工业相机 SDK 二次开发示例程序说明（VC6.0 版）

【摘要】

本文档主要介绍了使用工业相机 SDK(Software Development Kit)开发 C++程序方法及过程。在 SDK 开发包目录下，提供了 13 个 VC6.0 示例程序，其中 MFC 程序 5 个，分别为 BasicDemo、ReconnectDemo、SetIODemo、ForceIpDemo、MultipleCamera；控制台程序 8 个，分别为 ConnectSpecCamera、ConvertPixelType、Events、Grab_Callback、GrabImage、MultiCast、ParametrizeCamera_FileAccess、ParametrizeCamera_LoadAndSave。这些示例程序展示了工业相机 SDK 的各个接口的调用方式。

本文档就这五个 MFC 示例程序的操作方法和开发流程展开讨论，介绍各个示例程序的使用步骤和开发流程，方便用户快速入门使用 C++的 SDK。

【注意】

C++版示例程序兼容中英文，对关键的程序会有中英文的注释，且界面控件支持中英文根据安装时操作系统，可通过切换 Dialog 实现。

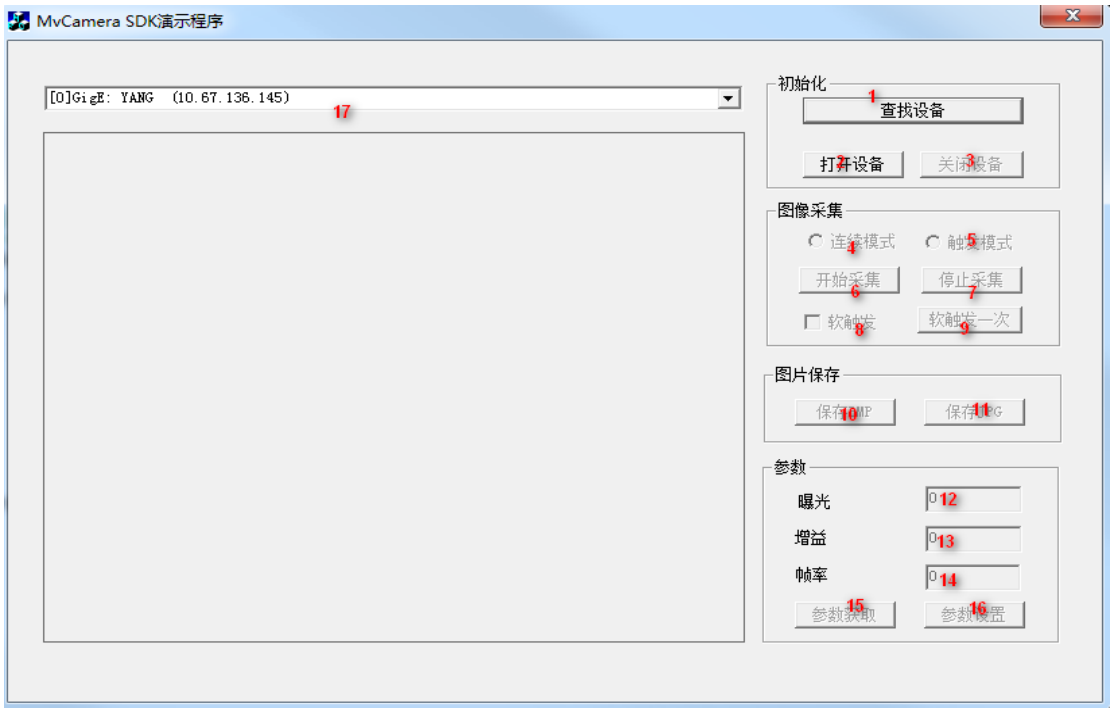
一．BasicDemo 使用步骤及开发流程

BasicDemo 是一个基本示例程序，包含了 SDK 使用过程中常用的一些接口调用，初次使用工业相机 SDK 进行二次开发的用户推荐首先参考 BasicDemo，其涵盖了大多数用户对 SDK 的使用方法示例需求。

1.1 Demo 软件使用步骤

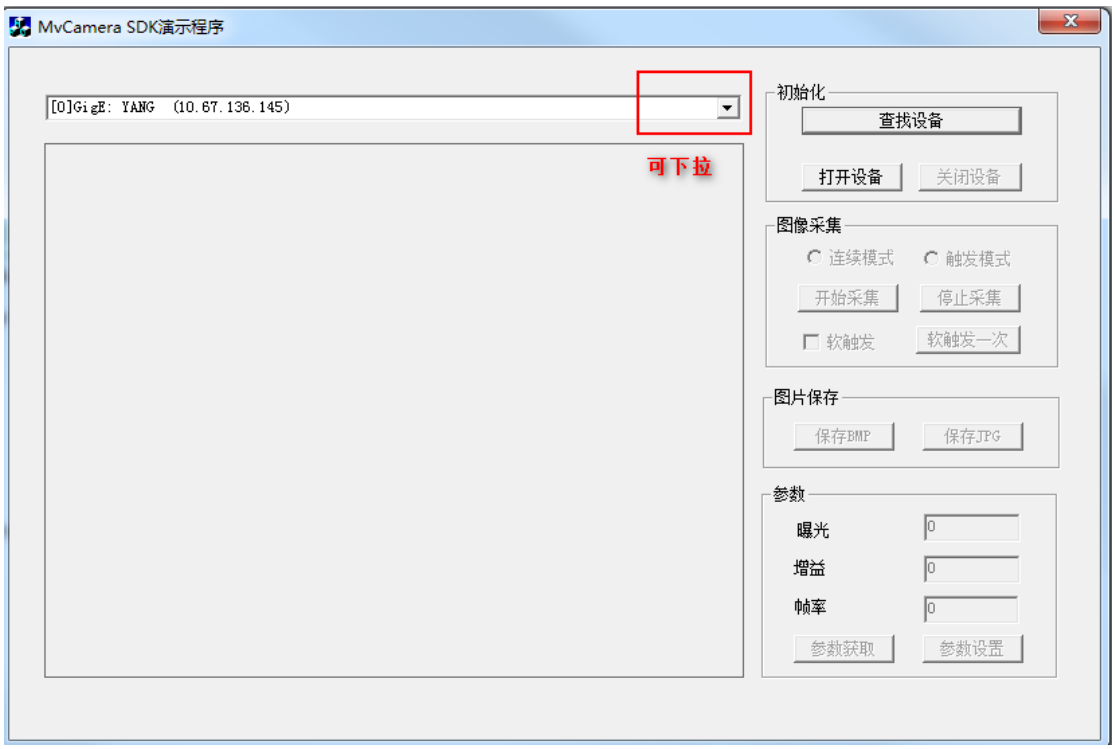
1.1.1 界面总体

软件界面总览，一共包括四个控制模块(初始化，图像采集，图片保存，参数控制)、一个下拉设备列表和一个图像显示区域

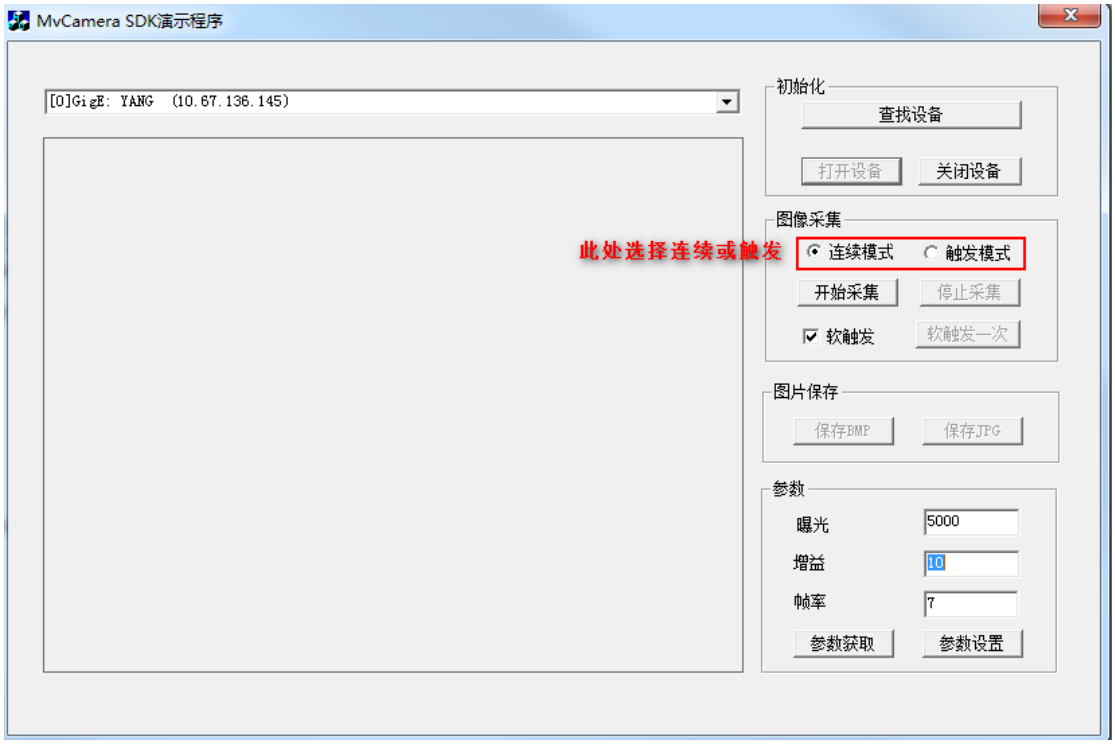


1.1.2 使用过程

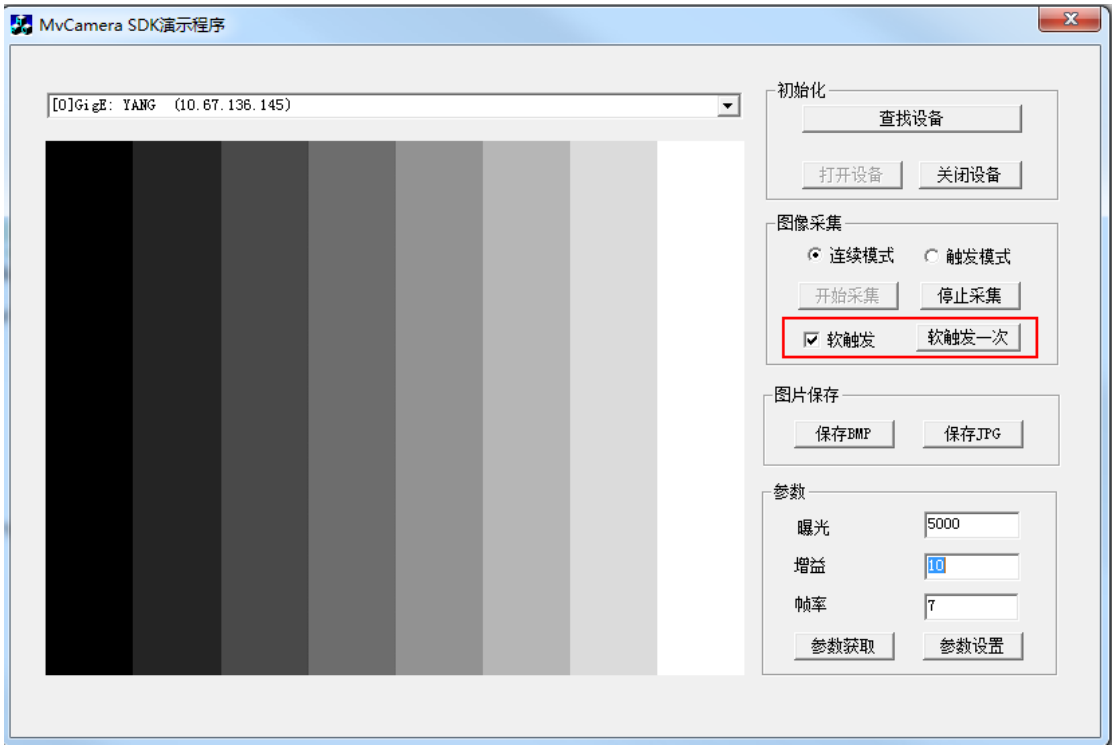
点击【查找设备】进行查找设备，这时（17）会出现当前在线的设备列表，命名方式为用户 ID 不为空时显示设备类型+设备名称+IP 地址，设备为空时显示设备类型+设备型号+IP 地址。选择其中一个设备



点击【打开设备】打开当前选中的设备，默认以连续方式打开设备。选择触发模式可以选中触发模式单选框。



在触发模式下，可以设置为软触发，当点击【开始采集】后，同时【软触发一次】也是可以点击从而完成触发一次功能



采用连续模式下，点击【开始采集】进行图像采集，左边的显示区域将会出现实时图像此时，若点击【保存 BMP】或者【保存 JPG】，将会在当前 exe 目录下出现一个 bmp

或 jpg 类型的图片，即为保存的当前图像

点击【获取参数】将会刷新当前的曝光时间、增益和帧率的数值，而更改【曝光】、【增益】、【帧率】的数值之后点击【设置参数】将会重新设置新的曝光时间、增益和帧率的数值

在使用过程中有任何异常或错误，都会以弹窗的形式出现提示，若没有任何提示，则认为一切正常地运行。

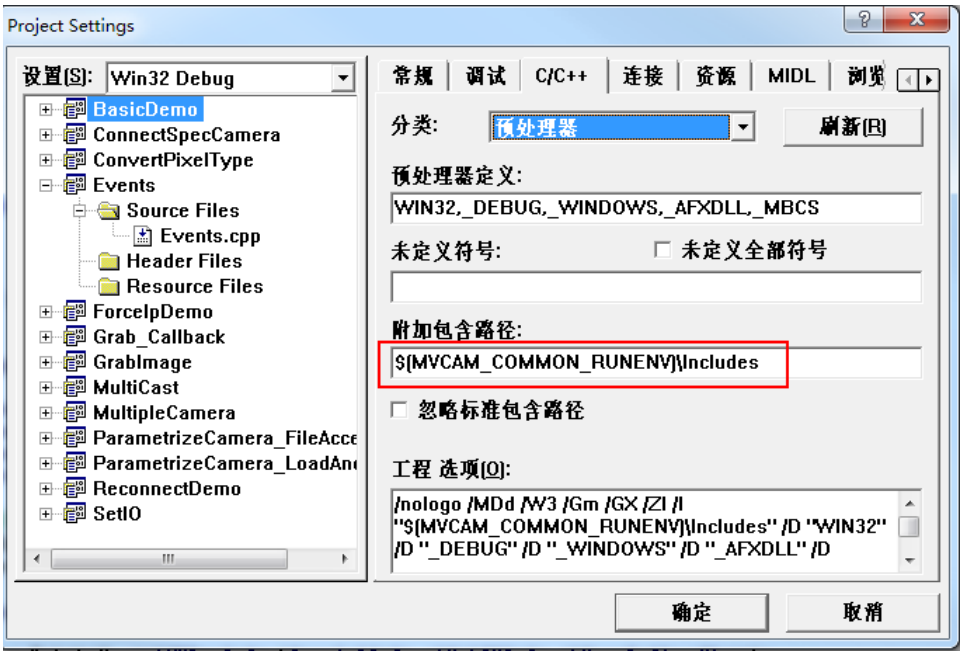
1.2 Demo 软件开发步骤

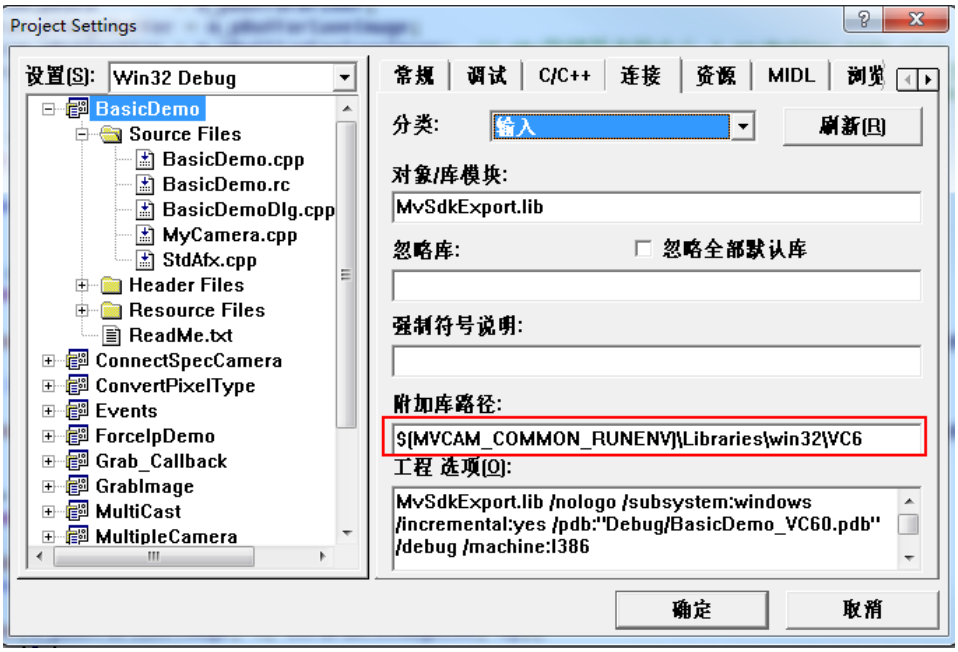
1.2.1 Dll 加载

安装好 MVS 的同时会把相应 32 和 64 的 dll 打到环境变量。

1.2.2 工程配置

创建 VS 工程并添加引用，加入 MvSdkExport.lib 和 MvSdkExport.h 到工程中。





1.2.3 引用命名空间

添加头文件和库文件引用之后，就可以调 MvSdkExport.h 中相机操作的函数。

```
/************************************************************************/
/* 相机的基本指令和操作                                                 */
/************************************************************************/
MVSDKEXPORT_API unsigned int __stdcall MV_CC_GetSDKVersion();
MVSDKEXPORT_API int __stdcall MV_CC_EnumerateTls();
MVSDKEXPORT_API int __stdcall MV_CC_EnumDevices(IN unsigned int nTLayerType, IN OUT MV_CC_DEVICE_INFO_LIST* pstDevList);
MVSDKEXPORT_API int __stdcall MV_CC_EnumDevicesEx(IN unsigned int nTLayerType, IN OUT MV_CC_DEVICE_INFO_LIST* pstDevList, IN const char* pManufacturerName);
MVSDKEXPORT_API bool __stdcall MV_CC_IsDeviceAccessible(IN MV_CC_DEVICE_INFO* pstDevInfo, IN unsigned int nAccessMode);
MVSDKEXPORT_API int __stdcall MV_CC_SetSDKLogPath(IN const char * pSDKLogPath);
MVSDKEXPORT_API int __stdcall MV_CC_CreateHandle(OUT void ** handle, IN const MV_CC_DEVICE_INFO* pstDevInfo);
MVSDKEXPORT_API int __stdcall MV_CC_CreateHandleWithoutLog(OUT void ** handle, IN const MV_CC_DEVICE_INFO* pstDevInfo);
MVSDKEXPORT_API int __stdcall MV_CC_DestroyHandle(IN void * handle);
MVSDKEXPORT_API int __stdcall MV_CC_OpenDevice(IN void* handle,IN unsigned int nAccessMode, IN unsigned short nSwitchoverKey);
MVSDKEXPORT_API int __stdcall MV_CC_CloseDevice(IN void* handle);
MVSDKEXPORT_API int __stdcall MV_CC_RegisterImageCallBackEx(void* handle, void(__stdcall* cbOutput)(unsigned char * pData, MV_FRAME_OUT_INFO_EX* pFrameInfo, void* pUser),void* pUser);
MVSDKEXPORT_API int __stdcall MV_CC_RegisterImageCallBackForRGB(void* handle, void(__stdcall* cbOutput)(unsigned char * pData, MV_FRAME_OUT_INFO_EX* pFrameInfo, void* pUser),void* pUser);
MVSDKEXPORT_API int __stdcall MV_CC_RegisterImageCallBackForBGR(void* handle, void(__stdcall* cbOutput)(unsigned char * pData, MV_FRAME_OUT_INFO_EX* pFrameInfo, void* pUser), void* pUser);
MVSDKEXPORT_API int __stdcall MV_CC_StartGrabbing(IN void* handle);
MVSDKEXPORT_API int __stdcall MV_CC_StopGrabbing(IN void* handle);
MVSDKEXPORT_API int __stdcall MV_CC_GetImageForRGB(IN void* handle, IN OUT unsigned char * pData , IN unsigned int nDataSize, IN OUT MV_FRAME_OUT_INFO_EX* pFrameInfo, int nMsec);
MVSDKEXPORT_API int __stdcall MV_CC_GetImageForBGR(IN void* handle, IN OUT unsigned char * pData , IN unsigned int nDataSize, IN OUT MV_FRAME_OUT_INFO_EX* pFrameInfo, int nMsec);
MVSDKEXPORT_API int __stdcall MV_CC_GetOneFrameTimeout(IN void* handle, IN OUT unsigned char * pData , IN unsigned int nDataSize, IN OUT MV_FRAME_OUT_INFO_EX* pFrameInfo, unsigned int nMsec)
MVSDKEXPORT_API int __stdcall MV_CC_Display(IN void* handle, void* hWnd);
MVSDKEXPORT_API int __stdcall MV_CC_SetImageNodeNum(IN void* handle, unsigned int nNum);
MVSDKEXPORT_API int __stdcall MV_CC_GetImageInfo(IN void* handle, IN OUT MV_IMAGE_BASIC_INFO* pstInfo);
MVSDKEXPORT_API int __stdcall MV_CC_GetDeviceInfo(IN void * handle, IN OUT MV_CC_DEVICE_INFO* pstDevInfo);
MVSDKEXPORT_API int __stdcall MV_CC_GetAllMatchInfo(IN void* handle, IN OUT MV_ALL_MATCH_INFO* pstInfo);
MVSDKEXPORT_API int __stdcall MV_CC_RegisterImageBuffer(void* handle, unsigned char* pBuf, unsigned int nBufSize);

/************************************************************************/
/* 设置和获取相机参数的万能接口                                         */
/************************************************************************/
MVSDKEXPORT_API int __stdcall MV_CC_GetIntValue(IN void* handle,IN const char* strValue,OUT MVCC_INTVALUE *pIntValue);
MVSDKEXPORT_API int __stdcall MV_CC_SetIntValue(IN void* handle,IN const char* strValue,IN unsigned int nValue);
```

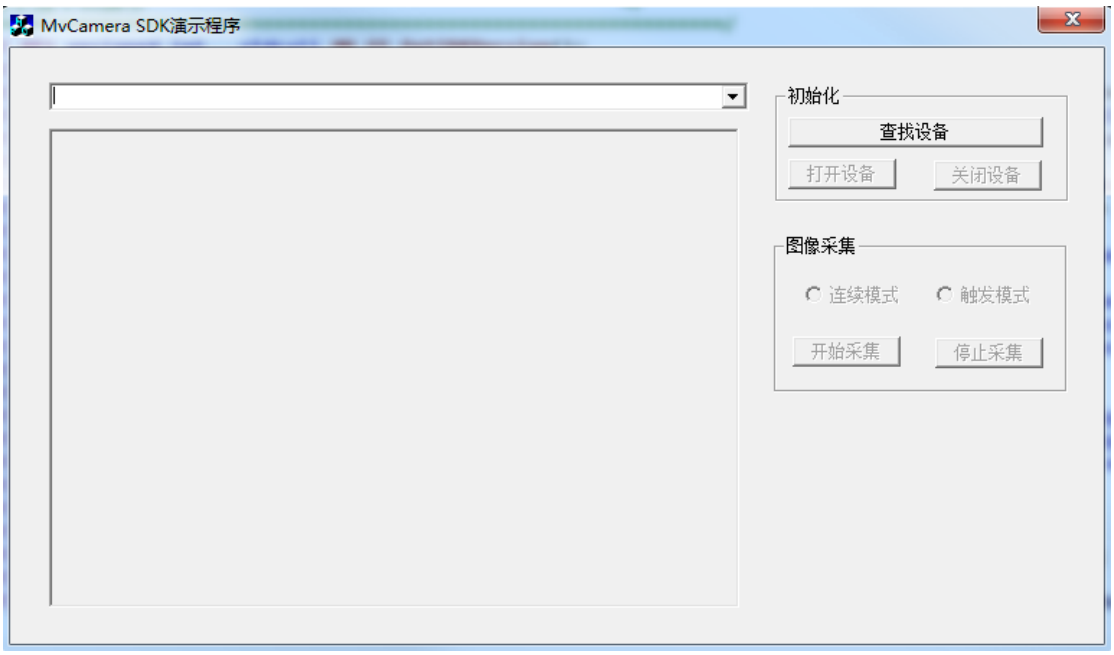
二．ReconnectDemo 使用步骤及开发流程

ReconnectDemo 重点展示了 SDK 中相机断线重连的操作步骤。告知用户如何使用断线回调以及如何重新连接相机。

2.1 Demo 软件使用步骤

2.1.1 界面总体

总体界面如下图。界面类似 BasicDemo，具有查找设备、打开设备、关闭设备、开始采集、停止采集、设置触发等功能。



2.1.2 使用过程

ReconnectDemo 中，当相机断线时，程序会进入异常回调，异常回调中，会根据当前选中的相机信息进行不断的尝试连接，当相机在线时则会被连接上。

2.2 Demo 软件开发步骤

关于相机操作的开发流程与 BasicDemo 相似。本节重点介绍回调函数的使用方法。

在 C++中，通过传函数指针实现回调功能。在工业相机 C++ SDK 中，异常断线的回调为 RegisterExceptionCallBack。

在 CBasicDemoDlg 类中实现断线重连的函数 ReconnectDevice，然后传入回调函数 RegisterExceptionCallBack 中，当打开相机操作之后，利用 SDK 中注册回调函数接口，注册回调函数。当相机异常断线时，程序会进入异常回调。用户可在异常回调中进行重新连接相机的操作。注册过程如下：

```
m_pcMyCamera->RegisterExceptionCallBack(ReconnectDevice, this);
```

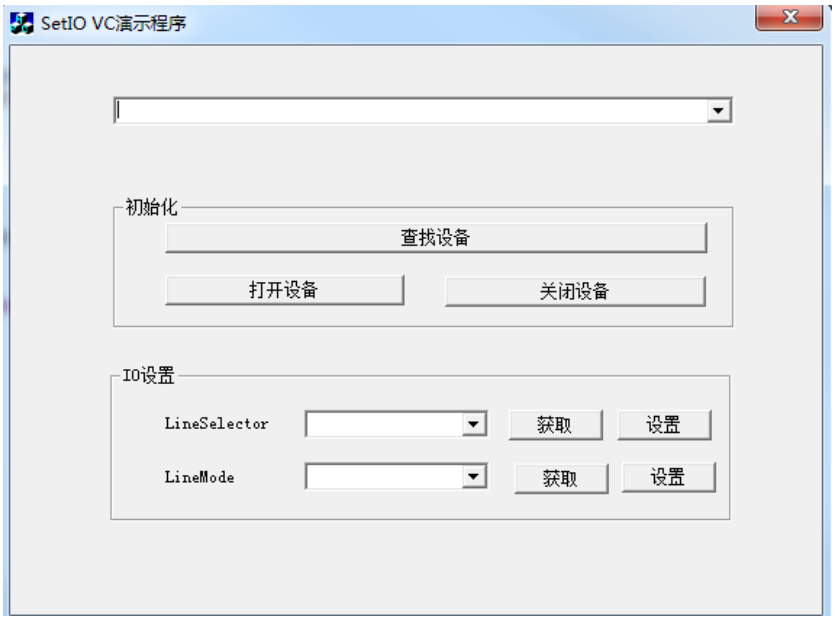
SetIODemo 使用步骤及开发流程

本节介绍的 Demo 主要实现对相机 IO 输入输出的控制。使用用户群体为需要对相机 IO 进行控制的用户。

3.1 Demo 软件使用步骤

3.1.1 界面总体

总体界面如下图所示。



3.1.2 使用过程

相机基本操作与 BasicDemo 相似。打开一个设备后可以对相机的 IO 属性进行获取和设置。IO 属性主要有 LineSelector 和 LineMode 两个。分别点击获取和设置可以对相应的属性进行读取和写入。

3.2 Demo 软件开发步骤

3.2.1 IO 属性

有关相机 IO 属性主要有两个：LineSelector 和 LineMode。LineSelector 指输出端口选择，目前相机主要有三个 IO 端口：Line0，Line1，Line2.其中，Line0 只可配置为输入，Line1 只可配置为输出，Line2 可配置为输入或者输出。LineMode 表示输入或者输出模式。

3.2.2 获取和设置接口

在示例程序中，获取和设置 IO 用到的接口分别为：MV_CC_GetEnumValue(IN void* handle,IN const char* strKey,OUT MVCC_ENUMVALUE *pEnumValue)，以及 MV_CC_SetEnumValue(IN void* handle,IN const char* strKey,IN unsigned int nValue)。

在 SDK 中，类似此类 Set 或 Get + 数据类型 + Value 的接口函数成为万能接口函数，其作用为获取或设置相机任意属性值。万能接口的第一个参数为属性名称，为一个 string 型字符串，相机属性名称可以通过查找 MvCameraNode.xls 文档查询。第二个参数为获取到的或者设置的属性值。

3.2.3 IO 操作

在本节示例程序中，主要用到的属性节点为”LineSelector”以及”LineMode”，其属性类型均为 Enumeration 类型。调用万能接口即可实现对其属性的操作。

获取操作如下：

```
nRet = m_pcMyCamera->GetEnumValue("LineSelector", &stSelector);
nRet = m_pcMyCamera->GetEnumValue("LineMode", &stSelector);
```

设置操作如下：

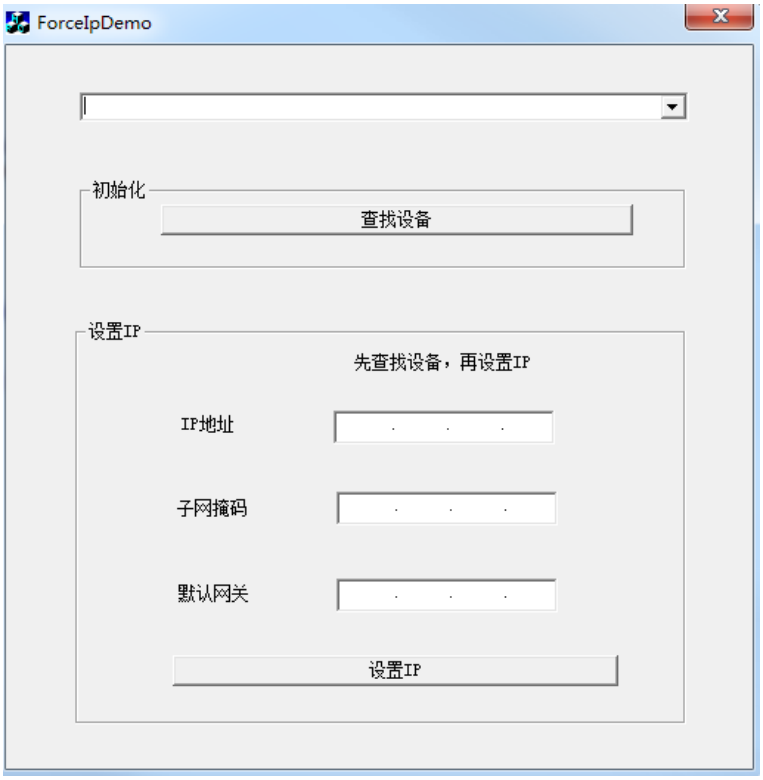
```
nRet = m_pcMyCamera->SetEnumValue("LineSelector", nValue);
nRet = m_pcMyCamera->SetEnumValue("LineMode", nValue);
```

三. ForceIpDemo 使用步骤和开发流程

4.1 Demo 软件使用步骤

4.1.1 界面总体

软件界面如下图所示。



界面主要分为两个模块：初始化模块和设置 IP 模块。

4.1.2 使用过程

首先，点击查找设备对网段内的设备进行枚举，软件自动选择列表中第一项。

然后，选择需要配置 IP 的设备。

在设置 IP 模块的提示信息中会提示本机网卡所在的网段并显示建议设置的 IP 范围。在输入框中输入想要设置的 IP，点击设置。

4.2 Demo 软件开发步骤

设置 IP 调用 SDK 中 MV_GIGE_ForceIpEx(IN void* handle, unsigned int nIP, unsigned int nSubNetMask, unsigned int nDefaultGateWay)接口。

四．MultipleDemo 使用步骤及开发流程

5.1 Demo 软件使用步骤

5.1.1 界面总体

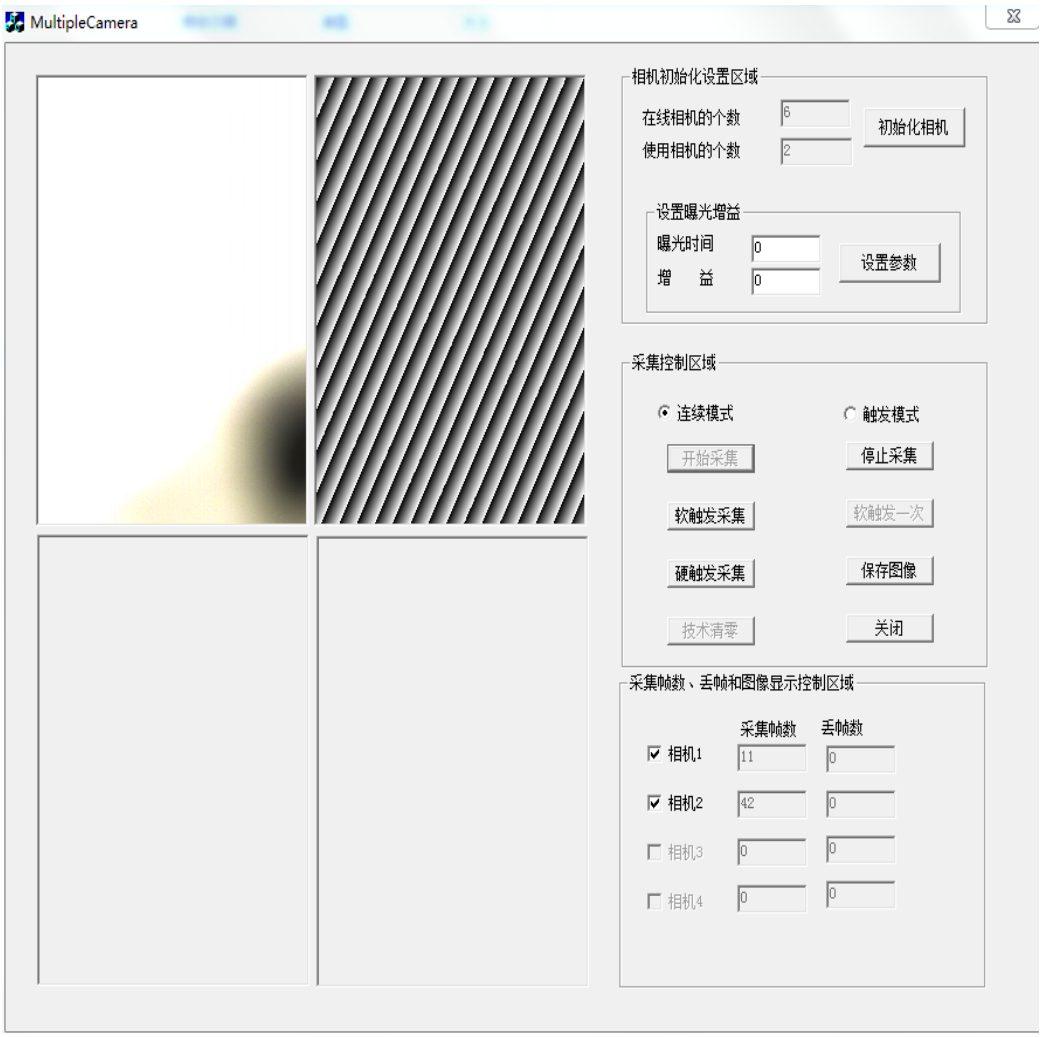
总览界面,软件界面主要包括三个控制模块(初始化、参数设置、采集图像)，四块图像显示区域以及帧数信息显示区域。

5.1.2 使用过程

打开软件，“在线设备数量”会自行枚举在线相机个数，在“使用设备个数”文本框内填写需要打开的相机个数 n，单击“初始化相机”，默认以连续方式打开 n 台设备。

在“曝光”和“增益”中填写修改的参数，单击“设置参数”，即可依次修改 n 台设备参数。同时可选择连续或者触发模式。

点击“开始采集”，左侧会显示预览图像。同时采集帧数和丢帧数会即时更新数据（1 秒更新一次）。此时若点击“保存图片”，会在当前 exe 目录下出现所保存的图片。若希望结束，则点击“停止采集”，“关闭设备”即可。



当出现异常和错误时，会以弹窗的形式提示。有一些操作成功时也会有提示。

5.2 Demo 软件开发步骤

5.2.1 多相机的实现

MultipleDemo 在 BasicDemo 基础上，在类中添加 m_nUseNumEdit 的成员变量，表示四台相机的序号，初始化时由“使用数量”和对应相机打开的序列号决定是否对某台相机执行操作。

5.2.2 总帧数、丢帧数、保存图片

总帧数在回调函数中计数（成员变量）。回调函数中同时完成保存图片的功能，判断是否点击保存图片的按钮确定是否保存当前帧为图片，保存完成后，修改相应标志位以免下次取流重复保存图片。丢帧数由调用

m_pcMyCamera[nUsingDeviceNum]->GetAllMatchInfo(&struMatchInfo)接口获取。