第25卷 第8期 计 算 机 工 程 1999年8月
Vol.25 № 8 Computer Engineering August 1999

·多媒体技术及应用· 文章编号：1000－3428(1999)08－0072－03 中图分类号：TP 399

Windows环境下的实时视频捕获技术

石 峻，余松煜
(上海交通大学图象通信与信息处理研究所，上海 200030)

摘 要：结合软件实现PSTN可视电话的实际应用，针对其实现中的一个重要环节--实时视频捕获，讨论了在Windows环境下利用Visual C++所提供的Video for Windows库函数实现实时视频采集的关键技术，给出了软件实现视频流捕获的结构流程和相应的编程示例。

关键词：视频捕获；可视电话；Video for Windows

Real-time Video Sequence Capturing Scheme in Windows

Shi Jun, Yu Songyu
(Institute of Image Communication and Information Processing, Shanghai Jiaotong University, Shanghai 200030)

【Abstract:】 Combining with the application of Videophone in PSTN by using software and aiming at an important aspect, real-time video sequence capturing, in the realization of it, the key techniques in the realization of real-time video sequence capturing using the Video for Windows library provided by Visual C++ in Windows are discussed. The software realization diagram for real-time video sequence capturing is proposed and corresponding program examples are provided.

【Key words】 Video sequence capturing; Videophone; Video for Windows

随着通信技术与多媒体技术的飞速发展，越来越多的应用不仅要求传输与处理事先存储下来的图象数据，还要求应用系统能够实时获取通信对方的活动图象信息，如可视电话系统，桌面视频会议系统等的多媒体应用中。因此如何实时获取数字视频数据就成为该类应用中的重要环节和关键前提。同时随着计算机运算、处理性能的飞速发展，原来许多需要专用硬件设备才能实现的技术，现在已经完全可以利用CPU的高性能通过软件来实现。例如现在的PSTN可视电话系统已经完全可以通过PC机用软件方案实现。在其软件实现方案中，Windows环境下的实时视频捕获就成为系统实现的重要前提。Visual C++所提供的Video for Windows库函数(简称VFW)是Microsoft 于1992年推出的关于数字视频的软件包，它能够灵活地实现从模拟视频源采集数字视频信号，并将其存储到文件中或者直接对视频缓存进行处理。

1 Video for Windows函数简介

Video for Windows 函数是Windows环境下实现实时视频捕获的重要工具，主要包括vfw.h头文件和vfw32.lib函数库。由于Video for Windows函数可以方便地实现视频、音频数据流到AVI文件的存储，在Visual C++中也将Video for Windows函数称为AVIcap窗口类函数。通过使用AVIcap窗口类函数，可以在应用中方便地集成视频采集功能。AVIcap为应用提供了一个访问视频采集硬件简便的、基于消息的界面，并且能够控制视频流数据存储到磁盘的过程。

1.1 AVIcap 窗口类的基本功能

AVIcap窗口类是完成由视频捕获硬件获取数据，并按照需要的格式进行存储、转换的重要手段，它提供的主要功能包括：

1) 动态地同视频和音频输入器连接或断开；

2) 设置视频捕获速率；

3) 提供设置视频源，视频格式以及是否采用视频压缩的对话框；

4) 设置视频采集的显示模式为Overlay或者Preview模式；

5) 实时获取每一帧数字视频数据；

6) 将一视频流和音频流捕获并保存到一个AVI文件中；

7) 按用户要求捕获某一帧数字的视频数据，并将单帧图象以DIB格式的文件保存；

8) 创建、保存、或载入RGB格式下的调色板；

9) 将捕获图象和相关的调色板拷贝到剪切板；

10) 指定捕获数据的文件名，并能够将捕获的内容拷贝到另一个文件。

合理使用以上主要功能是实现实时视频应用的关键。

1.2 AVIcap窗口类函数分类

(1) 回调函数

在Windows系统中，回调函数是一类特殊的函数，功能类似于中断函数。其调用过程由系统完成，而函数的具体内容则由用户自己设定。在系统中当某一回调函数被设定后，在某一特定的条件满足时，系统自动调用的该回调函数。在AVIcap窗口类中的回调函数如capVideo-StreamCallback，capStatusCallback等函数。该类函数在使用之前需要先由用户设定函数的内容，然后将其注册到系统中，判断调用条件是否满足以及对该函数的调用则都由系统自动完成，不需要显式的调用命令。

(2) 宏

在AVIcap窗口类中，宏的使用完全类似于普通的函数，其功能等同于发送相应的窗口消息。例如

capSetCallbackOnVideoStream与显式发送 WM_CAP_SET_CALLBACK_VIDEOSTREAM 消息所完成的功能完全相同。

(3) 普通函数

AVIcap窗口类中的普通函数只有两个：

作者简介：石 峻（1974～），男，博士生，研究方向为视频编码及视频通信应用

收稿日期：1998－11－03

capCreateCaptureWindow；

capGetDriverDescription。

1.3 AVIcap 窗口类中常用的结构

在AVIcap窗口类中有4种常用的结构：

1) CAPSTATUS：定义捕获窗口的当前状态；

2) CAPDRIVERCAPS：定义捕获设备的能力，如有无视频叠加(Overlay)能力，有无控制视频源、视频格式的对话框等；

3) CAPTUREPARMS：包含控制视频流捕获过程的参数，如捕获帧频、指定键盘或鼠标键以终止捕获，捕获的时间限制等；

4) VIDEOHDR：定义了视频数据块的头信息，在编写回调函数时常用到其数据成员lpData(指向数据缓存的指针)和dwBufferLength(数据缓存的大小)。

其中前3种结构都有相应的函数来设置和获得结构包含的信息。

2 视频捕获基本结构流程

视频捕获的结构框图如图1所示。

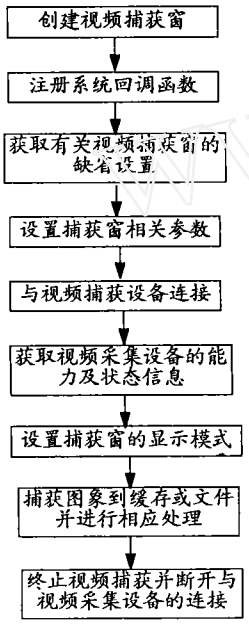

图1 视频捕获软件实现的结构框图

要进行视频捕获首先必须创建一个捕获窗，所有的捕获操作及其它设置都是以它为基础的。它是通过capCreateCaptureWindow函数创建的。

在系统回调函数中常用的为状态设置、出错处理、流捕获或帧捕获等回调函数，它们的注册是通过相应的宏完成的。如capSetCallbackOnError将系统出错处理的回调函数注册到系统中，当系统出错时自动调用进行处理。具体处理的内容由用户自己编写。

捕获窗参数设定及系统参数设定需要根据用户的具体要求决定。这里主要对视频显示的两种模式即：视频叠加模式(Overlay模式)和预览模式(Preview模式)加以详细说明。

1) Overlay模式：该模式为部分采集设备所具有的能力，可以通过CAPDRIVERCAPS结构中的fHasOverlay域来判断设备是否具备该模式。在Overlay模式下所捕获的视频数据的显示不需要经过占用系统资源，显示速度快，同时不影响系统的其它任务。该显示模式所支持的视频采集格式为YUV格式。通过capOverlay宏完成。

2) Preview模式：在该模式下显示要占用CPU资源，视频帧由系统调用GDI函数在捕获窗中显示。Previes模式的显示速度慢，该显示模式所支持的视频采集格式为RGB格式。通过capPreview宏实现。

Preview模式和Overlay模式都只是用以显示。与采集的过程并无多大关系。它主要应用于采集帧的本地回显。

参数设置完成之后，视频数据的采集是整个应用的关键，根据应用的不同可以将视频帧采集到的文件或采集的缓存直接加以处理。在可视电话的应用中需要实时的处理采集下来的帧数据，因此我们采用了将视频数据采集到缓存的方式。它的优势是速度快，实时性强。完成视频帧到文件的存储可以利用相应的宏实现，如capCaptureSequence宏将捕获帧存储到指定的文件。而实现视频帧到缓存的捕获则需要应用回调函数和相应的数据块结构VIDEOHDR。这里callback函数可以使用capSetCallbackOnFrame或capSetCallbackOnStream来注册。其不同点在于后者所能够达到的采集速率比前者要高，但需要用capCaptureSequenceNoFile加以激活。其相应的回调函数的接口及内容可以完全一致。

3 视频捕获程序示例

以下程序将视频帧捕获到指定的缓存内，以便后序处理。

(1) 定义全局变量

```
HWND ghWndCap; //捕获窗的句柄
CAPDRIVERCAPS gCapDriverCaps; //视频捕获设备的能力
CAPSTATUS gCapStatus; //捕获窗的状态
CAPTUREPARMS gCapParms;  //捕获窗的 参数
unsigned char RBuf[192*144*2];
unsigned char *ReadBuf = RBuf;//指定捕获帧的存储缓存指针
```

(2) 定义回调函数的具体内容

```
//当捕获过程发生错误时调用
LRESULT CALLBACK ErrorCallbackProc
(HWND hWnd, int nErrID, LPSTR lpErrorText)
{
  if (!ghWndCap) return FALSE;
  if (nErrID == 0) return TRUE;
lstrcpy(gachLastError,  lpErrorText);
MessageBox(hWnd, lpErrorText,
 gachAppTitle,MB_OK |
 MB_ICONEXCLAMATION) ;
  return (LRESULT) TRUE ;
}
//当捕获窗状态改变时调用
LRESULT CALLBACK StatusCallbackProc
(HWND hWnd, int nID, LPSTR lpStatusText)
{
  static int CurrentID;
  if (!ghWndCap) return FALSE;
  if (nID == IDS_CAP_END) {
if ((CurrentID ==
   IDS_CAP_STAT_VIDEOAUDIO)
     || (CurrentID ==
    IDS_CAP_STAT_VIDEOONLY))
 return(TRUE);
```

```
}
CurrentID = nID;
//更新捕获窗的状态
statusUpdateStatus(gCapStatus, lpStatusText);
return (LRESULT) TRUE ;
}
//当捕获过程中其它应用处于Yield时调用
LRESULT CALLBACK YieldCallbackProc
(HWND hWnd)
{
//在此仅作消息传递，可根据需要处理
 MSG msg;
 if (PeekMessage(&msg,
 NULL,0,0,PM_REMOVE)){
    TranslateMessage(&msg);
    DispatchMessage(&msg);
 }
 return (LRESULT) TRUE;
}
//当缓存中一帧视频数据满时调用
LRESULT CALLBACK capVideoStreamCallback
(HWND hWnd, LPVIDEOHDR lpVHdr)
{
 //将一帧的图象数据拷贝到指定缓存
 memcpy(ReadBuf,lpVHdr->lpData,
      lpVHdr->dwBytesUsed);
 return (LRESULT)TRUE;
}
```

(3) 启动视频捕获窗

视频捕获窗的启动过程主要由ViewCapture函数实现，视频数据的采集则由回调函数完成。

```
void ViewCapture(void)
{
        int nID;
        BOOL cError;
        CRect Rect;
        nID = 1;
        cError = TRUE;
// 创建视频捕获窗
        hwndParent = m_hWnd;
        GetClientRect(&Rect);
        hwndCap = capCreateCaptureWindow(
          "My capture Window",
          WS_CHILD|WS_VISIBLE,
          Rect.left,Rect.top,192,
          144,hwndParent,nID);
//由捕获窗获取视频捕获的缺省设置
capCaptureGetSetup(hwndCap,&gCapParms,sizeof(
CAPTUREPARMS));
//设置捕获窗的参数
gCapParms.dwRequestMicroSecPerFrame = 40000;
 gCapParms.fCaptureAudio= FALSE;
 gCapParms.fLimitEnabled= FALSE;
 gCapParms.fAbortLeftMouse=FALSE;
 gCapParms.fAbortRightMouse=FALSE;
 gCapParms.fYield=TRUE;
//注册回调函数
 capSetCallbackOnError(hWndCap, ErrorCallbackProc);
 capSetCallbackOnStatus(hWndCap, StatusCallbackProc);
 capSetCallbackOnYield(hWndCap, YieldCallbackProc);
 capSetCallbackOnVideoStream(hwndCap,
        capVideoStreamCallback);
//设置捕获窗参数
 capCaptureSetSetup(hwndCap,&gCapParms,
         sizeof(CAPTUREPARMS));
//建立与视频捕获设备的连接
capDriverConnect(hwndCap,NULL);
//获取捕获窗当前状态
capGetStatus(hwndCap,&gCapStatus, sizeof(gCapStatus));
//获取视频捕获设备的能力
capDriverGetCaps(hwndCap,
 &gCapDriverCaps,sizeof(CAPDRIVERCAPS));
//设置捕获窗的显示模式为Overlay模式
capOverlay(hwndCap,TRUE);
//激活capVideoStreamCallback回调函数
capCaptureSequenceNoFile(hwndCap);
}
```

(4) 关闭视频捕获窗

```
//注销回调函数
capSetCallbackOnError(hWndCap,NULL);
capSetCallbackOnStatus(hWndCap,NULL);
capSetCallbackOnYield(hWndCap,NULL);
capSetCallbackOnVideoStream(hwndCap,NULL);
//取消与视频捕获设备的连接
capDriverDisconnect(hwndCap);
```

在以上过程中，所捕获的视频帧被保存到了指定的位置，可以根据应用的需要进行处理，如进行压缩编码或传输等。

4 结束语

Visual C++所提供的Video for Windows函数是软件方式实现实时视频捕获重要工具，合理利用它所提供的函数、宏、结构以及回调函数可以方便地实现实时视频捕获，为后续的处理工作打好基础。在多媒体通信应用中尤为重要。本文所提出的应用框架适合于实时视频捕获与处理的大部分应用场合，根据不同的应用要求可以作相应的扩充和修改，作者应用该框架实现了可视电话系统视频输入端的实时视频捕获功能，并将其集成到可视电话的应用系统中，取得了满意的效果。

参考文献

1 Kruglinski D J著，王国印译. Visual C++技术内幕. 北京：清华大学出版社，1994

2 Norton P著，孙凤英译. MFC 开发Windows95/NT4应用程序. 北京：清华大学出版社，1998