院 计算机科学与技术系，大连 116023)
[2](吉林大学 计算机科学与技术学院，长春 130021)

摘 要：针对基于视频的交通检测提出了一套解决方案。使用了 Vistual Studio 2008 作为开发环境，并结合了微软的 DirectShow 视频处理库和 Intel 公司的 OpenCV 图像处理库进行开发。通过对三种不同场景的实验结果表明，对于车辆的识别度高达 90%以上。此系统在一般的设备下流畅运行，它能够满足实时性的要求，并且适应性也较好。

关键词：DirectShow；OpenCV；交通检测系统

Traffic Detection System Based on OpenCV Technology

FENG Yun-Peng[1], ZHANG Na[1], MA Rong[2]

[1](Department of Computer Science and Technology, Dalian Neusoft Institute of Information, Dalian 116023, China)
[2](Department of Computer Science and Technology, Jilin University, Changchun 116023, China)

Abstract: A set of solutions for traffic detection based on video is proposed. The system is developed by using Vistual Studio 2008 with Microsoft's DirectShow video processing library and Intel's OpenCV image processing library. Three experiments under different scenarios show that the accuracy of the vehicle identification can be up to 90%. The system can fluently run on common equipments, meet the real-time demand, and has good adaptability.

Keywords: DirectShow; OpenCV; traffic detection system

1 引言

随着信息技术的飞速发展，利用日趋强大的信息化手段对交通系统进行管理也成为了一种发展趋势，上世纪八十年代开始，智能交通系统（Intelligent Trans-portation System，简称 ITS）得到了快速的发展。面对现代化交通对信息需求的日益增长，ITS 不但可以在交通管理层面利用计算机强大的计算能力进行更加科学高效的交通规划和管理，并可以通过先进的自动化设备获得更丰富的交通信息。

本文实现了一个基于视频的交通检测系统，研究的范围包括视频帧的获取，图像处理，车辆的检测与跟踪，以及交通宏观信息的获取。详细探讨了车辆的检测与跟踪的技术方法，主要工作体现在如下几个方面：

(1) 采用 Directshow 技术对视频图像帧进行获取。

(2) 采用 OpenCV 库进行相关的图像处理并对感兴趣区域进行提取。

(3) 分析和探讨了基于连续图像帧之间的运动物体检测技术，比较和权衡了各种算法，在已经取得的研究成果的基础上，提出了基于检测线的车辆发现方法。

2 开发环境介绍

视频处理一般是基于视频中一帧一帧的数据，所以首要的任务就是将视频中一帧一帧的图像获得到。在这个过程中我们采用了基于 COM 组件技术的 DirectShow 开发包。DirectShow 通过对过滤器的拼接来完成视频播放和处理任务，使得开发者不必关注视频的解码方式，从而将更多的精力放在图像处

① 收稿时间:2010-09-17;收到修改稿时间:2010-11-18

理的部分。

而在图像处理过程中，我们需要一套强大而高效的 API，使我们不需要关注最底层的图像处理过程，从而将精力都放在上层的算法的研究中。而在这里我们选择了 Intel 公司的 OpenCV 图像处理函数库，下面就针对 OpenCV 和 DirectShow 开发包进行介绍。

2.1 OpenCV 的介绍

OpenCV 是 Intel 公司的一个开源的图像处理函数库。它包括 300 多个 C/C++函数的跨平台的中、高层 API，它不依赖于其它的外部库。

OpenCV 包含了计算机视觉和图像处理方面的许多通用算法，它包括以下功能：对图像数据的操作、对图像和视频的输入和输出、对矩阵和向量的操作、具有线性代数的算法程序、对各种动态数据结构进行操作、基本的数字图像处理能力、对各种结构进行分析、对摄像机定标、对运动进行分析、对目标进行识别，另外 OpenCV 还具有具有基本的 GUI 功能，还可对图像进行标注等。

在 OpenCV 库中，最常用也是最基本的一个数据结构是 Ip1Image，它在 OpenCV 中用来存储位图的数据结构，使用频率是非常高的，所以在这里有必要介绍一下它。Ip1Image 在 OpenCV 中的定义如程序 1 所示：

```
Typedef struct_Ip1Image{
   int nSize; int ID; int nChannels; int depth;
   int dataOrder; int origin; int width; int height;
   struct _Ip1Image *maskROI;
   struct _Ip1ROI *roi; void *imageId;
   struct_Ip1TileInfo *tileInfo;
   int imageSize; char *imageData;
   int widthStep; char *imageDataOrigin;
}Ip1Image;                                    (程序 1)
```

Ip1Image 结构是来源于 Intel Image Precssiong Libray，使用函数 cvCreateImage 来初始化 Ip1Image 结构，函数的定义如程序 2 所示，这个函数只是单纯分配一块没有存放内容的内存。当为其中的参数 depth 赋值为 IPL_DEPTH_8U 时，初始化的是一个单通道无符号整形图像，而赋值为 IPL_DEPTH_32F 时，初始化的是一个三通道浮点图像。

```
Ip1Image*cvCreateImage(CvSize size, int depth, int
channels);                                    (程序 2)
```

有关 OpenCV 实现其它功能的函数，文章中会贯穿的介绍。

2.2 DirectShow 的介绍

DirectShow 是微软在 Active Movie 和 Video for Windows 的基础上，推出的新一代的基于 COM 组件技术的流媒体处理的开发工具包。它为需要自定义解决方案的应用程序提供了对底层流控制结构的访问。从网络应用的角度看，DirectShow 可用于视频点播、视频会议和视频监控等领域。

当播放媒体文件时，Filter Graph Manager 首先建立一个 Filter Graph。在 Filter Graph 中，源过滤器负责读取原始的媒体数据流，变换过滤器完成对这些数据流的解码，最后由提交过滤器将解码的结果显示出来。此时的媒体数据已经转换为一帧一帧的图像，就可以方便的将它们一张张的捕捉下来了。

3 车辆检测模块的设计和实现

3.1 车辆检测模块的算法流程

使用 DirectShow，我们可以从交通视频中取得一帧一帧连续的图像，车辆的检测和发现操作就是基于对这些连续图像帧的处理和分析的。按照一般的运动检测方法和目标跟踪手段，我们将取到的原始真彩色图像帧首先做灰度化处理，然后进行平滑操作，并基于背景差分法和帧间差分法进行感兴趣区域的提取，最后得到二值化的感兴趣区域图像。根据连续帧的二值图像检测出新进入画面的车辆，之后进行基于自适应边框的精确车辆定位，从而完成车辆的检测。本章的后面部分将对其中各个步骤进行详细的介绍。

3.2 视频图像帧的预处理

3.2.1 图像灰度化处理

一般的图像处理运算都是在灰度图像上面进行的，灰度图是只保留了图像的亮度信息的图像，因为灰度图像可以在保留足够的内容信息的同时，有效的降低运算量。所以，我们首先要将原始的视频帧图像进行灰度化处理，即将真彩色图像中的每个像素点都转换成 8bit 长度的亮度值，使所有像素的灰度值都在[0,255]范围之内。

程序 3 是 OpenCV 中对图像进行颜色模型转换的函数，而图像的灰度化也属于一种颜色模型的转换：

```
void cvCvtColor(const CvArr* src, CvArr*dst, int
code);                                        (程序 3)
```

此函数将 RGB 颜色空间表示的真彩色图像进行灰度化的内部算法如公式(1)所示：

Y=0.212671*R+0.715160*G+0.072169*B (1)

我们平时比较常用的将 RGB 真彩色图像灰度化的算法如公式(2)所示：

Y=0.299*R+0.587*G+0.114*B (2)

这两个公式都是依据在 R、G、B 三个颜色通道中亮度值的贡献比例而得到的。

3.2.2 图像平滑处理

视频图像在采集的过程中会不可避免的引入噪声，噪声的来源主要有电磁转换和光电转换引入的噪声、获取图像时存在的不确定因素以及自然起伏性的噪声。

为了去除视频图像中的噪声，从而使图像中的有用信息更加清晰，我们就需要对已经进行了灰度化的图像进行平滑操作。运算要求能够有效的减少各种类型的噪声的同时，也要能够很好的保留原图像中的轮廓信息。

```
void cvSmooth(const CvArr* src,
              CvArr* dst,
              int smoothtype=CV_GAUSSIAN,
              int param1=3,
              int param2=0,
              double param3=0,
              double param4=0);
```
(程序 4)

OpenCV 中提供的图像的平滑处理的函数为 cvSmooth，它的声明如程序 4 所示，这个函数提供了很多种平滑算法供选择，包括简单不带尺度变换的模糊、简单滤波、高斯滤波、中值滤波和双向滤波等。在这里，我们采用了中值滤波来对经过了灰度化的图像进行滤波。

中值滤波的计算原理为：对于一幅图像，选取 N*N 像素大小的窗口，其中 N 为奇数，对于图像中的每一个能够放置这个窗口的位置，取出这个窗口中的所有像素点，将所有像素点的灰度值排序。由于 N 为奇数，那么在所有的 N*N 个像素点中必有一个中值，然后用此中值代替原窗口中心像素点的像素值。

3.3 背景差分法和帧间差分法的结合

在交通检测系统中，由于车辆颜色各异，有一些车辆的灰度范围与道路背景的灰度范围比较接近，在这种情况下，使用背景差分法往往很难得到清晰完整的车辆运动区域，往往会出现区域不联通和边界不清晰等问题。另一方面，由于识别的对象——车辆的颜色空间比较单一，在使用帧间差分法的时候，容易造成感兴趣区域内部的空洞，而相对的，边界信息会保存的比较完整[1]。在这样的情况之下，本系统采用了背景差分法和帧间差分法相结合的方法来进行感兴趣区域的提取，同时具备两种方法的优点，也互相弥补了对方的不足。

```
void cvAbsDiff(const CvArr* src1,
               const CvArr* src2,
               CvArr* dst);
```
(程序 5)

在 OpenCV 中实现两帧差分运算的函数为 cvAbsDiff，如程序 5 所示，它的功能就是将两幅灰度图像做差再取绝对值，它得到的结果也是灰度图像。图 1 展现的是交通视频中的某一帧图像经过一系列处理而得到的灰度图与此时动态生成的虚拟背景做差分之后的二值化图像。此时，车辆和行人已经能够分辨出来了。

类似的方法，通过函数 cvAbsDiff 将当前帧的灰度图像与上一帧的灰度图像做差分，车辆的边缘信息得到了比较好的保留，但帧间差分法也导致了车辆区域内部形成了一定程度的空洞，这是由于车辆内部的颜色比较一致而导致的。而这个缺点就可以通过融合背景差分法而得到弥补。

背景差分法与帧间差分法相结合，就是先分别通过这两种方法所得到灰度图像，并分别对它们进行二值化处理，最后对这两幅二值图像进行按对应像素或运算而得到的。在 OpenCV 中，两幅图像的对应像素进行或运算的函数如程序 6 所示，其中参数 src1 和 src2 是两幅输入图像的指针，而参数 dst 是存放结果的图像指针，mask 参数仍然表示一个蒙板，只有当 mask 中与输入图像对应像素的元素值为 1 时，才对输入图像中的对应像素进行运算，从而使两幅图片的或运算能够更加灵活的实现。而在本系统所使用的方法中，两幅图像直接无蒙板做或运算即可。图 3 显示了图 2 与图 1 进行或运算之后得到的结果。可以看出，相比图 1 和 2，图 3 的车辆内部更加饱满，边缘也更加清晰，每一辆车的白色块相对更加的完整，用肉眼也可以比较容易分辨出一辆一辆的车，这样能够为后续的车辆的识别运算提供保障。

```
void cvOr(const CvArr* src1,
          const CvArr* src2,
          CvArr* dst,
          const CvArr* mask=NULL);
```
(程序 6)


图 1 二值背景差分图像


图 2 二值帧间差分图像


图 3 或运算之后的二值图像

3.4 车辆的发现

在得到二值化的感兴趣区域之后，我们就开始使用得到的二值图像进行车辆的发现，本系统采用了一种基于检测线的方法进行车辆的发现[2,3]。该方法的基本流程如下：

1) 通过视频与真实世界的空间透视关系，能够估计出车辆的长宽的大致范围。在车辆可能进入的垂直方向设置检测线，检测线的宽度大概在估计出的车辆长度的一半[4]。如图 4 所示，车辆可能进入的位置就在视频图像的下部与上部两个地方，于是我们就设置两条宽度不同的检测线。


图 4 带有检测线的视频图像

2) 将检测线纵向切割成宽度为两个像素的小矩形条。根据每个小矩形中的感兴趣区域的多少来决定它是否是一个合格的区域，计算方法如公式(3)所示：

$$F(rec)=\begin{cases}WRITE(rec)/SUM(rec)>P & TRUE\\ WRITE(rec)/SUM(rec)<P & FALSE\end{cases} \quad (3)$$

其中，F(rec)表示参数 rec 代表的小矩形是否是一个合格的区域，WRITE(rec)表示该小矩形内感兴趣像素点的个数，而 SUM(rec)表示该小矩形内像素点的总数，P 表示一个阈值，在本系统中取 P=0.7。

3) 算法从检测线上左边第一个小矩形开始判断，遇到第一个 F(rec)的结果为 TRUE 的小矩形，记录它的位置，然后继续向右判断，直到遇到第一个 F(rec)结果为 FALSE 的小矩形时，就结束这个感兴趣区块的扩充。此时，从记录的第一个 F(rec)的结果为 TRUE 的小矩形到最后一个 F(rec)的结果为 TRUE 的小矩形就产生了一块连续的“合格”小矩形区间。

4) 判断这个区间的宽度是不是大于预设的一个阈值 W，如果大于 W，那么认为从第一个“合格”小矩形到最后一个“合格”小矩形所构成的大矩形就是一个候选的新的车辆位置，否则就认为它是噪声。

5) 重复步骤 3-4，直到判断到检测线中最后的一个小矩形为止。

6) 对于新产生的候选车辆位置，进行重复性判断，如果这个矩形与任何标示现存车辆的矩形的重合度小于一个阈值 P，那么就认为这个候选区域是一个新发现的车辆，建立这个车辆的对象，否则，则认为它是现有车辆的一部分。其中，P=D(AB)/D(A)+D(B)，D(AB)表示区域 A 与区域 B 重合的区域的大小，而 D(A)与 D(B)分别表示区域 A 与区域 B 的面积。

4 实验与结论

图 5 显示的是长春市西安大路与光明路交会的十字路口在全天 24 小时内车流量的统计，纵坐标表示的是车流量，单位是辆/小时，横坐标表示的是时间，分辨度为两个小时，从直方图中我们可以看出，在早晨八点钟和下午六点钟上下班的高峰期的时候车流量比较高，而凌晨两点到凌晨四点的车流量是全天中最小的。获得这些信息，能够有效的对交通管理工作起到帮助作用，使交通部门更合理的对人员进行安排。

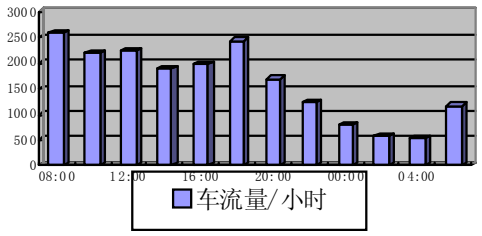

图 5 一天的车流量统计

（下转第 121 页）

代码，然后重新编译内核。步骤如下：

3.1 将设备驱动文件拷贝到/linux/driver/char 目录下

3.2 在/linux/driver/char 目录下 Makefile 中增加如下代码

obj-$(CONFIG_ XMU_GPIO) +=xgpio；

在/linux/driver/char 目录下 Kconfig 中增加如下代码：

```
Config   XMU_GPIO
    tristate “XMU_GPIO”
    depends on XILINX_DRIVERS
    select XILINX_EDK
    help
    This option enables support for Xilinx GPIO.
```

3.3 重新编译内核，进入 Linux 目录，执行以下代码

#make menuconfig

在 Character Devices-->中找到＜＞XMU_GPIO 选中为加载模块的形式：＜*＞XMU_GPIO，然后保存退出。

#make

这样得到的内核包含了用户的设备驱动程序。

4 GPIO驱动程序的测试

在应用程序中利用函数 open()系统调用 xgpio_open()函数来使能 GPIO 中断，当中断发生时，执行中断处理程序；应用程序执行 read()函数时，系统调用了 xgpio_read()函数读取 GPIO 数据寄存器的值；当应用程序执行 close()函数时，系统调用 xgpio_release()函数，屏蔽 GPIO 中断。此时，驱动程序测试结束。

5 结语

Linux2.6 内核引入的平台设备机制，使得内核对设备的管理更加简便。本文介绍了基于 PowerPC 架构的嵌入式 Linux 平台设备驱动的一般设计方法。在基于 FPGA 的嵌入式系统中，外设通过 GPIO 的 IP 核与 CPU 的互连，因此，本文介绍的设备驱动程序的设计方法，具有的一定的通用性，对底层驱动程序开发人员有较好的参考价值。此外，在 Linux 系统中，字符设备和块设备都被映射到文件系统的文件和目录，很好地体现了“一切都是文件”的思想。所有的字符设备和块设备都被统一地呈现给用户，通过文件系统的调用接口 read()、write()等函数即可访问字符设备和块设备[1]。

参考文献

1 宋宝华.Linux 设备驱动开发详解.北京:人民邮电出版社, 2008.

2 Linux 系列教材编写组.Linux 操作系统分析与实践.北京:清华大学出版社, 2008.

3 周立功,陈明计,陈渝.ARM 嵌入式 Linux 系统构建与驱动程序开发范例.北京:北京航空航天大学出版社, 2006.

4 董志国,李式巨.嵌入式 Linux 设备驱动开发.计算机工程与设计, 2006,20(27):3737-3740.

（上接第 108 页）

本系统的车流量统计部分还可以根据不同的要求统计不同的时段和不同路况下的交通流量情况，例如当给定道路的方向后，就可以统计出在某一给定时段中向东、西、南、北不同方向行驶的车流量的变化规律，这样，能够给宏观的交通调控和调度一个合理的统计依据。这样，使得下一步让系统根据道路的车流量状况自动进行交通灯的调度成为可能。

参考文献

1 张超.视频监控中的图像匹配和运动目标检测[硕士学位论文].武汉:华中科技大学,2008.34－37.

2 贺春林.一种基于视频的车辆检测算法.计算机科学,2005,32(5):243－245.

3 张玲,易卫明,何伟,郭磊民,陈丽敏.一种基于视频的车辆检测新方法.信息与电子工程,2006,4(4):264－267.

4 王圣男,郁梅,蒋刚毅.智能交通系统中基于视频图像处理的车辆检测与跟踪方法综述.计算机应用研究,2005,9:11－12.

5 Bertozzi M, Broggi A, Castelluccio S. A real time oriented system for vehicle detection. Journal of System Architecture, 1997,43(3):317－325.