文章编号:2095-476X(2015)3/4-0075-04

基于FFmpeg的AVS视频播放器设计与实现

刘嘉[1], 柳英飞[2]

(1.郑州轻工业学院 软件学院, 河南 郑州 450001;
2.郑州轻工业学院 计算机与通信工程学院, 河南 郑州 450001)

摘要:针对目前数字电视监控系统的AVS视频解码方案僵化、更改不易、升级困难等问题,设计了一种基于FFmpeg编解码库和DirectX平台的AVS视频C++类播放器.该设计采用多线程数据库队列技术对数据进行共享,通过控制视频播放速度并同步到音频播放时钟上,保证了音视频播放流畅.经测试,此视频播放器支持大多数媒体格式、依赖少、易于扩展,具有一定的市场前景.

关键词:FFmpeg;AVS;多媒体;多线程
中图分类号:TP317.4 **文献标志码**:A **DOI**:10.3969/j.issn.2095-476X.2015.3/4.016

Design and implementation of the AVS video player based on FFmpeg

LIU Jia[1], LIU Ying-fei[2]

(1. *College of Software Engineering, Zhengzhou University of Light Industry, Zhengzhou* 450001, *China*;
2. *College of Computer and Communication Engineering, Zhengzhou University of Light Industry, Zhengzhou* 450001, *China*)

Abstract: To solve the problem that the AVS video decoding scheme in present digital television monitoring system was rigid which caused difficulty in change and upgrading and so on, the AVS video C++ class player was designed based on FFMpeg codec library and DirectX platform. The design adopted the multi-thread database queue technology to share data, and synchronized to the audio play clock by controlling the video playback speed to ensure the smooth play of audio and video. The test result showed that the video player was supporting most media formats, less dependent and easy to expand, and it had a certain market prospects.

Key words: FFmpeg; AVS; multimedia; multi-thread

0 引言

AVS是由我国自主制定的数字音视频编解码技术标准,是基于创新技术和部分开放技术构建的自主标准,它的压缩编码效率比MPEG-2高出2~3倍.AVS采用较少的预测模式,使方案更加简洁,芯片实现复杂度大为降低[1].此外,AVS是一套包含系统、视频、音频、媒体版权管理在内的完整标准体系,为数字音视频产业提供了更全面的解决方案[2].

目前国内外多以添加专用AVS视频解码芯片的方式来实现AVS视频的解码,但该种方式存在设计方案僵化、更改不易、升级困难等问题,而利用嵌入式硬件本身优势与特点并以软件协同完成解码的解决方案显然更具优势[3].基于此,本文拟设计一种基于多线程技术、能够嵌入系统中并支持AVS视频格式的C++类播放器,使其达到较优的运行速度,并具有设计灵活、易于扩展的特点.

收稿日期:2014-11-04
基金项目:河南省基础与前沿技术研究计划项目(142300410248)
作者简介:刘嘉(1983—),女,山西省阳城县人,郑州轻工业学院讲师,硕士,主要研究方向为计算机网络.

1 系统设计

1.1 总体设计

本设计基于多线程技术,采用 FFmpeg 编解码库设计,并使用 DirectX 进行音视频的输出;同时使用 Windows 标准程序库作为开发基础,以实现对播放器类的控制;内部由播放器类统一进行控制,播放器类中分别嵌入输入线程类、解码线程类及输出线程类,线程间同步的队列也存储在播放器类中;对外提供播放、暂停、跳转、音量、停止播放等接口.

FFmpeg 是在 Linux 下开发的可支持多种操作系统的开源项目. 要取得 Windows 下使用的动态链接库,有多种解决方案,本系统采用 Windows 下搭建 MinGW + MSys 的模拟 Linux 环境进行编译.

1.2 播放器工作流程

播放器工作流程如图 1 所示.

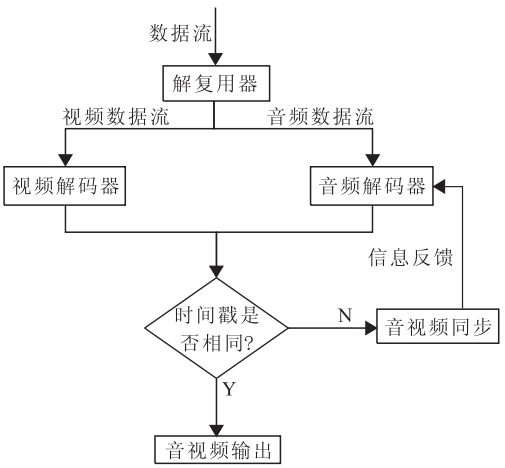

图 1 播放器工作流程图

1) FFmpeg 解复用. 一个多媒体文件既有视频又有音频,因其压缩算法各不相同,所以解码的第一步是将绑在一起的音频流和视频流分开[3]. 通过文件格式解析器(demuxer)对每个视频文件进行解析,文件解析的过程是根据已知的数据布局,在相应的文件位置找到相关数据.

2) FFmpeg 解码. 通过解复用得到的帧包含有音频帧和视频帧. FFmpeg 首先根据解码器 id 通过 avcodec_find_decoder( ) 找到适用的解码器,然后通过 avcodec_open( ) 打开解码器,以解复用得到的包为参数调用 avcodec_decode_video( ) 或 avcodec_decode_audio( ) 分别解码视频数据和音频数据.

3) DirectDraw 视频输出. 通过调用 DirectDrawCreate( ) 创建 DirectDraw 对象用于代表一个显卡,利用 QueryInterface( ) 实现接口 IDirectDraw2,调用 IDirectDraw2::CreateSurface( ) 创建用于代表物理屏幕或逻辑屏幕的 DirectDrawSurface 对象.

通过 IDirectDrawSurface2::Flip( ), IDirectDrawSurface2::BltFast( ) 等方法,切换显示页或映射部分屏幕内容.

通过函数 IDirectDraw2::CreatePalette( ) 创建 DirectDrawPalette 对象,由它代表显卡的物理调色板,每个 DirectDrawPalette 必须附着(attach)在一个 DirectDrawSurface 上,不同的 DirectDrawSurface 对象可以有不同的 DirectDrawPalette.

DirectDrawClipper 对象由函数 IDirectDraw2::CreateClipper( ) 创建,DirectDraw 用它来处理屏幕的剪贴.

4) DirectSound 音频输出. DirectSound 是 DirectXAudio 的一个较底层的部件,可实现多个声音的混合播放[4]. 通过调用 DirectXAPI 函数提供的 DirectSoundCreate8( ) 创建一个 IDirectSound8 接口对象,通过调用 SetCooperativeLevel 进一步设置设备的协调级别,如果需要设置主缓冲区的新播放格式,可通过调用 IDirectSound8 接口的 CreateSoundBuffer 函数,从声卡内存中分配缓冲区,次缓冲区用来装入实际的声音数据. DirectSound 提供了对动态流缓冲区播放的功能,通过手动设置 DirectSound 的通知机制,利用 IID_IDirectSoundNotify 的 SetNotificationPositions 实现把一部分音频样本放入,待接到 DirectSound 的通知后,将新的数据填充到缓冲区中,实现流式缓冲的播放.

5) 同步控制. 当使用 FFmpeg 解复用器得到一个包时,PTS 和 DTS 的信息也会保存在包中[4]. 因此可以通过包的 PTS 和 DTS 信息来同步音视频的播放. 本文播放器采用视频同步到音频的解决方案,即利用音频的采样率,依赖 DirectSound 的信号通知机制,当播放到指定位置的声音样本时,记录下此刻的时间,以及样本的播放时间戳,需要播放视频时,比较当前帧的视频时间戳与音频时间戳来决定视频帧的延迟时间以及是否需要丢弃.

2 系统实现

2.1 多线程数据队列的实现

本系统采用了 Windows 下的 Critical Section 实现临界区控制,并且采用信号量 Semaphore 控制队列中资源数目实现生产与消费的同步.

信号量的操作被封装在方法 PSource( ) 与

VSource( ) 中,分别对应信号量的 P,V 操作,为了使线程能够及时响应关闭及暂停操作逻辑,防止线程死锁的发生,操作为非阻塞式的实现.

初始时信号量为 0,待读线程将读得的包插入队列时,信号量增加,而解码线程从包队列中取出包时,信号量减少,信号量用于维护资源的数量信息,并且使输入与解码线程同步.

2.2　多线程的实现

播放器使用了多线程来实现各线程之间共享数据、互相协同,共同完成视频从文件中经编码复用过的数据流转为能够播放的原始音频样本与画面的工作. 一个线程相当于一个过滤器,从视频文件中读入,然后一步步将流进行转化,源源不断的送往屏幕与音响设备,实现视频的播放. 所有线程均以 CFPlayer::BeginThreads( ) 开始,以 CFPlayer::EndThreads( ) 结束,其中 Read Thread 状态图如图 2 所示.

2.3　画面绘制的实现

画面的绘制是通过 CFPlayer::DrawScreen( ) 来实现的,其传入数据为 AVFrame,在本函数中首先创建能显示 YUV420P 格式图像的 DirectDraw 的离屏表面,然后通过 DirectDraw 的 Lock 获得图像缓冲区,将图像数据以对齐方式填入缓冲区[5],其过程如图 3 所示.

2.4　音频播放的实现

音频播放采用 DirectSound 的动态缓冲区播放功能,实现大音频文件内容的播放. 设置 DirectSoundBuffer 的通知机制,当有接收到通知时,可以通过 DirectSoundBuffer 的 Lock 获得 DirectSound 需要存放下一部分声音样本的缓冲区,将解码并经转换过的声音样本存入该缓冲区中,DirectSound 就会接着进行播放. 宏观上,DirectSound 的播放是连续的. 由于解码出的音频帧样本长度和通过 Lock 获得的缓冲区长度并不对应,所以需要实现一个后备的缓冲区,用于暂存自音频帧中拷入,并准备向 DirectSound 缓冲区中拷出的音频样本,将多余的样本内容留待下一次向 DirectSound 缓冲区中拷贝时使用[6]. CFPlayer::GetFrameBuffer( ) 实现了从解码过后的音频帧队列中取得帧数据的功能. 至此,音频播放功能基本完成.

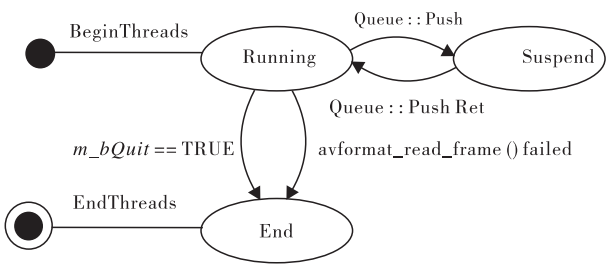

图 2　ReadThread 状态图

2.5　音视频同步的实现

DirectSound 的音频播放是按照设定好的采样率及样本字节数进行播放的[7],其播放速率比较稳定,故本播放器音视频同步的实现是以音频帧的 PTS 作为参照,按照视频同步到音频的方法实施. 在解码得到音频帧及视频帧时,帧中的额外信息记录了帧的 PTS,在音频播放线程中 DirectSound 要读取音频帧数据时,记录了当前音频帧的 PTS 信息,当需要播放视频帧时,取得其 PTS 值,并与前一帧播放时间进行差值计算,得到当前帧需要的延迟时间,并将该 PTS 值与音频帧的播放时间 PTS 时行差值计算,得到音视频播放不同步的时间差值,将前后两个差值进行比较,如果视频帧播放时间超前音频帧,则提高延时,若视频帧播放时间滞后音频帧,则减少延时,其他则按正常延时.

2.6　随机播放的实现

随机播放通过 FFmpeg 的流查找函数 avformat_seek_file( ) 实现. 同时由于解码上下文随着读文件位置的改变,需要进行刷新,通过 FFmpeg 的avcodec_flush_buffers( ) 实现清空当前解码帧所用到的一些

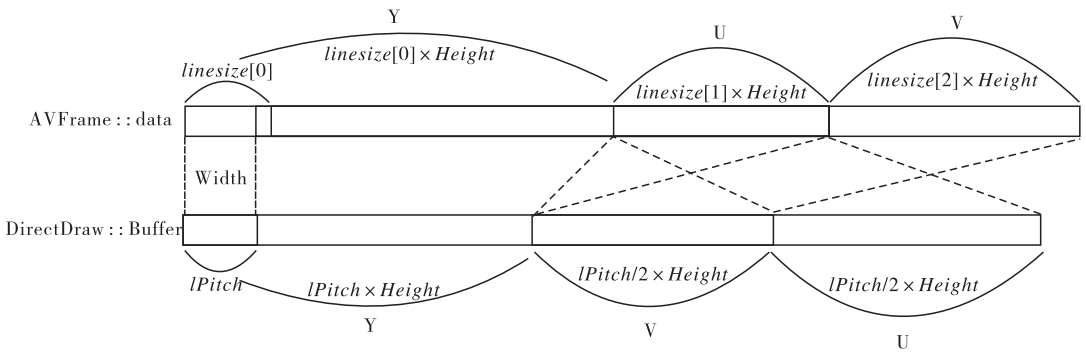

图 3　YUV420P 图像数据向 DirectDrawBuffer YV12 图像缓冲区拷贝

上下文内容. 随机播放的具体实现过程是:当调用 Seek( )时取得当前播放位置信息,通过计算得到要跳转到的位置信息,并设置 Seek 请求标记,通过方法 CFPlayer::StreamSeek( ) 实现;当读入线程检测到 Seek 请求标记时,清空当前包队列,插入定义刷新标记的定义包,以通知音视频解码器刷新解码上下文结构,并通过 avformat_seek_file 实现跳转. 随机播放功能线程间协作图如图 4 所示.

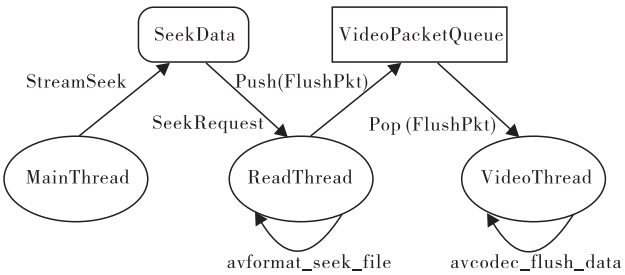

图 4 随机播放功能线程间协作图

2.7 测试评价

以 CFPlayer 为播放器对象,内部包含音频及视频包 m_qVideoPacket,m_qAudioPacket 队列,用于读线程 ReadThread 与音视频编解码线程 VideoDecodeThread,AudioDecode Thread 的数据共享与同步. 经过测试,该应用程序能够较好地实现多媒体的播放与暂停;能够实现对多媒体的快进、倒退及声音大小的调节,拥有一定的安全性、兼容性和实用性. 功能相对完善,基本支持所有格式的多媒体;界面相对简单,总体评价良好.

3 结语

为满足未来广播电视行业对 AVS 编码的音视频电视监控的需求,设计了一种便于转接到数字电视监控系统中的 AVS 视频播放器,实现了多线程数据队列、画面的最终绘制、音频的播放、音视频的同步及随机播放等功能. 因其无需硬件解码芯片,设计灵活简洁、易于护展、升级方便,具有一定的市场应用前景.

参考文献:

[1] 陈跃,王昭惠. MPEG 标准与视听技术发展趋势[J]. 郑州轻工业学院学报:自然科学版,2002,17(3):23.

[2] 刘丽霞,边金松,张琍,等. 基于 FFmpeg 解码的音视频同步实现[J]. 计算机工程与设计,2013,34(6):2087.

[3] 吴文相,邵时. 基于 ARM 的嵌入式 AVS 视频播放器的设计[J]. 计算机应用与软件,2010,27(1):247.

[4] 李晓妮. 面向 H.264 的嵌入式音视频同步编码技术研究[D]. 长春:吉林大学,2012.

[5] 刘建敏,杨斌. 嵌入式 Linux 下基于 FFmpeg 的视频硬件编解码[J]. 单片机与嵌入式系统应用,2011(6):28.

[6] Fu F F,Yi Q M,Shi M. Functional verification based on FPGA for AVS video decoder[J]. Semiconductor Photonics and Technology,2009,15(4):219.

[7] 付丽方. 基于 AVS 的播放系统的设计与实现[D]. 成都:电子科技大学,2008.