2017年9月21日

主要变化：

1. 加入ReadDev 按钮，从设备中读取信息并显示在主界面中
2. 增加InitCon按钮，主要用来初始化UDP链接，创建设备实例。
3. VideoCap按钮中，会先发送开启命令给硬件，硬件才会启动数据发送。
4. StopCap按钮中，会发送停止发送命令给硬件，硬件停止数据发送。
5. 增加CloseCon按钮，关闭UDP连接，并且删除设备实例。

在UsbControlDlg.cpp中主要函数：

void \_stdcall RawCallBack(LPVOID lpParam,LPVOID lpUser)

//该函数每当6个摄像头的数据全部到来之后，会被调用6次，依次包含6个摄像头数据的内容

void \_stdcall RawCallBack(LPVOID lpParam,LPVOID lpUser)

{

//传递进来的是imgFrame型的类指针

imgFrame \*thisFrame=(imgFrame\*)lpParam;

if(thisFrame==NULL)

return;

//传递进来的是主窗口的指针

CUsbControlDlg \*pDlg=(CUsbControlDlg\*)lpUser;

//使用OpenCV来显示

cv::Mat frame(thisFrame->m\_height,thisFrame->m\_width,CV\_8UC1,thisFrame->imgBuf);

//show\_channel由窗口中的选择窗口下拉菜单控制，默认显示摄像头0的图像

if(show\_channel==thisFrame->m\_camNum)

{

//使用BMP来显示的方式，在这里没有使用，但是代码都保留了，可以参考

//memcpy(imgBuf,thisFrame->imgBuf,thisFrame->m\_height\*thisFrame->m\_width);

//BITMAPINFO\* m\_bmi;

//pDlg->BMPHeader(thisFrame->m\_width,thisFrame->m\_height,thisFrame->imgBuf,pDlg->m\_bmi);

//StretchDIBits(pDlg->m\_pDisplay->GetMemDC()->m\_hDC,0,0,g\_width,g\_height,0,0,g\_width,g\_height,thisFrame->imgBuf,pDlg->m\_bmi,DIB\_RGB\_COLORS,SRCCOPY);

//pDlg->m\_pDisplay->Display();

cv::imshow("disp",frame);

cv::waitKey(1);

//memset(imgBuf,0,g\_width\*g\_height);

}

//该标志由主窗口中的savefile勾选框控制，帧率高的时候慎点！！！

if(b\_save\_file)

{

CString strName;

CString camFolder;

camFolder.Format(L"c:\\c6UDP\\cam%d",thisFrame->m\_camNum);

if(CreateDirectory(camFolder,NULL)||ERROR\_ALREADY\_EXISTS == GetLastError())

{

int iFileIndex=1;

do

{

//如果保存所有图片的话，在C盘c6UDP文件夹内会进行保存，cam0文件夹内会保存所有0号摄像机的内容，直到cam5。 strName.Format(L"c:\\c6UDP\\cam%d\\V\_%d.bmp",thisFrame->m\_camNum,thisFrame->timestamp);

++iFileIndex;

} while (\_waccess(strName,0)==0);

CT2CA pszConvertedAnsiString (strName);

std::string cvfilename(pszConvertedAnsiString);

cv::imwrite(cvfilename,frame);

}

}

if(snap==true)

{

//cv::imwrite("snap.jpg",frame);

snap=false;

}

//h\_vw.write(frame);//保存录像

}

在imgFrame.h中

该类包含摄像头图像的宽，高，摄像头号，时间戳，图像保存的指针地址，数据长度在类初始化时创建为宽\*高

class imgFrame

{

public:

int m\_width;

int m\_height;

int m\_camNum;

long timestamp;

int status;

byte\* imgBuf;

imgFrame(int width,int height,int camNum):m\_width(width),m\_height(height),m\_camNum(camNum)

{

timestamp=0;

imgBuf=new byte[height\*width];

memset(imgBuf,0,height\*width);

}

~imgFrame(void)

{

delete imgBuf;

}

};

CCTAPI.h中主要是接口函数。

Class camInstance：它是每个相机的最顶层的包裹层，每增加一个板子增加一个camInstance类即可。

该类中的成员函数主要就是原来的单个相机的使用函数

Dll文件输出的函数转变为可以对多个相机进行操作的函数，大部分函数增加了一个camNum参数，表示对第几个相机进行操作。

CCT\_API int addInstance(LPVOID \*lpUser,LPMV\_CALLBACK2 CallBackFunc);

该函数用来增加camInstance，每个相机实例都一定需要最终的显示等功能，因此在这里将回调函数挂接上。

传入：

LPVOID \*lpUser传入的为主窗口指针，最终传递到回调函数中

LPMV\_CALLBACK2 CallBackFunc，传入回调函数，就是UsbControlDlg.cpp中的RawCallBack函数。

返回值：

相机板编号。相机板的编号将传入到后续函数中。

CCT\_API int initCCTAPI(int CamNum);

该函数主要用于对指定的相机板初始化网络和初始化内部默认参数

Int CamNum 相机板的编号

CCT\_API int startCap(int height,int width,int camNum=1);

开始采集

Int Height,图像高度

Int width，图像宽度

camNum相机板编号

CCT\_API int stopCap(int camNum);

停止采集

CCT\_API int setMirrorType(DataProcessType mirrortype);

通过软件进行镜像。由于摄像头自身已经带有镜像功能，因此该函数不再使用。

CCT\_API int getFrameCnt(int camNum=1);

帧数统计，该值会在主窗口中由1s定时器调用然后清0

CCT\_API int getDataCnt(int camNum=1);

数据量统计，该值会在主窗口中由1s定时器调用然后清0

CCT\_API int sendProp(clientPropStruct prop,int camNum=1,int s=0);

该函数的作用是发送主板的参数，包括IP地址，MAC地址，包等待时间（该值一般不要更改），包长度（一般不要更改），这些值统一写入到结构体clientPropStruct之后再统一由本函数发送。具体使用参见void CUsbControlDlg::OnBnClickedButtonUdprop()。s=0的时候发送出去，s=1的时候更新本机参数但不发送。由于这些值是保存到EEPROM内的，写入次数有限，因此测试的时候一般不要更改。

CCT\_API int sendOrder(camPropStruct camprop,int camNum=1,int s=0);

该函数的作用是发送摄像头的参数，包括图像宽，高，曝光值，增益值，镜像，这些值统一写入到结构体camPropStruct之后再统一由本函数发送。具体使用参见void CUsbControlDlg::OnBnClickedButtonCamOrder()。s=0的时候通过网口发送出去，s=1的时候更新本机参数但不发送

CCT\_API int sendSoftTrig(int camNum=1,int s=0);

软触发功能。S=0时关闭软触发，s=1时开启软触发，s=2时发送软触发一次。软触发功能开启时硬件实际运行在自动触发模式，当软件收到s=2触发一次的命令时，API向回调函数推送一次（含6张图）最新的图像数据。

全局变量：

int board1=-1;//board id

int board2=-1;

void CUsbControlDlg::OnBnClickedBtnVideocapture()

{

首先增加相机板，这里第二个相机的回调函数是RawCallBack2，该回调函数内部与之前基本相同，只是使用”disp2”窗口进行显示，

board1=addInstance((LPVOID\*)this,RawCallBack);

board2=addInstance((LPVOID\*)this,RawCallBack2);

CString s\_temp;

CStringA s\_tempA;

m\_eMAC.GetWindowTextW(s\_temp);

s\_tempA=s\_temp;

const size\_t newsizea = (s\_tempA.GetLength()+1);

char \* pc=new char [newsizea];

strcpy\_s(pc,newsizea,s\_tempA);

unsigned char byAddress[6] = {'\0'};

prop.MACaddr=ConverMacAddressStringIntoByte(pc,byAddress);

DWORD addr;

m\_eCAMIP.GetAddress(addr);

prop.camIP=addr;

m\_ePCIP.GetAddress(addr);

prop.pcIP=addr;

m\_ePACK.GetWindowTextW(s\_temp);

int tempint=\_tstoi(s\_temp);

if(tempint!=0)

prop.packetSize=tempint;

m\_eINTV.GetWindowTextW(s\_temp);

tempint=\_tstoi(s\_temp);

if(tempint!=0)

prop.interval\_time=tempint;

IP地址等属性配置完毕后，即可初始化board1

sendProp(prop,board1,1);

initCCTAPI(board1);

prop.camIP=htonl(inet\_addr("192.168.2.8"));

prop.pcIP=htonl(inet\_addr("192.168.2.9"));

board2的参数与board1基本相同，只是IP地址不同，修改后初始化board2

sendProp(prop,board2,1);

initCCTAPI(board2);

CUsbControlDlg \*temp=this;

Board1进行采集

if(startCap(g\_height,g\_width,board1)<0)

{

SetDlgItemText(IDC\_STATIC\_TEXT,L"设备打开失败！");

return;

}

else

{

SetDlgItemText(IDC\_STATIC\_TEXT,L"采集中...");

CheckRadioButton(IDC\_RADIO\_NORMAL,IDC\_RADIO\_XYMIRROR,IDC\_RADIO\_NORMAL);

SetTimer(1,1000,NULL);

SetTimer(2,50,NULL);

}

Board2进行采集

if(startCap(g\_height,g\_width,board2)<0)

{

SetDlgItemText(IDC\_STATIC\_TEXT,L"设备打开失败！");

return;

}

}