基于 DirectShow的视频图像处理系统设计与实现[3]

范伊红[1),2)] 黄 涛[1)] 彭海云[3)] 吕运朋[2)]
(河南科技大学电子信息工程学院[1)] 洛阳 471003) (郑州大学物理工程学院[2)] 郑州 450052)
(周口师范学院[3)] 周口 466000)

摘 要 文章对 DirectShow技术作了概括介绍,并给出了一种通用的视频图像处理系统的 DirectShow应用软件实现过程。

关键词 DirectShow 图像处理 视频图像
中图分类号 TN911.73

Design and Implementation of Video Image Processing System based on DirectShow

Fan Yihong[1),2)] Huang Tao[1)] Peng Haiyun[3)] Lu Yunpeng[2)]
(Electronic Information Engineering College, Henan University of Science and Technology[1)], Luoyang 471003)
(School of Physical Science & Technology, Zhengzhou University[2)] Zhengzhou 450052)
(Zhoukou Normal College, Zhoukou 466000);

Abstract This paper introduces DirectShow and presents software implementation of a Video Image Processing System based on DirectShow.

Key words DirectShow, Image Processing, Video Image
Class number TN911.73

1 引言

视频图像的实时处理算法国内外作了很多的研究,当前主要的研究内容是视频图像的实时性处理及相关的图像识别问题。然而图像识别及图像的其它处理算法都必须首先从视频序列中实时的捕捉出需要的各帧图像,然后才能对图像进行运动目标检测、特征提取、模式识别等操作。

DirectShow就是一种基于 COM技术的多媒体编程接口。它给出了一种全新的多媒体数据处理模型,并封装了采集、压缩和解压缩等一系列算法,为视频监控、视频会议等多媒体应用系统的开发提供了良好的平台。本文介绍了一种基于 DirectShow的视频图像处理系统的实现。

2 DirectShow的体系结构

DirectShow本质上是基于 COM组建模型且规定了一组接口的多媒体应用框架。在 DirectShow中最基本的概念是过滤器。一个过滤器(Filter)就是一个 COM对象,具有特定的功能,过滤器对象通过其输入针和输出针之间的有序连接构成过滤器图表,过滤器图表(Filter Graph)由过滤器图表管理器(Filter Graph Manager)对象管理,用户通过过滤器图表管理器提供的接口就可以实现多媒体的回放、采集等功能。

DirectShow中的过滤器按实现的功能可分为三种基本类型,源过滤器(Source Filter)、变换过滤器(Transform Filter)及渲染过滤器(Rendering Filter)。Source Filter主要负责获取数据,数据源可以是文件、因特网计算机里的采集卡(WDM驱动的或 VFW驱动的)、数字摄像机等。Transform Filter主要负责数据格式的转换。Rendering Filter主要负责数据的最终去向,将数据送给显卡、声卡进行多媒体的演示,或者输出到文件进行存储。Filter一般有一个或多个 Pin组成,Filter之间通过 Pin相互连接,构成一条顺序的链路。Source Filter只有输出 Pin没有输入 Pin,Transform Filter既有输出 Pin也有输入 Pin,Rendering Filter只有输入 Pin没

3 收到本文时间:2006年 2月 20日

有输出 Pin。

开发 DirectShow应用程序,一般有 3个阶段。第一个阶段,创建一个 Filter Graph Manager组件。第二阶段,根据实际的应用,创建一条完整的 Filter链路,该过程可以调用 Filter Graph的接口方法手工编程实现、也可以调用 Filter Graph Manager的接口方法实现智能连接。第三个阶段,调用 Filter Graph Manager上的各个接口方法进行控制,并完成 Filter Graph Manager与应用程序的事件交互。

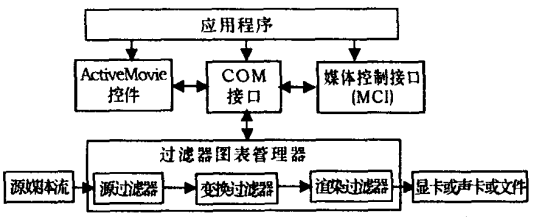

图 1　DirectShow的体系结构图

DirectShow应用程序开发过程中的常用 COM接口有:

IGraphBuilder,它是 Filter Graph Manager的接口,建立和管理一系列的过滤器,用来层层过滤和处理源媒体流;

ICaptureGraphBuilder2,它是组件 Capture Graph Builder的一个接口,用来简化 Filter Graph的创建;

IMediaControl,用来控制多媒体流在过滤器图表中的流动,如流的启动和停止;

IMediaEvent,可以捕获播放过程中发生的事件到应用程序中;

IVideoWindow,用来控制视频窗口的属性;

IMediaSeeking,媒体查找接口,用于媒体流的定位;

IBaseFilter,从 IMediaFilter接口继承,可以定义一个具体的过滤器指针,用于对多媒体数据的处理;

IPin,用来管理两个过滤器之间的针脚,从而连接过滤器。

3　系统的设计与实现

本系统采用两个线程的工作模式,主线程实现数据的采集与回放,另一个线程进行数据处理。系统使用 Sample Grabber Filter抓帧过滤器抓取视频帧,然后对每一帧应用具体的图像处理算法。抓帧过滤器是一个转换过滤器,支持 ISampleGrabber接口。它把取样无改变地向下游传输,当取样穿过该过滤器时便能得到这些帧。帧抓取过滤器没有首选的媒体类型,它在被插入之前,通过调用 ISampleGrabber::SetMediaType方法,指定参数 AM_MEDIA_TYPE设置输入针的媒体类型,媒体类型能确保过滤器管理器在过滤器图表中适当的位置插入抓帧过滤器。SampleGrabber可以有两种工作模式,缓冲区模式和回调模式,通过调用 ISampleGrabber::SetBufferSamples(TRUE/FALSE)设置。在缓冲区模式下,调用 ISampleGrabber::GetCurrentBuffer方法获得最近缓冲的帧,由于当前帧将覆盖前一帧,可以设置 ISampleGrabber::SetOneShot(TRUE),使 Sample Grabber抓取一帧后,立即停止运行。在回调模式下,要定义一个回调类,它实现 ISampleGrabberCB接口。抓帧过滤器能在每一个取样到达时调用一次回调方法。ISampleGrabberCB有两个回调方法可以实现图像处理,SampleCB和 BufferCB方法,通常,仅需实现其中的一个方法。如果你指定 BufferCB方法,如前所述,需调用 ISampleGrabber::SetBufferSamples使缓冲有效。系统的具体实现如下所示。

3.1　硬件系统

系统的硬件部分包括摄像机和视频采集卡,其主要任务是将模拟视频图像转换成适合于计算机处理的数字视频信号。可以根据系统要求选择不同的配置。

3.2　软件系统

系统的软件部分是一个基于 DirectShow的应用程序,用 VC++6.0编写。其核心代码如下:

1)初始化 com组件。

```
CoInitialize(NULL);
```

2)创建 Filter Graph Builder与 Capture Graph Builder

```
IGraphBuilder * mGraph = NULL;
ICaptureGraphBuilder2 * pBuilder = NULL;
CoCreateInstance(CLSID_FilterGraph, NULL,
CLSCTX_INPROC_SERVER,
IID_IGraphBuilder, (void * * )&mGraph);
CoCreateInstance(CLSID_CaptureGraphBuilder2
er2, NULL, CLSCTX_INPROC,
IID_ICaptureGraphBuilder2, (void * * )
&pBuilder);
pBuilder->SetFiltergraph(mGraph);
```

3)在 Filter Graph加入视频源过滤器

```
//创建一个系统枚举组件
```

```
ICreateDevEnum 3 pSysDevEnum =NULL;
CoCreateInstance (CLSID _ SystemDeviceEnum, NULL, CLSCTX_INPROC_SERVER, IID_ICreateDe2 vEnum, (void3 3 ) &pSysDevEnum);
```

//指定枚举的类型目录,获得 IEnumMoniker 接口

```
IEnumMoniker 3 pEnumCat =NULL;
pSysDevEnum - > CreateClassEnumerator (CLSID_VideoInputDeviceCategory, &pEnumCat, 0);
```

//使用 IEnumMoniker接口枚举所有的设备标识,因系统只有一个视频设备,因没用循环

```
IMoniker 3 pMoniker =NULL;
IBaseFilter 3 m_pFilter =NULL;
ULONG cFetched;
if (pEnumCat - > Next (1, &pMoniker, &cFetched) = = S_OK)
{ //创建 Filter实例
pMoniker - >BindToObject(NULL, NULL, IID_ IBaseFilter, (void3 3 ) &m_pFilter);
mGraph - >AddFilter(m_pFilter, L“Video De2 vice”);
}
pMoniker - >Release();
pEnumCat - >Release();
pSysDevEnum - >Release();
```

4)在 Filter Graph加入 AVIDecompressor Filter

```
IBaseFilter 3 pAviDecompressorF =NULL;
CoCreateInstance (CLSID _ AVIDec, NULL, CLSCTX_INPROC_SERVER,
IID _ IBaseFilter, (void 3 3 ) &pAviDecompressorF);
mGraph - >AddFilter(pAviDecompressorF, L“AVIDecompressor”);
pAviDecompressorF - >Release();
```

5)在 Filter Graph加入 Sample Grabber Filter 并设置媒体类型、缓冲模式。

```
IBaseFilter 3 pGrabberF =NULL;
CoCreateInstance (CLSID _ SampleGrabber, NULL, CLSCTX_INPROC_SERVER,
IID_IBaseFilter, (void 3 3 ) &pGrabberF);
mGraph - >AddFilter(pGrabberF, L“Sample Grabber”);
ISampleGrabber 3 pGrabber;
pGrabberF - > QueryInterface (IID _ ISam2 pleGrabber, (void 3 3 ) &pGrabber);
pGrabberF - >Release();
AM_MEDIA_TYPE mt;
ZeroMemory (&mt, sizeof (AM _ MEDIA _ TYPE));
mt majortype =MEDIATYPE_Video;
mt subtype =MEDIASUBTYPE_RGB24;
pGrabber - >SetMediaType(&mt);
pGrabber - >SetBufferSamples(true);
pGrabber - >SetOneShot(false);
```

6)连接各 Filter。

```
pBuilder - >RenderStream (&PIN_CATEGORY _CAPTURE, &MEDIATYPE_Video, m_pFilter, pAv2 iDecompressorF, pGrabberF);
pBuilder - > RenderStream (NULL, &MEDIATYPE_Video, pGrabberF, NULL, NULL);
pBuilder - >Release();
```

7)获得有关接口,

```
IMediaControl 3 mMediaControl;
IVideoWindow 3 mVideoWindow;
mGraph - >QueryInterface (IID_IMediaControl, (void 3 3 ) &mMediaControl);
mGraph - > QueryInterface (IID _ IVideoWin2 dow, (void 3 3 ) &mVideoWindow);
```

8)定义一个类 CSampleGrabberCB : public IsampleGrabberCB 在该类中重载 BufferCB 方法。在 BufferCB 中可以加入用户具体的图像处理函数。在类中可以定义一些公用属性,用于传递相关参数。如图像类型。

9)设置 Sample Grabber回调模式

```
CSampleGrabberCB CB;
```

//向类 CsampleGrabberCB 传递图像类型信息。

```
AM_MEDIA_TYPE mt;
pGrabber - >GetConnectedMediaType( &mt );
VIDEOINFOHEADER 3 vih = (VIDEOINFO2 HEADER3 ) mt pbFormat;
CB. Width = vih - >bmiHeader. biWidth;
CB. Height = vih - >bmiHeader. biHeight;
FreeMediaType( mt );
pGrabber - >SetBufferSamples( true );
pGrabber - >SetOneShot( false );
pGrabber - >SetCallback( &CB, 1 );
```

10)用接口对 Filter Graph进行控制

例如 mMediaControl - >Run(),用 mVideoW2 indow设置显示窗口 (代码略)。

11)释放还未释放的指针 (代码略)。

4 结束语

实验证明,利用 DirectShow的多媒体开发具有代码量少、通用性强、实时性好的特点。由于 DirectShow对视频捕获设备的封装,对不同的捕获设备提供相同的基本接口,使得本系统对不同硬件和各种图像处理具有很大的通用性和适应性。

参 考 文 献

[1]陆其明 DirectShow开发指南 [M]. 北京:清华大学出版社, 2003

[2]胡　涛,刘　睿,张志刚 利用 DirectShow技术实时捕捉视频流中的图像帧 [J]. 计算机应用, 2003, 6, 211～213

[3]屠添翼,石跃祥 视频监控系统中的图像采集和视频有效存储 [J]. 计算机应用研究, 2005, 8, 241～242

[4]何斌,等 VisualC + +数字图像处理 [M] 北京:人民邮电出版社, 2002

(上接第 119页)

一座典型办公大楼在一天内的客流量根据不同的时间段可以进行 4种不同交通流模式的有规律切换,即上行高峰 (UP)、下行高峰 (DP)、正常运行 (NM)、空闲期 (D)。在不同的交通流模式下,有不同的要求,为体现这一要求,给这 3个控制目标分配以不同的权重。

算法的实现过程如图 5所示,首先根据当前时间对电梯运行的交通流模式进行识别,选择与识别出来的交通流模式相对应的控制目标权重向量,采集电梯状态信息,包括呼梯者的等待时间 tw、轿厢内乘客的受影响度 d、轿厢内人数 n,判断 S_{en},将 t_w、d、n模糊化,根据一系列模糊推理规则,对候梯者满意度 S_{wt}与乘客满意度 S_{pa}进行模糊推理并反模糊化,最后将 S_{wt}、S_{pa}、S_{en}按照选择的权重向量进行加权平均,得到目标函数值 S,选择 S值最大的电梯作为最终的派梯结果。

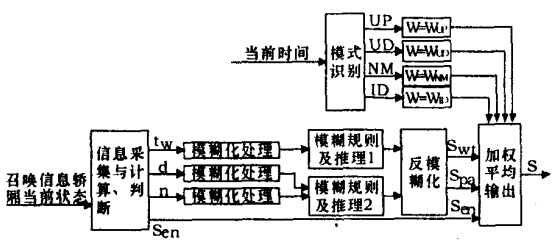

图 5　群控调度算法结构框图

5 结束语

本次设计用 CAN总线构建电梯群控网络,实现了对电梯运行状态和召唤信息进行动态调度。该系统能根据一天中不同的时间段选择相应的交通流模式,对电梯进行合理的调度,在保证候梯者与乘客都满意的前提下,有效地降低了能量的损耗,具有很高的实用价值。同时,由于 CAN总线具有可靠性高、实时性好、价格低廉、设计独特等特点,将会在电梯控制系统中得到越来越广泛的应用。

参 考 文 献

[1]袁淑娟,陈仁文. CAN总线网络节点的实现及应用 [J]. 江南大学学报　2005, 6　235～239;

[2]杨　志. 基于 CAN总线的电梯串行通讯 [J]. 微计算机信息　2005, 21(7 - 2): 56～57;

[3]余　华,邹雪城. 基于 CAN总线的电梯群控系统的通讯接口设计与实现 [J]自动化技术与应用　2005, 24(1): 34～36;

[4]宗　群,王朝阳. 电梯群控系统多目标模糊控制算法的设计 [J]. 制造业自动化　22(7): 23～25;

[5]赵亚伟,徐宝全. 基于现场总线的电梯群控模型及其算法 [J]. 控制工程　2004, 11(2): 114～117。