一种局域网实时视频传输的实现方法

白强 曾明强
（零八一总厂电信室 广元 628017）

摘 要：本文讲述了在Visual C++开发环境下编写网络实时视频传输的一种方法。通过用Divx编解码器来对视频帧进行编解码，阐述了从压缩、组帧、发送、接收、解码整个设计思路，同时展示了视频传输中的相关核心代码。

关键词：音频/视频 VFW Divx客户端/服务器 TCP/UDP 视频压缩比

1 引言

随着 Internet 的不断发展，实时视频传输在各行各业得到广泛的使用。大到网络视频会议小到 QQ 视频聊天，我们处处可见网络视频传输的影子。网络传输的技术越来越成熟，传输速度快，稳定性好。但是，如果传输的数据量很大，同样要引起网络工作不稳定，导致数据堵塞，产生延迟现象，视频画面迟钝。本文提出了一种基于 Visual C++开发环境，使用 VC++自带的 VFW 包进行开发，通过 Divx 编解码，进行网络实时视频传输的方法。该方法能够有效地解决由于网络的局部不稳定导致的视频图像重影、抖动、花屏，得到良好质量的视频画面。

2 几种音频/视频格式简介

常用的几种音频/视频格式有：ASF、nAVI、AVI、MPEG、DIVX、Quick Time、REAL VIDEO。其中，ASF 是一种可以直接在网上观看视频节目的压缩格式，其使用了 MPEG4 的压缩算法，压缩率和图象的质量都不错；nAVI 是由 Microsoft ASF 压缩算法的修改而来的，改善了原始的 ASF 格式的一些不足，让 NAVI 可以拥有更高的帧率 (frame rate)；AVI 兼容好、调用方便、图象质量好，但缺点是尺寸大；MPEG 是它包括了 MPEG-1，MPEG-2 和 MPEG-4。MPEG 具有很好的兼容性。其次，MPEG 能够比其他算法提供更好的压缩比，最高可达 200：1。更重要的是，MPEG 在提供高压缩比的同时，对数据的损失很小；DIVX 视频编码技术是 Microsoft mpeg4v3 修改而来，使用的 MPEG4 编码压缩技术是一种高压缩比有损视频压缩技术。用它来压缩一部容量为 5-lOG 的 DVD，保持相同的分辨率和 AC3 音轨压缩比接近 10：1。高压缩比使个人可以更轻松廉价的保存高品质影片；QuickTime 是 Apple (苹果) 公司创立的一种视频格式，是一种优良的视频编码格

式。REAL VIDEO (RA、RAM) 格式定位在视频流应用方面，但其图象质量较差。

3 视频压缩编码基本概念

视频压缩的目标是在尽可能保证视觉效果的前提下减少视频数据率。视频压缩比一般指压缩后的数据量与压缩前的数据量之比。视频压缩中常用到以下一些基本概念：

(1) 有损和无损压缩：在视频压缩中有损 (Loss) 和无损 (Lossless) 的概念与静态图像中基本类似。无损压缩也即压缩前和解压缩后的数据完全一致。有损压缩意味着解压缩后的数据与压缩前的数据不一致。在压缩的过程中要丢失一些人眼和人耳所不敏感的图像或音频信息，而且丢失的信息不可恢复。丢失的数据率与压缩比有关，压缩比越小，丢失的数据越多，解压缩后的效果一般越差。此外，某些有损压缩算法采用多次重复压缩的方式，这样还会引起额外的数据丢失。

(2) 帧内和帧间压缩：帧内 (Interframe) 压缩也称为空间压缩 (Spatial compression)。当压缩一帧图像时，仅考虑本帧的数据而不考虑相邻帧之间的冗余信息，这实际上与静态图像压缩类似。帧内压缩一般达不到很高的压缩。

采用帧间 (Interframe) 压缩是基于许多视频或动画的连续前后两帧具有很大的相关性，或者说前后两帧信息变化很小的特点。也即连续的视频其相邻帧之间具有冗余信息，根据这一特性，压缩相邻帧之间的冗余量就可以进一步提高压缩量，减小压缩比。帧问压缩也称为时间压缩 (Temporal compression)，它通过比较时间轴上不同帧之间的数据进行压缩。帧间压缩一般是无损的。

(3) 对称和不对称编码：对称性 (symmetric) 是压缩编码的一个关键特征。对称意味着压缩和解压缩占用相同的计算处理能力和时间，对称算法适合于实时压缩和传送视频，如视频会议应用就以采用对称的压缩编码算法为好。不对称或非对称意味着压缩时需要花费大量的处理能力和时间，而解压缩时则能较好地实时回放，也即以不同的速度进行压缩和解压缩。一般地说，压缩一段视频的时间比回放 (解压缩) 该视频的时间要多得多。

4 网络实时视频传输相关知识

在网络上有效的、高质量的传输视频流，需要多种技术的支持，包括视频的采集、压缩、编解码技术、应用层质量控制技术等。同样，传输速率对视频图象质量也很重要，常用的传输视频的技术中：xDSL 技术利用电话线路的双铜线实现每秒数兆比特的数据传输；利用无线技术传输视频可达到 400kbit/s；宽带 ISDN 能够以高于 150Mbit/s 的速率传送视频和其他数据；ATM 网络能够提供的速率达 25Mbit/s~1Gbit/s。；分布式排队双总线技术 (DQDB) 在双工的模式下可以提供 34~140Mbit/s 的传输速率。

网络的带宽是有限的，所以需要对传输的视频图象进行压缩。在各种音频、视频格式中，MPEG-4 被广泛的应用于网络环境下的视频传输。由于 MPEG-4 具有：可以达到很高的压缩比，灵活的编码和解码复杂性，基于对象的编码方式，允许视频、音频对象的交互，有很强的容错能力等优点。所以本文采用 Divx 编解码器对视频进行编码、压缩，实际上 Divx= (视频) MPEG-4+ (音频) MP3。

网络通信有 TCP 和 UDP 两种协议，TCP 传输控制协议位于 IP 协议层的上层，对数据

要求严格，通过提供校验、流控制以及序列信息来弥补 IP 协议可靠性差的缺陷。UDP 是一种无连接的传输服务，它不保证数据包以正确的序列被接收，而且，它不提供错误校验或序列编号。UDP 具有快速性、不精确性，相比较其比 TCP 更适合于网络环境下的视频传输。对于 Divx 编解码技术，是以帧为单位进行编解码的，分为关键帧和非关键帧。在传输过程中，由于压缩率比较高，只要一帧中错一比特位，将影响其它几百、几千的比特位，造成图像模糊、花屏，只有等到下一次关键帧的到来才有可能恢复图像清晰。为了保证传输的正确性，本文选择使用 TCP 来进行网络通信。综合使用 VFW 技术、流媒体技术，较好的解决局域网中实时视频传输容易引起的重影、抖动、花屏的问题。

5 实时视频传输实现

本文的实时视频传输在一个局域网内测试，采用服务器/客户机模式，采用 VC++编程实现。工作流程如图 1 所示。

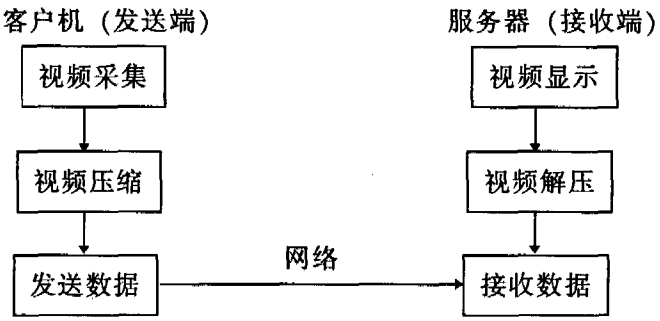

图 1 实时视频传输工作流程

这个设计思路为：在客户端，视频图象帧的采集通过 AVICap 从视频采集卡捕获视频图像，得到位图格式的视频帧，然后用 Divx 编码器对视频帧进行压缩，通过 Winsock 实现将压缩后的视频数据实时传输传输到服务器，服务器接收完数据后，把数据交给 Divx 解码器解压，最后实现视频显示。

具体实现：在 Visual C++开发环境下，采用 VFW 技术，客户端通过 capDriverConnect () 连接捕获视频图象窗口到视频驱动程序上，capPreviewRate () 设置视频图象的捕获和显示的速度，通过 capSetCallbackOnFrame () 定义回调函数。当采集卡采集到一幅图像后，系统就会自动调用回调函数，然后再回调函数中使用 ICSeqCompressFrame () 函数进行压缩。然后再通过 Winsock 将压缩后的数据发送到服务器端。服务器端接收完一帧以后，交给 ICDecompress () 解压，最后用 SetDlBitsToDevice () 将图像显示出来。

1、视频帧格式

视频采集的数据是位图型式的视频帧，Divx 编码器压缩后形成以帧为格式的 Mpeg4 流。Divx 解码器也是以帧的格式解压。所以以帧为单位发送视频数据流。帧的格式如图 2 所示。

帧开始标志	帧大小	帧编号	帧类型	帧数据

图 2 视频帧格式

每个视频帧由5个字段组成，各字段的描述如下：帧开始标志，表示一帧开始，占4个字节空间。帧大小，表示整个帧的大小，占4个字节空间。帧编号，表示帧的顺序编号，占4个字节空间。帧类型，表示此帧是否是关键帧，占1个字节空间。帧数据，存放压缩后一帧的完整数据。

2、视频数据采集

利用VFW技术捕获视频，视频数据的实时采集主要是通过调用AVICap32.dll创建AVICap窗口类，由AVICap窗口类中的消息、宏函数、结构以及回调函数来完成：捕获视频流至AVI文件 (capCaptureSequence)、捕获视频流至缓存 (capCaptureSequenceNofile)、捕获视频流至AVI文件 (capCaptureSingleFrame)、本地预览 (capPreview/capOverlay) 和捕获单帧预览 (capGrabFrame/capGrabFrameNoStop) 等。

VFW提供回调函数，允许应用程序精确控制视频流的捕获、检测错误、监控状态变化，以及在捕获两帧数据的空隙和每捕获新帧时对实时数据进行处理。AVICap在捕获视频方面具有明显优势，它能直接访问视频缓冲区，不需要生成中间文件，实时性很高，它也可将数字视频保存到事先建好的文件中。实际应用表明，通过这种方法，提高了视频采集的效果和程序运行的效率，同时也减少了对硬件的依赖性，提高了程序的兼容性和移植性。

视频数据采集初始化主要代码：

```
HWND   m_hWndCapture;          //定义采集窗口句柄
this->m_hWndCapture=::capCreateCaptureWindow (" CaptureWindow",
    WS_VISIBLEIWS_CHILD,
    0,0,320,240,               //窗口位置大小
    this->m_hWnd,              //主窗口
    1) ;                       //采集窗口 ID
    capDriverConnect (this->m_hWndCapture, 0); //连接采集驱动
    capSetVideoFormat (this->m_hWndCapture, &this->m_InInfo, sizeof (BITM
    APINFO));                                  //设置视频格式
```

3、视频帧发送

为了实时将压缩好的数据发送到接受端。在发送端创建一个线程SendMyThread用来发送数据，同时主线程仍不停的采集数据并进行压缩。流程图如图3所示。

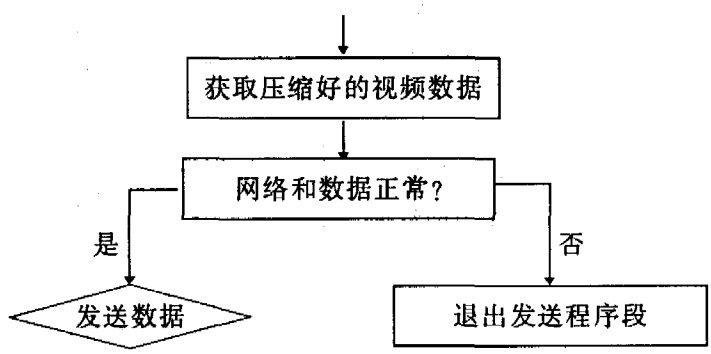

图3 发送线程工作流程

发送程序主要代码如下：

```
BOOL  bKeyFrame;
msend=true;              //准备就绪
SuspendThread (sendThread);          //挂起线程
msend=false;                    //线程正在发送数据
m_OutActSize=this->m_InInfo.bmiHeader.biSizeImage;
BYTE*Buf= (BYTE-) ICSeqCompressFrame (&m_CV,0, lpVHdr->lpData,
&bKeyFrame, (long*) &m_OutActSize); //获取视频压缩数据
if (this->m_bConnect=TRUE&&m_OutActSize<1000) //网络正常，数据正常
  {
     VIDEO_DATA  VideoData;
     memset (&VideoData,0,sizeof (VIDE_DATA));
     VideoData.bKeyFrame=bKeyFrame;
     memcpy (VideoData.Buf,Buf,m_OutActSize); //拷贝视频数据
     VideoData.nSampleNum=this->m_SampleNum;
     VideoData.nUsedSize=m_OutActSize;
     //发送视频数据到客户端
     this->m_ServerStreamSock.Send (&VideoData, sizeof (VIDEO_DATA));
     m_SampleNum+=1;
  }
  else
  {
     AfxMessageBox ("网络出错或数据出错"); //出错退出发送程序
       Break;
  }
```

4、视频帧的接收

接收端接收到数据流时候，首先从数据流中寻找帧开始标志，再取出帧的大小，然后再从接收缓冲区中读入该帧剩余的数据，再寻找下一帧的开始标志，如此循环。图4是接收端的程序流程。

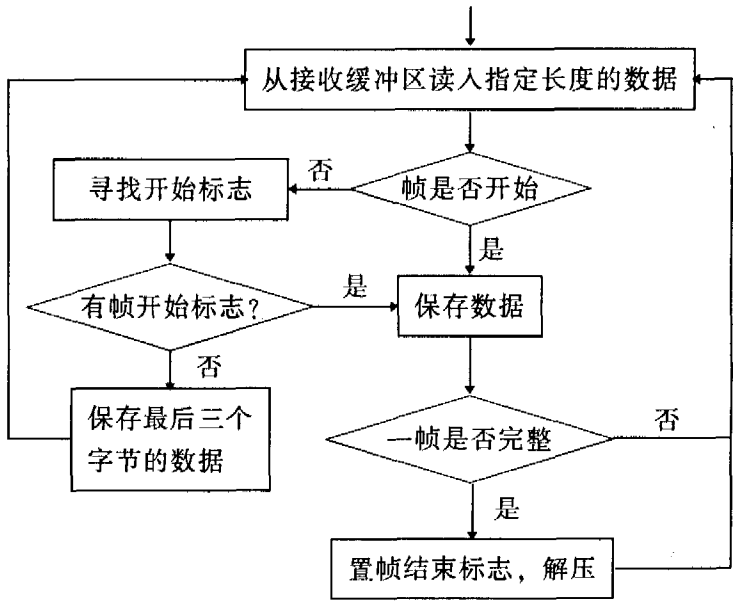

图4 接收端的工作流程

接收端创建一个线程专门用来接收数据。自定义线程名为 RecMyThread，主要程序代码如下：

```
while(temp! =SOCKET_ERROR)
{
  if(! isStart)//帧数据是否开始,true 表示开始
  {
  if(endNum>3)            //纪录当前接收未处理的数据
    endNum=0;
  temp=recv(clisock,(char*)(recBuf+endNum),1000,0);      //从缓冲区读取数据
  startPos=serchStr(temp+endNum);                        //查找帧开始标志
  if(startPos! =-1)
  {
    isStart=true;
    endNum=temp+endNum-startPos-4;
    memcpy(imageBuf,recBuf+startPos+4,endNum);//保存帧数据
  }
  else
  {
    memcpy(recBuf,recBuf+temp+endNum-3,3);//保存最后三个字节的数据
    endNum=3;
  }
 }
 else{
   if(endNum<4){//判定紧跟开始标志的数据,如果小于 4 表示不能获得帧大小
     temp=recv(clisock,(char*)(recBuf),1000,0);//读入数据
     memcpy(imageBuf+endNum,recBuf,temp);//保存数据
     endNum+=temp;
     if(endNum<4)
       continue;
     frameSize=*((int*)imageBuf);//获得帧大小
     if(frameSize<500 || frameSize>50000){//异常处理(帧大小非法)
       isStart=false;//丢弃数据重新查找帧开始标志
       endNum=0;
       continue;
     }
     frameSize-=endNum+4;
   }
   else {
```

```
        while(frameSize>0&&temp! =SOCKET_ERROR){//获得完整帧的剩余数据
        temp=recv((clisock,(char*)(imageBuf+endNum),frameSize,0);
        endNum+=temp;
        frameSize-=temp;
        }
        if(frameSize<=0){//帧结束置位,解压
          isStmt=false;
          endNum=0;
          deCompless();//调用 ICDecompress 解压
        }
      }
    }
  }
```

6 结论

通过网络传输视频的方法多种多样，采用的编解码方式各异，达到的效果各有千秋。本文介绍了一种在局域网下用 VC 编程实现视频实时传输的方法，该方法对将要进行多媒体开发的技术人员有一定的参考价值。该方法在 100M 的局域网、10M 局域网和 11M 无线局域网中进行了测试。测试时让一个目标在镜头前 (发送端) 移动，观察接收端视频的显示。测试结果：100M 的局域网传输的视频图像清晰，很流畅；10M 的局域网传输的视频图像清晰，人眼感觉流畅，偶尔出现停顿，丢帧率 1%左右；11M 无线局域网传输的视频图象清晰，但画面经常出现停顿，丢帧率 2%-3%。实验表明通过该方法实现的网络视频传输能达到良好质量的视频画面，满足视频实时传输的要求。

参考文献

1.《数字视频信息处理与传输教程》，刘富强. 机械工业出版社
2.《Visual C++高级开发范例解析》，胡小军等. 电子工业出版社
3.《多媒体数据压缩技术》，高文. 电子工业出版社

作者简介

白强，男，助理工程师，毕业于株洲工学院计算机系。
曾明强，男，工程师，毕业于信息工程学院计算机工程系。