X86 开放平台 SDK
开发指南(V1.0.0)

编　　制	工业智能相机团队
审　　核	
批　　准	

目录

1. 简介

1.1 硬件架构及平台

X86 智能相机 SDK，是 Windows 的 32 位的 SDK，（支持 32/64 位操作系统或者程序调用），支持 XP/Win7/Win8/Win10 操作系统；

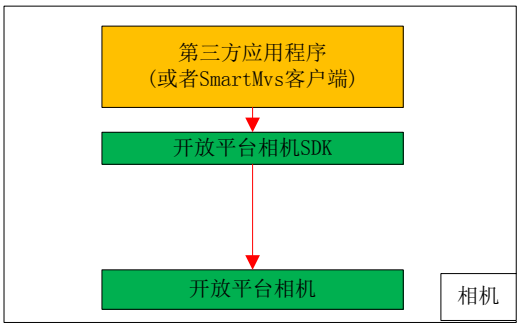
用户可以不用关心相机的内部实现，通过调用智能相机的 SDK 直接对智能相机进行各种配置，取流等操作，然后可以用取到的图像进行各种上层应用.

1.2 功能

开放平台 SDK 支持对开放平台相机的帧率，曝光，增益等各种图像参数的设置，支持对硬件 IO 触发模式的设置，用户可以根据需求，设置触发模式或者非触发模式来获取图像数据.

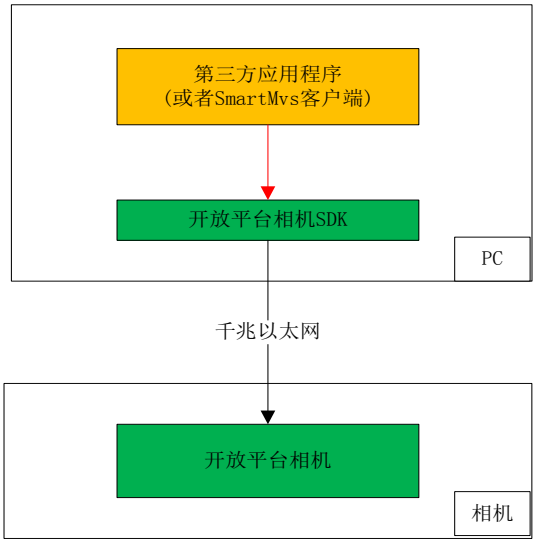
1.3 开放平台 SDK 的使用场景

场景 1：用户可以直接在相机内部进行开发，在相机内部直接调用 SDK，用来进行各种参数设置，模式设置，获取图像.具体见下图：



开放平台 SDK 使用场景 1

场景 2：用户可以在相机外面的第三方设备上调用 SDK 进行开发，第三方设备和开放平台相机通过千兆以太网进行连接.



开放平台 SDK 使用场景 2

2. 基本环境配置

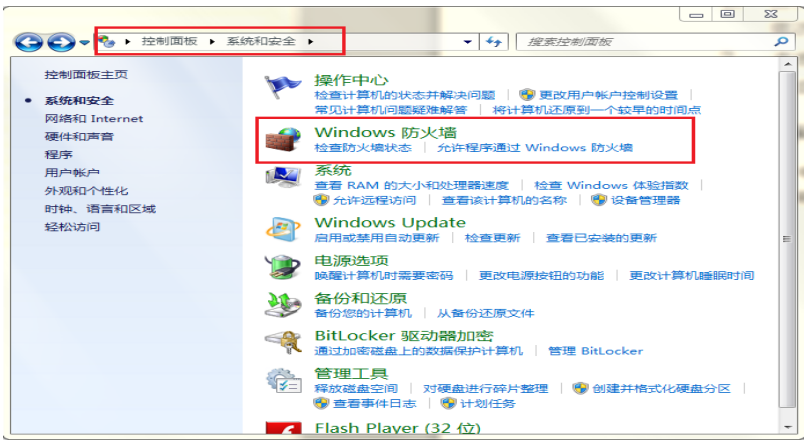
2.1 防火墙设置

说明：为保证客户端运行及图像传输稳定性，在使用客户端软件前，请关闭系统防火墙系统的防火墙并关闭杀毒软件.

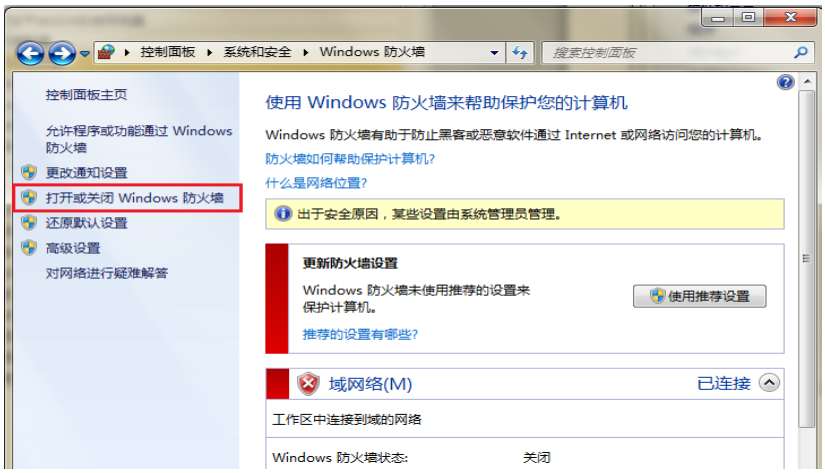
关闭防火墙步骤:

1. 打开系统防火墙：

   依次点击 开始》 控制面板》 系统和安全》 防火墙



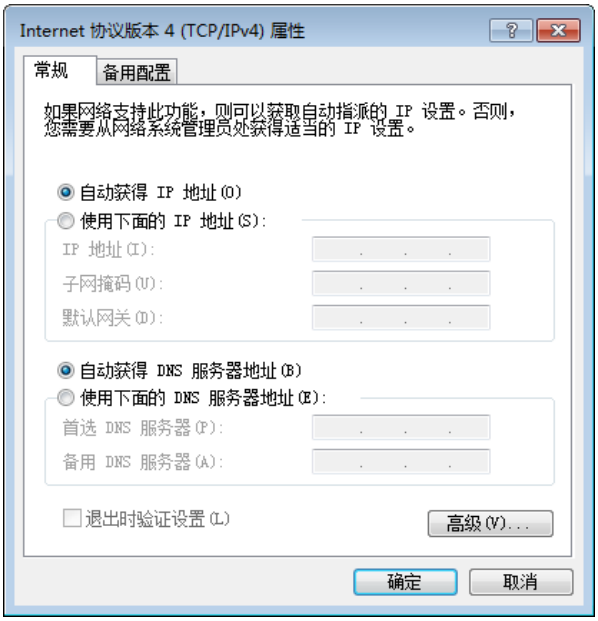
2. 点击左侧打开和关闭防火墙。



3. 在自定义界面，选择 关闭 Windows 防火墙（不推荐）。

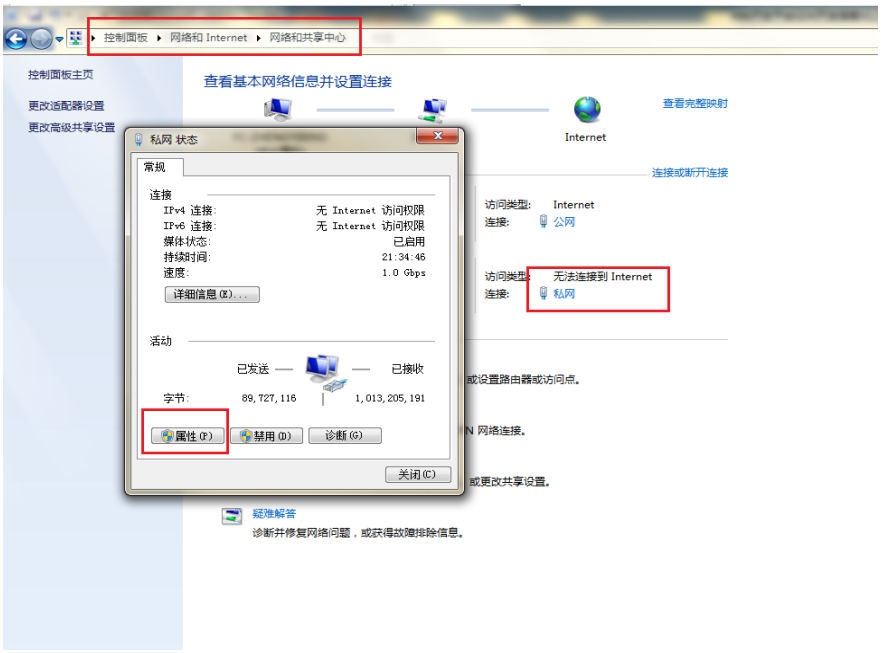
2.2 网络设置

依次打开电脑上的控制面板》网络和 Internet》网络和共享中心》更改适配器配置，选择对应的网卡，将网卡配置成自动获得 IP 地址或手动分配与相机同一网段地址。

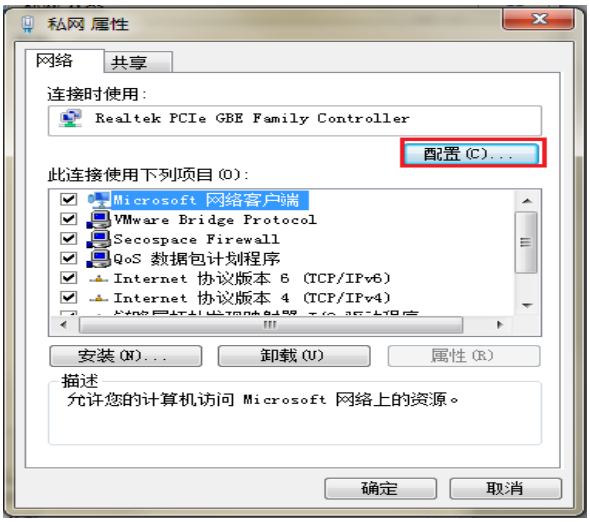


开启网卡巨帧，设置方法如下：

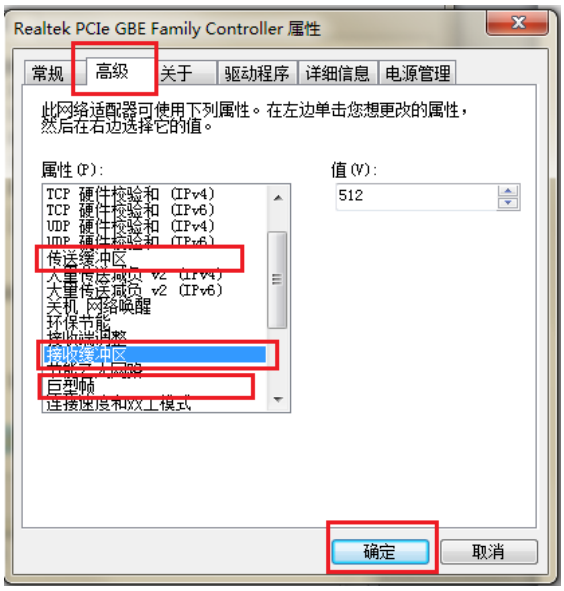
1. 依次点击控制面板→网络和 Internet→网络和共享中心→网络适配器，选中对应的网卡，点击属性。



2. 打开属性中配置



3. 选择高级菜单，本地网卡大型数据帧设置为最大值 9014 字节，传输缓冲区和接收缓冲区均设置为 2048，中断节流率设置为极值。[最大值视具体网卡情况不同，设置为最大值即可]，具体设置下图所示



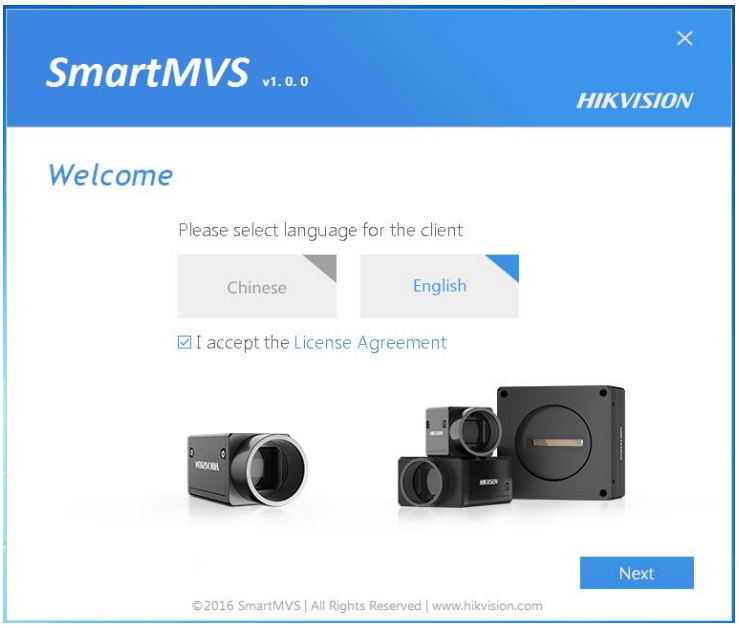
2.3 SmartMVS 客户端安装

SmartMVS 是智能相机的配置客户端,用户可以使用此软件对相机进行各种配置,可以查看各种属性节点的在属性,任何 SmartMVS 上实现的功能,都可以调用 SDK 来实现.开发者可以用 SmartSDK 来看各个参数节点的名称,范围,含义等.

SmartMVS 客户端安装步骤:

(1)双击客户端的安装包(SmartMVS_STD_1.X.X_201XXXXXXX.exe),点击下一步(next)

进入安装界面,选择合适的语言,合理的安装路径,点击下一步.



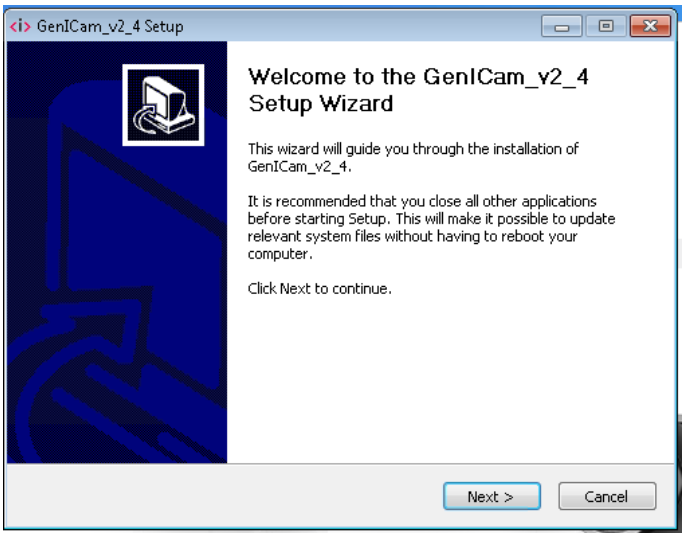
(2):提示安装 GenICam,请检查一下系统中是否有 GenICam v2.4,如果已经存在,则不需要安装,直接选择 Cancel

检查方法:

1. 依次打开:控制面板\所有控制面板项\程序和功能\

2. 查看所有安装的程序, 检查是否有 GenICam v2.4 的程序,如果有多份,请卸载多份,保留一份.

如果 PC 中没有 GenICam 存在,选择 next,正常安装.



(3):安装完毕后,即可正常打开 SmartMVS 软件,

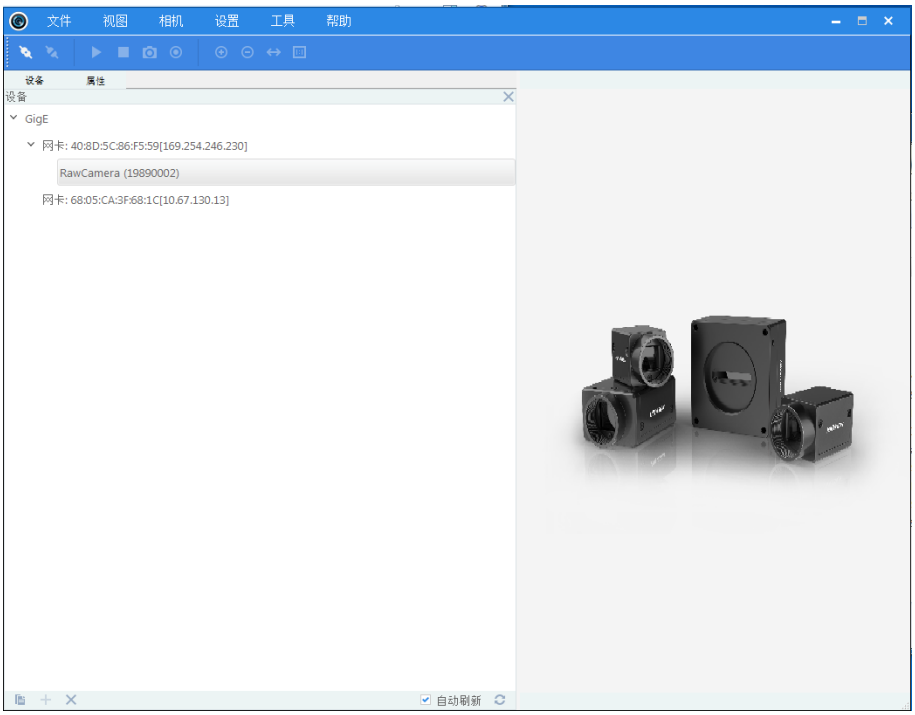
3. SmartMVS 基本使用方法

软件安装完毕后,可以使用 SmartMVS 对相机进行各种配置操作, 对相机的各种操作,具体请参考 <<海康读码智能相机应用手册.docx>>.下面仅介绍两种基本的工作模式.

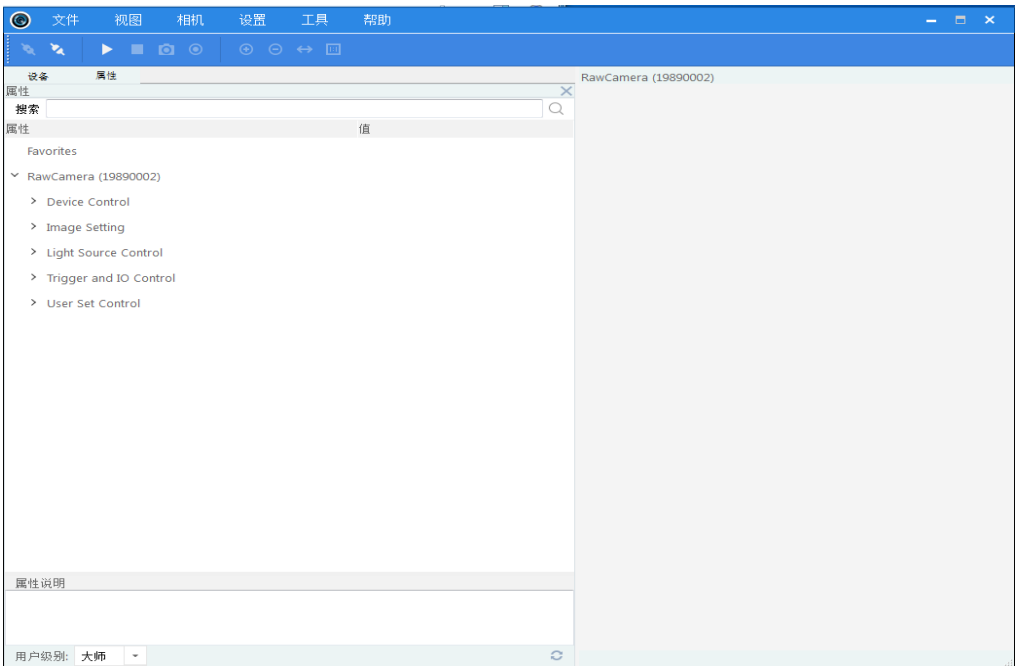
3.1 调试模式

调试模式: 相机持续采集图像,在这个过程中用户可以调节相机的焦距,曝光等参数,使相机出图清新,达到工作的需求.

1. 双击智能相机，连接相机

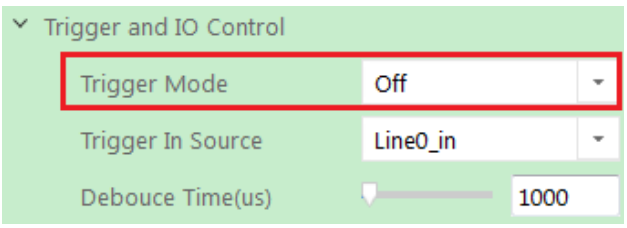


2. 切换到属性页面



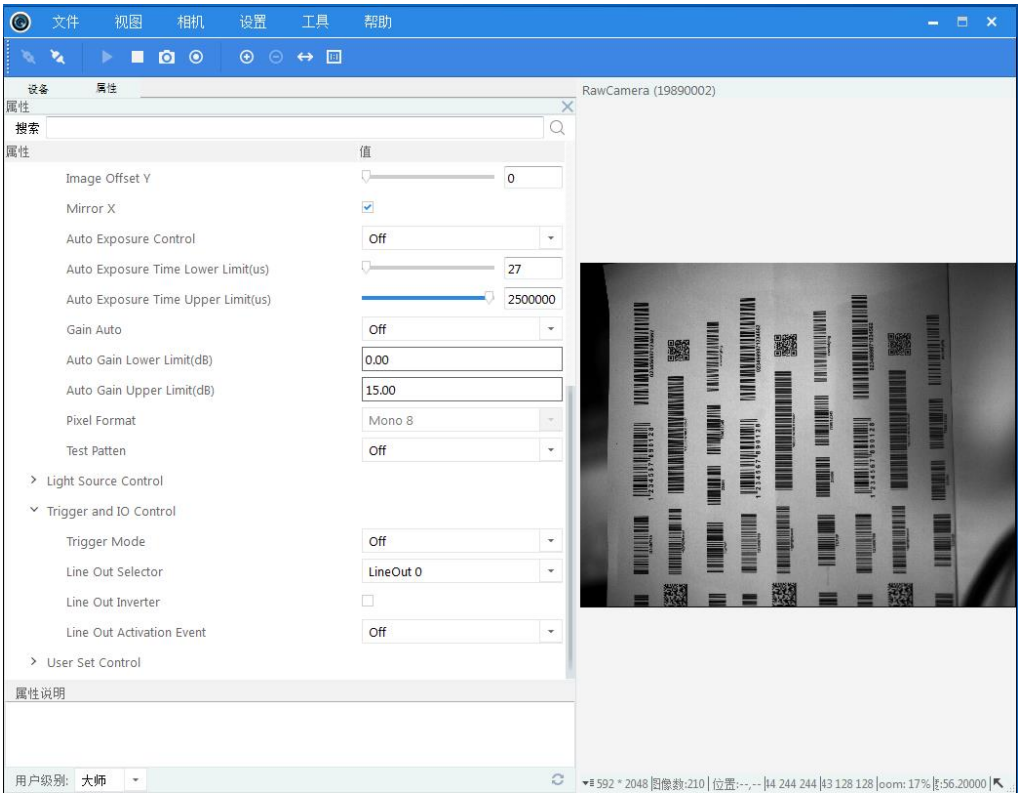
3. 调成像效果

1）触发模式调整为 Off；(off: 关闭触发开关,相机持续取流)



2）开启预览.

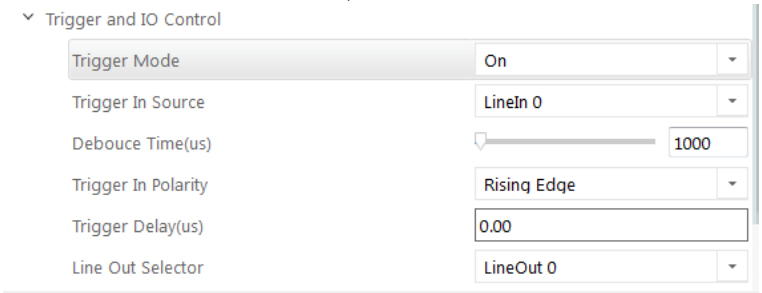
3）调整光圈、焦距、曝光时间、增益、光源等使图像达到要求.



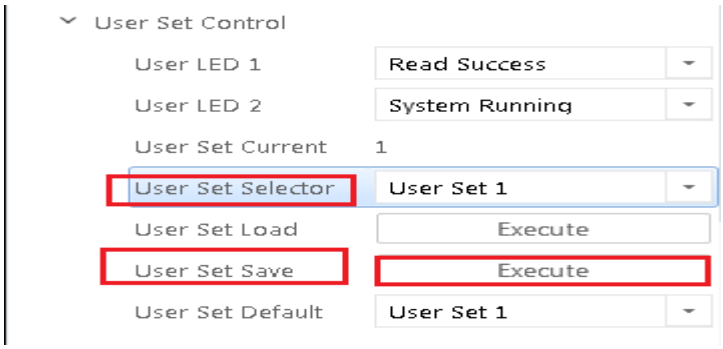
3.2 工作模式

工作模式是指：不是一直在取图,可以配置特定的外部触发,相机收到外部触发信号后，相机才采集图像,输出图像，外部触发信号结束之后，停止采集图像,停止输出图像.

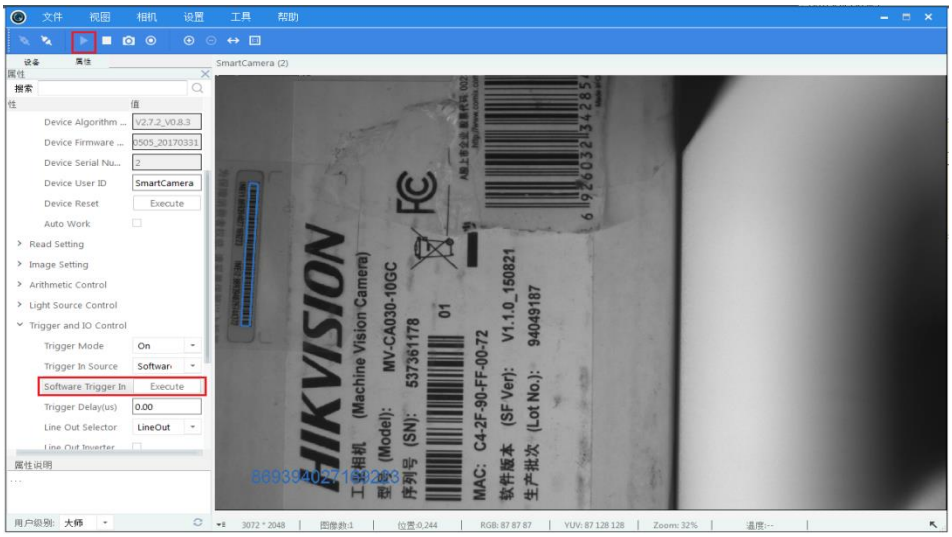
1）使用调试模式,调节图像清晰可用.
2）选择触发模式 ON(选择对应的触发线 LineIn0 等,选择合适的触发策略(上升沿/下降沿/高电平/低电平),合适的最小触发时间(少于这个时间的触发认为是误触发)



3) 配置完毕,保存参数.



4) 开启取流,相机开始工作.



4. 开放平台相机 SDK 开发

4.1 SDK 目录介绍

如上图,是提供的 SDK 开发组件.

lib: 静态库文件

bin:动态库文件

include: 头文件

demo: 使用 Lib Bin Include 文件开发的小的样例.

doc: 文档资料

4.2 添加 SDK 到工程

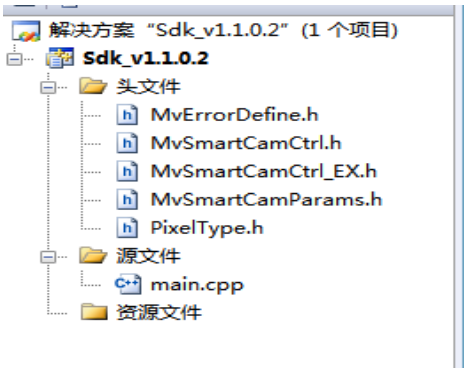
使用 VS2008 把 SDK 添加到对应的工程中的的使用方法.

1. 使用 VS2008 打开需要的项目

2. 把 Include 下的文件添加到工程的头文件目录下

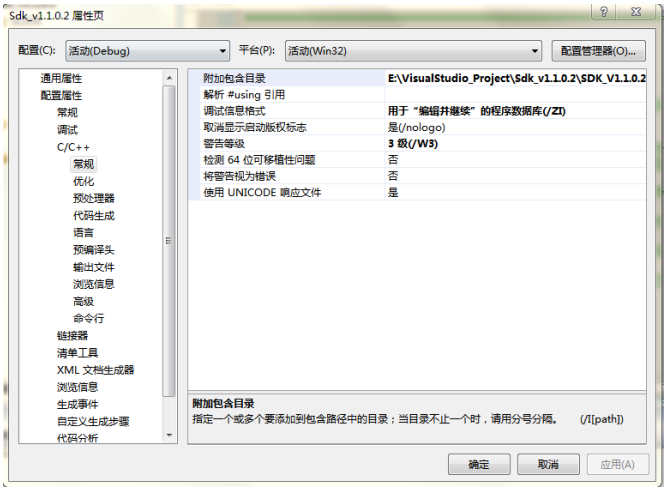
   右键->添加->新建筛选器->添加文件

   右键->添加->添加现有项-→选中 Include 目录下的所有文件.



3. 把 SDK 目录下的 Include 的目录路径添加工程的目录中.

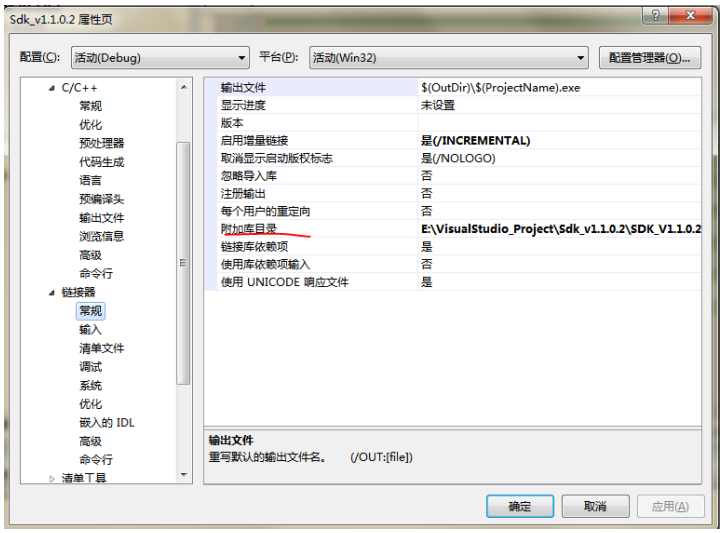
解决方案资源管理器→配置属性→C/C++→常规→附加包含目录



4. 把 SDK 目录下的静态库(Lib)添加工程中

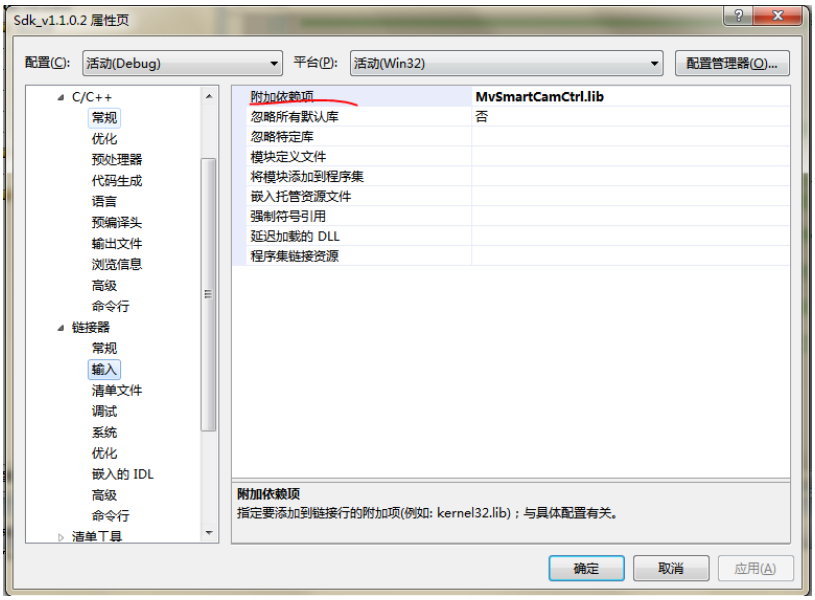
把 Lib 库放到当前工程的附加库的文件夹下或者把 Lib 的目录添加到附加库目录中

附加库目录路径:解决方案资源管理器→配置属性→链接器->常规->附加库目录



5. 添加对应的静态库(Lib 文件夹下)的文件名到工程.

解决方案资源管理器→配置属性→链接器→输入→附加依赖项



6. 把 BIN 下面的文件,放到项目生成的目标程序文件夹下.

注意:

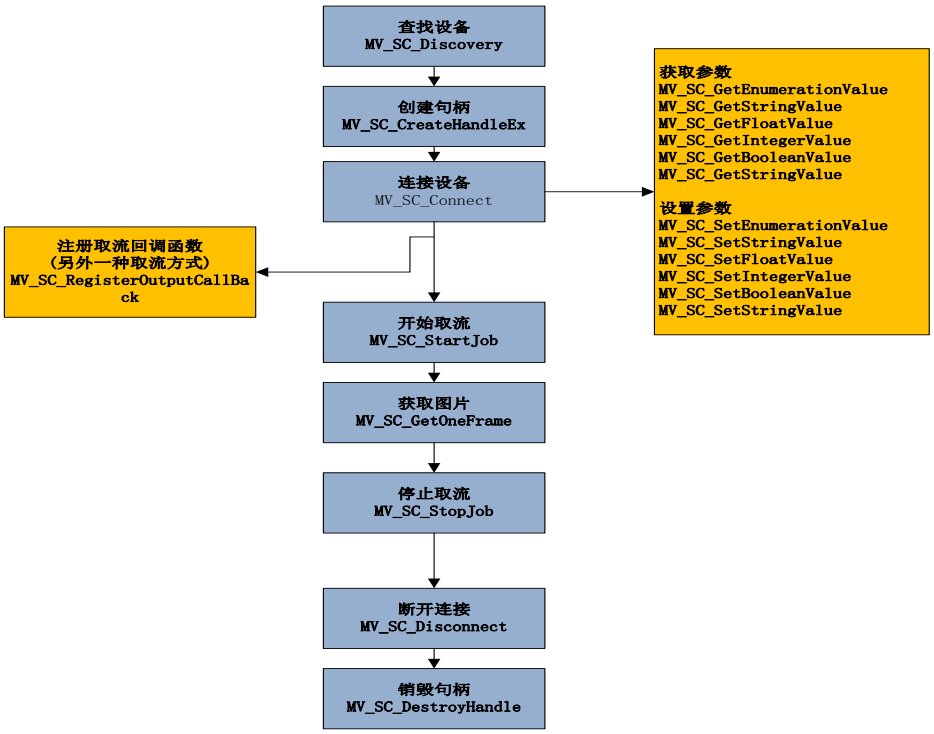
1. 解决方案资源管理器,有 Release 和 Debug 两种配置,如果两种都需要,那么需要都配置.

2. GenICam_VC80_Win32_i86_v2_4_0.exe 此文件在安装 SmartMVS 时,会提示用户进行安装,如果已经安装,则不需要再次安装此文件. 如果当前系统中没有安装过此文件,则需要手动安装

此文件.

4.3 基本的接口调用流程

1. 使用 SDK 直接获取数据的流程



4.4 如何获取和设置参数

1. 使用 SDK 获取参数和设置参数的接口来实现.

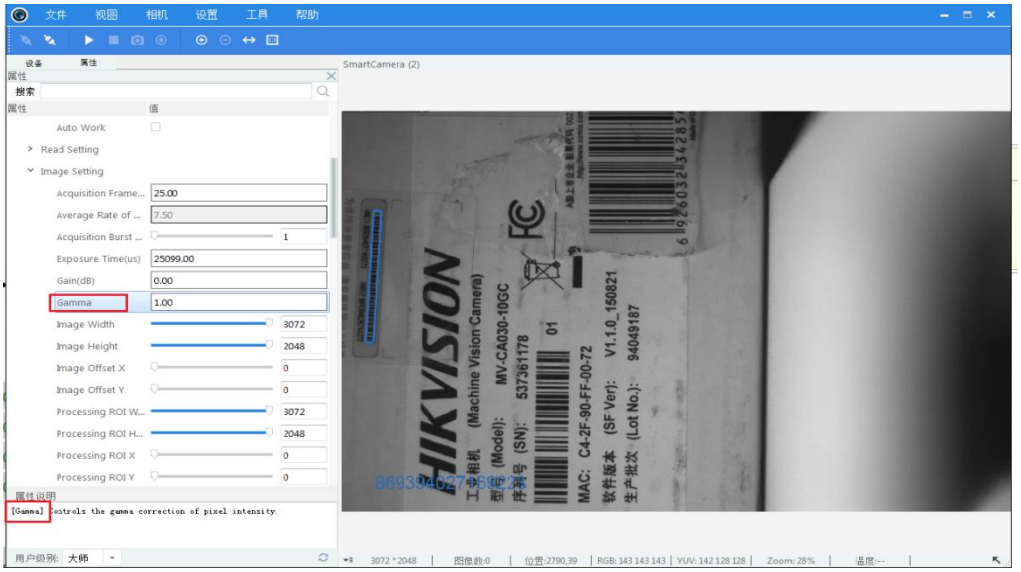
具体设置不同类型参数,有不同的接口:

Eg: MV_SC_GetEnumerationValue/MV_SC_GetStringValue

MV_SC_SetEnumerationValue/MV_SC_SetStringValue

2. SDK 中对应的节点的名称获取方法有:

一): 使用 SmartMVS 点击对应的节点,查看属性说明中的说明,找到对应的节点名称.



二)：参照<<x86 相机节点说明.xlsx>> 找到对应节点的类型,范围,意义.

	AcquisitionFrameRate	Acquisition Frame Rate	IFloat	>0.0，单位fps	RW	取流帧率
	ResultingFrameRate	Resulting Frame Rate	IFloat	>0.0，单位fps	RO	实际fps出图帧率
	AcquisitionBurstFrameCount	Acquisition Burst Frame Count	IInteger	1-1023	RW	一次触发采集图为帧数
	ExposureTime	Exposure Time(us)	IFloat	>0.0，单位us	RW	曝光时间
	Gain	Gain(dB)	IFloat	>0.0，单位dB	RW	增益
	Brightness	Brightness	IInteger	0-255	RW	亮度
	Gamma	Gamma	IFloat	0.0-4.0	RW	Gamma值
	Width	Image Width	IInteger	>320	RW	图像宽度
	Height	Image Height	IInteger	>240	RW	图像高度
	OffsetX	Image Offset X	IInteger	与图像宽度的和不能大于图像宽度最大值	RW	水平偏移
	OffsetY	Image Offset Y	IInteger	与图像高度的和不能大于图像高度最大值	RW	垂直偏移
ImageSetting	ReverseX	Mirror X	IBoolean		RW	水平翻转
	PixelFormat	Pixel Format	IEnumeration	目前只支持Mono8格式	RW	像素格式
	ExposureAuto	Auto Exposure Control	IEnumeration	0: Off 1: Once 2: Continuous	RW	自动曝光模式
	GainAuto	Gain Auto	IEnumeration	0:Off 1:Once 2:Continuous	RW	自动增益模式
	AutoGainLowerLimit	Auto Gain Lower Limit(dB)	IFloat	>0.0，单位dB	RW	自动增益下限值
	AutoGainUpperLimit	Auto Gain Upper Limit(dB)	IFloat	0-15	RW	自动增益上限值
	TestImageModel	Test Patten	IEnumeration	7:off 0:mono bar 3:checkboard 4:oblique mono bar	RW	测试图像

4.5 接口说明

一):获取 SDK 版本信息

```
/*@fn          MV_SC_GetSDKVersion
**@brief       获取当前的SDK版本
**@param       无
**@return      成功：返回MV_OK;失败：返回错误码
*/

MVSMARTCAMCTRL_API unsigned int __stdcall MV_SC_GetSDKVersion();
```

二): 发现设备/连接设备/断开设备

```
/*@fn          MV_SC_Discovery
**@brief       枚举设备
```

```
**@param    pstDevList      设备信息列表结构体指针
**@return   成功：返回MV_OK;失败：返回错误码
*/
MVSMARTCAMCTRL_API int __stdcall MV_SC_Discovery(IN OUT MV_SC_DEVICE_INFO_LIST * pstDevList);


/*@fn       MV_SC_DiscoveryEx
**@brief    枚举设备，扩展接口，可以指定枚举端口号
**@param    pstDevList      设备信息列表结构体指针
**@param    nDevPort        指定设备端口号
**@return   成功：返回MV_OK;失败：返回错误码
*/
MVSMARTCAMCTRL_API int __stdcall MV_SC_DiscoveryEx(IN OUT MV_SC_DEVICE_INFO_LIST * pstDevList,
IN const unsigned short nDevPort);


/*@fn       MV_SC_CreateHandleEx
**@brief    创建设备句柄，扩展接口，可以指定创建设备句柄的端口号
**@param    handle          设备句柄
**@param    pstDevInfo      设备信息结构体指针
**@param    nDevPort        指定设备端口号
**@param    bLogEnable      是否创建Log文件
**@return   成功：返回MV_OK;失败：返回错误码
*/
MVSMARTCAMCTRL_API int __stdcall MV_SC_CreateHandleEx(  OUT void ** handle
                                                      , IN  MV_SC_DEVICE_INFO* pstDevInfo
                                                      , IN  const unsigned short nDevPort
                                                      , IN  const bool bLogEnable);


/*@fn       MV_SC_DestroyHandle
**@brief    销毁设备句柄
**@param    handle          要销毁的设备句柄
**@return   成功：返回MV_OK;失败：返回错误码
*/
MVSMARTCAMCTRL_API int __stdcall MV_SC_DestroyHandle(IN void * handle);


/*@fn       MV_SC_Connect
**@brief    连接设备
**@param    handle          设备句柄
**@return   成功：返回MV_OK;失败：返回错误码
```

```
*/
MVSMARTCAMCTRL_API int __stdcall MV_SC_Connect(IN void * handle);

/*@fn       MV_SC_Disconnect
**@brief    断开连接
**@param    handle          设备句柄
**@return   成功：返回MV_OK;失败：返回错误码
*/
MVSMARTCAMCTRL_API int __stdcall MV_SC_Disconnect(IN void * handle);
```

[关键样例代码:]

```
int main(void)
{
    int nRet = MV_OK;
    int m_nDeviceToOpen = 0;
     MV_SC_DEVICE_INFO_LIST m_pstDeviceInfoList;
     void *   g_pBaseCamerahandle = NULL;
     bool bFindConnectCam = false;
     int nMvScDeviceGroupId = 0;

     //发现网络中的设备
     nRet = MV_SC_DiscoveryEx(&m_pstDeviceInfoList, 3956);
     if (MV_OK != nRet)
     {
          printf("MV_SC_DiscoveryEx failed, errcode is(%d)!\r\n",nRet);
          return nRet;
     }

     unsigned int nCount = 0;
     for (nCount = 0; nCount < m_pstDeviceInfoList.nDeviceNum; nCount ++)
     {
          //找到对应需要的设备” 19890002”
          if(0==strncmp("19890002"
,(const char *)m_pstDeviceInfoList.astDeviceInfo[nCount].chSerialNumber,
strlen("19890002")))
          {
               nMvScDeviceGroupId = nCount;
               bFindConnectCam = true;
               break;
          }
     }

      //没有找到设备,返回
```

```
    if (true != bFindConnectCam)
    {
        //释放资源
        return MV_E_NODATA;
    }

    // 创建裸相机的 Handle  不创建日志
nRet=MV_SC_CreateHandleEx(&g_pBaseCamerahandle, &m_pstDeviceInfoList.astDeviceInfo[nMvScDeviceGroupId],
3956,
true);
    if (MV_OK != nRet)
    {
        printf("Base camera create handle failed, errcode is [%#x]\r\n", nRet);
        //释放资源
        return nRet;
    }

    // 连接相机
    nRet = MV_SC_Connect(g_pBaseCamerahandle);
    if (MV_OK != nRet)
    {
        printf("Failed connect camera, errcode is [%#x]\r\n", nRet);
        //释放资源
        return nRet;
    }

// 中间可以在此处进行各种相机的操作
// 获取参数,设置参数,读取配置等.

    // 断开相机
    nRet = MV_SC_Disconnect(g_pBaseCamerahandle);
    if (MV_OK != nRet)
    {
        printf("Failed disconnect camera, errcode is [%#x]\r\n", nRet);
//释放资源
        return nRet;
    }

    // 销毁句柄
    nRet = MV_SC_DestroyHandle(g_pBaseCamerahandle);
    if (MV_OK != nRet)
    {
        printf("Failed destroy handle, errcode is [%#x]\r\n", nRet);
        return nRet;
```

```
    }
    return nRet;
}
```

三): 读取参数/设置参数

读取和设置Integer类型节点的接口如下:

```
/*@fn        MV_SC_GetIntegerValue
**@brief     获取Integer型参数值
**@param     handle            设备句柄
**@param     strName           参数对应的XML节点名称
**@param     pnValue           返回的参数值指针
**@return    成功：返回MV_OK;失败：返回错误码
*/
MVSMARTCAMCTRL_API int __stdcall MV_SC_GetIntegerValue(IN void * handle, IN const char * strName,
IN OUT unsigned int * pnValue);
```

```
/*@fn        MV_SC_SetIntegerValue
**@brief     设置Integer型参数值
**@param     handle            设备句柄
**@param     strName           参数对应的XML节点名称
**@param     nValue            要设置的参数值
**@return    成功：返回MV_OK;失败：返回错误码
*/
MVSMARTCAMCTRL_API int __stdcall MV_SC_SetIntegerValue(IN void * handle, IN const char * strName,
IN unsigned int nValue);
```

[关键样例代码:]

```
{
// 发现设备
// 创建句柄
// 连接设备

    unsigned int nImageWidth = 0;
    //获取图像的宽度
    nRet = MV_SC_GetIntegerValue(g_pBaseCamerahandle,"Width",&nImageWidth);
    if (MV_OK != nRet)
    {
        printf("Failed get width from camera, errcode is [%#x]\r\n", nRet);
        //释放资源
```

```
        return nRet;
    }

    //更新图像的宽度
    nRet = MV_SC_SetIntegerValue(g_pBaseCamerahandle,"Width",(nImageWidth-100));
    if (MV_OK != nRet)
    {
        printf("Failed get width from camera, errcode is [%#x]\r\n", nRet);
        //释放资源
        return nRet;
    }


//断开连接
//销毁句柄
}
```

三): 取流/停止取流/获取流信息

```
/*@fn        MV_SC_StartJob
**@brief     开始工作流程
**@param     handle          设备句柄
**@return    成功：返回MV_OK;失败：返回错误码
*/
MVSMARTCAMCTRL_API int __stdcall MV_SC_StartJob(IN void * handle);

/*@fn        MV_SC_StopJob
**@brief     停止工作流程
**@param     handle          设备句柄
**@return    成功：返回MV_OK;失败：返回错误码
*/
MVSMARTCAMCTRL_API int __stdcall MV_SC_StopJob(IN void * handle);


/*@fn        MV_SC_GetOneFrame
**@brief     获取一帧图像数据
**@param     handle          设备句柄
**@param     pData           获取的图像数据指针
**@param     pData           图像数据缓存区大小
**@param     pstImageInfo    图像帧信息结构体指针
**@return    成功：返回MV_OK;失败：返回错误码
*/
MVSMARTCAMCTRL_API int __stdcall MV_SC_GetOneFrame(IN void * handle,
```

```
                                                    IN OUT unsigned char * pData ,
                                                    IN unsigned int nDataSize,
                                                    IN OUT MV_SC_IMAGE_OUT_INFO * pstImageInfo);
```

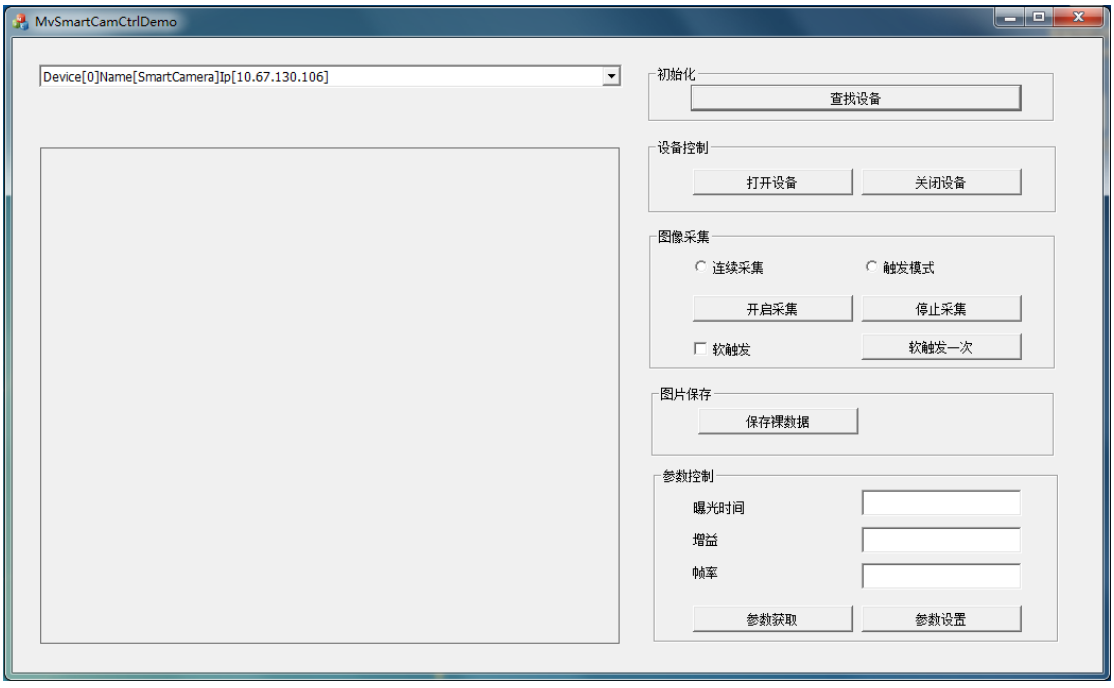
[关键代码]

```
{
    //开始取流
    nRet = MV_SC_StartJob(g_pBaseCamerahandle);
    if (MV_OK != nRet)
    {
        printf("Failed start job, errcode is [%#x]\r\n", nRet);
        //释放资源
        return nRet;
    }
    MV_SC_IMAGE_OUT_INFO    stImageInfo = {0};
    int    nDataSize =              7*1024*1024;
    unsigned char *pData = (unsigned char *) malloc(nDataSize) ;
    MVSC_NETTRANS_INFO    stNetTransInfo = {0};
    //用户可在此处循环操作
    //获取图像数据
    nRet = MV_SC_GetOneFrame(g_pBaseCamerahandle,
                                    pData ,
                                    nDataSize,
                                    &stImageInfo);
        if (MV_OK != nRet)
        {
            printf("Failed Get One Frame, errcode is (%x)!\r\n",nRet);
            //释放资源
            return nRet;
        }

   // 图像宽
    printf("Get one fram success, and get image Width is (%d)!\r\n",stImageInfo.nWidth);

    // 图像高
    printf("Get one fram success, and get image Height is (%d)!\r\n",stImageInfo.nHeight);

    // 触发序号（仅在电平触发时有效）
    printf("Get one fram success, and get image TriggerIndex is (%d)!\r\n",stImageInfo.nTriggerIndex);

    // 帧号
    printf("Get one fram success, and get image FrameNum is (%d)!\r\n",stImageInfo.nFrameNum);
```

```
    // 当前帧数据大小
    printf("Get one fram success, and get image FrameLen is (%d)!\r\n",stImageInfo.nFrameLen);

    //停止取流
    nRet = MV_SC_StopJob(g_pBaseCamerahandle);
    if (MV_OK != nRet)
    {
        printf("Failed stop job, errcode is [%#x]\r\n", nRet);
        //释放资源
        return nRet;
    }
}
```

4.6 实例程序说明

目前提供 VC/C#的样例 demo,该示例程序包含了相机开发的基本流程和常见接口调用。界面如下：



5. 常见问题

5.1 问题排查思路

1. SmartMVS 看图像很慢

用 Wireshark 抓包,检查码流对应数据包的长度,根据数据包的长度,判断巨帧是否有效.

2. 用 SmartMVS 配置参数错误,取流失败.

检查 SmartMVS 和相机的版本是否匹配.

5.2 典型问题解决方法

1. 开启预览，没有图像；

1） 开启了触发模式，但实际没有触发信号；

2. 开启预览，图像是黑的

1） 曝光时间设置的过小，调节曝光时间可以改善；

2） 调节增益；

3） 调光圈；

3. 开启预览，图像质量差

1） 检查 pc 是否已开启了巨帧；

2） 检查 pc 的本地速度是否是 1Gbps；